Senior Design Project:

Cognitive Assistance with LiDAR Localization (C-ALL)

by

Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee

sshah6@stevens.edu, nmistry5@stevens.edu, sgaber@stevens.edu, schatte1@stevens.edu

May 24, 2025

sshah6@stevens.edu, nmistry5@stevens.edu, sgaber@stevens.edu, schatte1@stevens.edu

Senior Design Project:
Cognitive Assistance with LiDAR Localization (C-ALL)

Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee
sshah6@stevens.edu, nmistry5@stevens.edu, sgaber@stevens.edu, schatte1@stevens.edu

This document provides the requirements and design details of the C-ALL (Cognitive Assistance with LiDAR Localization) Project. The following table (Table 1) is updated by the authors when major changes are made to the architecture design or new components are added. Updates are added to the top of the table. Most recent changes to the document are seen first and the oldest last.

Table 1: Document Update History

Date	**Updates**
05/02/2025	Usability Testing, Conclusion, Software Design and Implementation, Appendix, Cover Page, System Architecture, Hardware Device Implementation: • Completed Usability Testing (Chapter 12) (NM) • Completed Conclusion (Chapter 13) (NM) • Completed Software Design and Implementation (Chapter 10) (NM) • Completed Appendix C (NM) • Updated Cover Page (NM) • Updated System Architecture (Chapter 8) (NM) • Completed Hardware Device Implementation (Chapter 11) (NM)
04/25/2025	Weekly Report 26, Software Design and Implementation: • Completed Weekly Report 26 (Chapter A) (NM) • Updated Software Design and Implementation (Chapter 10) (NM)
04/18/2025	Weekly Report 25, Report Cover Page, Use Cases, Software Design and Implementation: • Completed Weekly Report 25 (Chapter A) (NM) • Updated Report Cover Page to Include Logo (NM) • Updated Requirements naming references for use case tables (Chapter 6) (NM) • Updated Software Design and Implementation (Chapter 10) (NM)

Table 1: Document Update History

Date	Updates
04/11/2025	Weekly Report 24, Conclusion, Software Design and Implementation, Glossary, Bibliography: • Completed Weekly Report 24 (Chapter A) (NM) • Updated Conclusion (Chapter 13) (NM) • Updated Software Design and Implementation (Chapter 10) (NM) • Updated Glossary (NM) • Updated Bibliography (NM)
04/04/2025	Weekly Report 23, Usability Testing, Software Design and Implementation, Sequence Diagram, System Architecture: • Completed Weekly Report 23 (Chapter A) (NM) • Updated Usability Testing chapter (Chapter 12) (NM) • Created Software Design and Implementation chapter (Chapter 10) (NM) • Updated Sequence Diagram (Figure 8.8) (SG) • Updated System Architecture chapter (Chapter 8) (NM)
03/28/2025	Weekly Report 22, Usability Testing, Conclusion, Bibliography, Glossary: • Completed Weekly Report 22 (Chapter A) (NM) • Created Usability Testing chapter (Chapter 12) (NM) • Created Conclusion chapter (Chapter 13) (NM) • Updated Bibliography (NM) • Updated Glossary (NM)
03/14/2025	Weekly Report 21, Use Cases, Requirements, Classes: • Completed Weekly Report 21 (Chapter A) (NM) • Added new use case for making software updates (Table 6.6) (NM) • Updated requirements to include new use case that was added (Chapter 5) (NM) • Added a new activity diagram and description for use case 5 (Figure 8.7) (NM) • Updated class diagram (Figure 8.1) and CRC cards (Figure 8.2) (SC) • Updated Use Case Diagram (Figure 6.1) to include new use case that was added (SC)
03/07/2025	Weekly Report 20, Deployment Diagram: • Completed Weekly Report 20 (Chapter A) (NM) • Updated Deployment Diagram (Figure 8.10) (SG)
02/28/2025	Weekly Report 19, Hardware Device Implementation: • Completed Weekly Report 19 (Chapter A) (NM) • Completed Hardware Device Implementation chapter (Chapter 11) (NM)

Table 1: Document Update History

Date	Updates
02/21/2025	Weekly Report 18, System Architecture, User Interface Design, Preliminary Implementation, GitHub Repository, Hardware Device Implementation, Use Cases, Component Diagram, Requirements: • Completed Weekly Report 18 (Chapter A) (NM) • Updated Chapter 8 so the explanations of all updated diagrams are now accurate and reflect any project changes (NM) • Updated previous chapters of the report (Chapter 7, 8, 9, 14) (NM) • Added a new chapter for the hardware device implementation (Chapter 11) (NM) • Updated use case diagram explanation (Chapter 6) (NM) • Updated Component Diagram (Figure 8.9) (SC) • Updated User Requirements (Chapter 5) (NM)
02/14/2025	Weekly Report 17, Use Cases, System Architecture, Team Declaration, Development Plan, Requirements: • Completed Weekly Report 17 (Chapter A) (NM) • Ensured consistency between all Use Case names and diagrams (Chapter 6, 8) (SC, SG) • Updated class diagram and CRC cards (Chapter 8) (SC) • Updated details in previous chapters of report to reflect any changes in current project and scope (Chapter 3, 4, 5, 6) (NM)
02/07/2025	Weekly Report 16, Proposed Budget, System Architecture: • Completed Weekly Report 16 (Chapter A) (NM) • Updated Proposed Budget (Chapter 2) (NM) • Updated Activity Diagrams (Chapter 8) (SG) • Updated Sequence Diagram (Chapter 8) (SG)
01/31/2025	Weekly Report 15, Use Cases: • Completed Weekly Report 15 (Chapter A) (NM, SC) • Updated use case diagrams to conjoin all use cases into a single view (Figure 6.1) (SC) • Updated logical view to include new class diagram and CRC cards (Figures 8.1,8.2) (SC)
01/24/2025	Weekly Report 14, Class Diagram: • Completed Weekly Report 14 (Chapter A) (NM, SC) • Updated class diagram (Chapter 8) (SC)

Table 1: Document Update History

Date	Updates
12/19/2024	Use Cases: • Updated use case diagrams (Chapter 6) (SC)
12/13/2024	Weekly Report 13, Preliminary Implementation: • Completed Weekly Report 13 (Chapter A) (NM) • Continued refining Preliminary Implementation (Chapter 9) (NM, AS, SC, SG)
12/06/2024	Weekly Report 12, Class Diagram: • Completed Weekly Report 12 (Chapter A) (NM) • Updated class diagram (Chapter 8) (SC)
11/26/2024	Preliminary Implementation, Proposed Budget, GitHub Repository: • Added Preliminary Implementation chapter (Chapter 9) (NM) • Updated project budget and created a Proposed Budget chapter (Chapter 2) (NM) • Created GitHub Repository chapter (Chapter 14) (NM)
11/22/2024	Weekly Report 11: • Completed Weekly Report 11 (Chapter A) (NM)
11/19/2024	Glossary: • Added more terms throughout the document to the glossary (NM)
11/18/2024	User Interface Design: • Completed UI Prototyping (Chapter 7) (SG)
11/15/2024	Weekly Report 10, System Architecture: • Completed Weekly Report 10 (Chapter A) (NM) • Completed UML Diagrams (Chapter 8) (SG, SC) • Updated all UML diagram descriptions (Chapter 8) (NM, SC)
11/08/2024	Weekly Report 9, User Interface Design, System Architecture: • Completed Weekly Report 9 (Chapter A) (NM) • Developed User Interface Design (Chapter 7) (NM) • Added System Architecture chapter (Chapter 8) (NM)
10/31/2024	Weekly Report 8: • Completed Weekly Report 8 (Chapter A) (NM)

Table 1: Document Update History

Date	Updates
10/23/2024	Weekly Report 7, Requirements, Use Cases: • Completed Weekly Report 7 (Chapter A) (NM) • Added to Project Specification (Chapter 5) (NM, SG, SC) • Completed Use Cases Chapter (Chapter 6) (NM, SG, SC) • Completed Use Case Diagrams (Chapter 5) (AS)
10/17/2024	Weekly Report 6, Use Cases: • Completed Weekly Report 6 (Chapter A) (NM, SC) • Added Use Case Chapter (Chapter 6) (SG)
10/10/2024	Weekly Report 5, Glossary, Requirements: • Completed Weekly Report 5 (Chapter A) (NM) • Added a Glossary (NM, SC) • Added a Requirements Chapter (Chapter 5) (AS, NM, SC, SG)
10/1 - 10/3/2024	Weekly Report 4, Development Plan: • Completed Weekly Report 4 (Chapter A) (NM) • Updated Development Plan (Chapter 4) (AS, NM, SG, SC)
09/26/2024	Weekly Report 3, Development Plan: • Completed Weekly Report 3 (Chapter A) (NM) • Updated Development Plan (Chapter 4) (NM, SC)
09/20/2023	Weekly Report 2, Use Case Interview, Project Proposal: • Updated Project Proposal and Mission Statement (Chapter 3) (AS, NM, SG, SC) • Conducted interview with a potential user of the visually impaired community to gather research and determine potential use cases (NM, SG) • Completed Weekly Report 2 (Chapter A) (NM, SC)
09/16/2023	Team Declaration: • Added a chapter on Team Declaration (Chapter 3) in which team name, team members, and a brief project proposal is provided. Focuses on 3 components: mission statement, key drivers, and key constraints. (AS, NM, SG, SC) • Finalized team member roles and responsibilities (AS)

Table 1: Document Update History

Date	Updates
09/12/2024	Introduction, Weekly Reports: • Added chapters on Introduction (Chapter 1) and Weekly Reports (Chapter A). The Introduction provides a brief description of everyone on our team. The Weekly Reports will address what our team accomplished the past week, what we will address next week, a list of current action items, and any issues and risks. (NM)
09/10/2024	Finalize Project Idea, Create Overall Report Template: • Finalized project idea (AS, NM, SG, SC) • Updated Final Report Manual template for easier references of requirements, figures, and other labels (AS, NM, SG, SC)

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

– Neeti Mistry

The product we aim to develop, C-ALL (Cognitive Assistance with LiDAR Localization), is designed to enhance the navigation and independence of visually impaired individuals in urban environments. C-ALL is made up of a lightweight wearable glove hardware device and a mobile application, using LiDAR technology. The LiDAR sensor, integrated Apple's newer iPhone models, scans the user's surroundings. Using geographical location tracking through ARKit, C-ALL can assist users in identifying obstacles and guiding them to their destinations by providing real-time feedback.

The mobile app allows users to set destinations and receive haptic feedback, indicating the direction for the user to follow in order to avoid obstacles. This enables users to navigate with greater confidence and independence, without relying solely on traditional aids such as canes or guide dogs. Unlike expensive alternatives, C-ALL is priced affordably at 500 dollars, making it an accessible solution for a wider range of users. C-ALL aims to revolutionize mobility for visually impaired individuals in dynamic urban environments.

** [1] *How Do Blind and Visually Impaired People Get Around?*
The source "How Do Blind and Visually Impaired People Get Around?" from The Chicago Lighthouse provides valuable information about the tools and resources available to visually impaired individuals for mobility and navigation. It covers various assistive technologies such as orientation and mobility training, the use of guide dogs, white canes, and other innovations designed to help individuals navigate their environments. The resource also emphasizes the importance of accessibility in urban spaces and offers practical advice for improving mobility and independence. For the C-ALL project, this source helps by offering a broader understanding of the existing solutions for mobility challenges faced by visually impaired individuals. By referencing this information, we can contextualize our product within the landscape of mobility aids. It also helps us understand the limitations of current technologies (e.g., guide dogs, white canes), reinforcing the need for our innovative solution. **

1.1 About the Team

1.1.1 Ahmad Shah

I am a senior pursuing a Software Engineering major with a minor in Computer Science. My skillset involves creating and training Machine Learning models, specifically with visual data, tackling issues regarding detection and tracking. In the summer of 2024, I worked alongside Florida International University through the National Science Foundation leveraging machine learning and drone swarms for persons identification during disasters. I am ecstatic about working in a diverse team to develop an amazing product for our senior design project.

1.1.2 Neeti Mistry

My name is Neeti Mistry and I am a senior Software Engineering major with a minor in Computer Science. My curiosity drives me and I am always eager to learn. This past summer, I interned as a Cybersecurity Project Manager at Merck. I am excited to apply those skills to my senior design project and moving forward in my career. I love to travel and am really excited to move and explore new places after graduation.

1.1.3 Sara Gaber

My name is Sara Gaber I am a Software Engineering major with a minor in Cybersecurity. I gained some work experience last summer as an intern for Fiserv working in data processing. In my free time I enjoy listening to music and making art, specifically painting and drawing. I hope to eventually move further with Fiserv or find a place at a startup.

1.1.4 Sohan Chatterjee

I am a fourth year student at Stevens pursuing a B.E. in Software Engineering. I have abilities to work front-end, back-end, and full stack in various instances from web development to robotics to data analysis. Through programming courses and my most recent internship, I have developed a diversified skill set in software development and look forward to applying my knowledge in a professional environment. I have a keen interest in understanding the relationship between technology and other disciplines and want to continue exploring the potential of software.

Chapter 2

Proposed Budget

– Neeti Mistry

2.1 Introduction

This chapter outlines the estimated budget required for the development of the Cognitive Assistance with LiDAR Localization (C-ALL) project. The budget includes both hardware and software components necessary to build the prototype and perform initial testing and development. The items listed here reflect costs for Phase 1 of the project.

2.2 Budget Breakdown

This section will list the specific items needed and their corresponding costs. It is broken down into categories such as hardware, software, and materials, as shown below:

Hardware:

- iPhone 12 Pro (for testing): $350
- Battery: $21
- Triple Axis Compass Magnetometer Sensor Module 3.3V 5V for Arduino and Raspberry Pi: $12
- Raspberry pi 4: $120
- Limit Switches: $6
- Breadboard Kit: $9
- Wires: $14
- Prototype Materials (Glove): $10
- Servo Motors: $21

2.3 Total Estimated Budget

This section will summarize the total cost estimate for the initial phases of the project.

Total Estimated Budget: $532.53

The following link presents a detailed breakdown of the budget items and their costs in a table. This includes hardware materials, with relevant links, that are necessary for the first phase of development. The spreadsheet is also linked for further details.

Spreadsheet Link

Chapter 3

Team Declaration

– Ahmad Shah, Neeti Mistry, Sara Gaber, and Sohan Chatterjee

3.1 Team Name and Team Members

Team Name: Cognitive Assistance with LiDAR Localization (C-ALL)

Team Members:

- Ahmad Shah
- Neeti Mistry
- Sara Gaber
- Sohan Chatterjee

3.2 Project Proposal

The Cognitive Assistance with LiDAR Localization (C-ALL) project aims to create an assistive navigation system for visually impaired individuals, leveraging LiDAR technology, a wearable haptic feedback device (hardware), and a mobile application. The system is designed to provide real-time obstacle detection, route guidance, and enhanced mobility, enabling users to navigate independently and confidently in various environments. This project addresses the lack of affordable and effective assistive technologies, offering an innovative solution that enhances accessibility and independence. The final deliverable is a fully functional prototype of the C-ALL system, including a mobile application, LiDAR integration, and a wearable feedback device.

3.3 Mission Statement, Key Drivers, and Key Constraints

Mission Statement:

The mission of the C-ALL (Cognitive Assistance with LiDAR Localization) project is to develop an innovative assistive technology solution that empowers visually impaired individuals to

navigate their surroundings with greater independence and confidence. By integrating advanced LiDAR technology, haptic feedback, and mobile applications, we aim to create an accessible, reliable, and user-friendly product that provides real-time spatial awareness. Our goal is to address a critical business need in the assistive technology market, enhancing mobility solutions and improving the quality of life for individuals with visual impairments. We intend to deliver a scalable product that can be further developed for broader commercial use, ensuring both technical innovation and societal impact.

Key Drivers:

The development of C-ALL (Cognitive Assistance with LiDAR Localization) is driven by the need to provide enhanced mobility solutions for visually impaired individuals. Current assistive technologies often rely on limited sensory input or cumbersome devices, leaving significant room for improvement in terms of user experience, accuracy, and real-time feedback. The increasing affordability and accessibility of LiDAR technology, combined with advancements in mobile applications and audio feedback, present a unique opportunity to offer a more intuitive and effective solution.

Our motivation stems from a desire to bridge this technological gap and address a pressing societal need. By leveraging different technologies, we aim to empower visually impaired individuals to navigate their environments with greater confidence, independence, and safety. The potential to impact lives in meaningful ways, combined with the opportunity to innovate in the assistive technology space, drives our commitment to this project.

Key Constraints:

The development of C-ALL (Cognitive Assistance with LiDAR Localization) faces several constraints that must be carefully managed to ensure the project's success. These constraints include technical and user-specific limitations, all of which shape the project's scope and feasibility.

Technical Limitations: One of the primary constraints is the computational complexity of processing LiDAR data in real-time. LiDAR sensors generate large volumes of point cloud data that require significant processing power to convert into a simplified 3D representation. In our current set-up, we plan to offload data processing to a central server during testing, but in the final product, the system will need to rely on more compact, affordable hardware. Balancing processing speed and accuracy within the constraints of limited hardware resources (such as mobile processors or compact GPUs) will be a significant challenge.

Additionally, integrating multiple components—LiDAR sensors, hardware device, and mobile applications—requires seamless communication. Latency and synchronization issues between these components could lead to inaccuracies in real-time navigation. We must ensure that the system maintains low latency while processing data and communicating feedback, especially in complex environments.

User Experience and Accessibility: Designing an intuitive and user-friendly interface for visually impaired users is another constraint. The product must accommodate the diverse needs of users with varying degrees of visual impairment, and any misalignment in the interface or haptic feedback system could lead to confusion or errors. The mobile application, for instance, must comply with accessibility standards and provide seamless interaction through voice commands or other assistive technologies.

Time Constraints: Time management is a significant factor in this project. The development timeline for our senior design project is limited, and achieving all technical milestones within this time frame requires careful planning. Prototyping, testing, and iterating the product to meet the necessary performance and reliability standards will need to be done efficiently. Any delays in one area (e.g., hardware procurement, software development, or testing) could push back our progress and affect the overall project delivery.

Chapter 4

Development Plan

– Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee

4.1 Introduction

The C-ALL (Cognitive Assistance with LiDAR Localization) project aims to develop an innovative assistive technology solution that empowers visually impaired individuals to navigate their surroundings with greater independence and confidence. The product consists of a wearable device mounted on a glove, a mobile application developed for iPhone, and a back-end computation system. By integrating advanced LiDAR technology, haptic feedback, and mobile applications, we aim to create an accessible, reliable, and user-friendly product that provides real-time spatial awareness.

We are addressing the lack of effective navigation tools that provide real-time obstacle avoidance and directional guidance for visually impaired individuals. Our success criteria include developing a functional prototype that allows users to navigate a straight path while avoiding obstacles, providing directional feedback through the glove-mounted device, and enabling users to input desired destinations for navigation. Key supporting documents include the Requirements Document, Architecture Review Document, and User Interface Design Specifications.

4.2 Roles and Responsibilities

1. **Development Lead** - Ahmad Shah
    - **Responsibility:** Oversee development and lifecycle
2. **Buildmeister** - Sohan Chatterjee
    - **Description:** Make sure programs and tools used by Developers are able to effectively transfer data and communicate without having software issues.
      Ex) Make sure everyone is using the same versions of coding languages, document necessary packages used and required for each program to run in program documentation as an “install executable.”
      (You can make a script automatically download required pip installs so team members

don't need to worry about having the right versions for each software).
Also handle battery and power distribution to systems created by Ahmad Shah (Developer) and Sara Gaber (Developer).

3. **Architect** - Sara Gaber

   - **Responsibility:** Design overall system structure
   - **Description:** Will ensure that the architecture of the system and tools used fits the project's needs. We want to make our project solution cost effective. When other Developers use packages to make their respective projects, the Architect will find alternatives that would be more efficient.

4. **Developers** - Ahmad Shah, Sohan Chatterjee, Sara Gaber

   - **Ahmad Shah:** Utilize LiDAR sensor
   - **Sohan Chatterjee:** Acquire access to LiDAR sensor from iOS and research appropriate view field of LiDAR sensor.
     (What would be optimal? Do we need to see items on the floor? How much to the floor should it see? How high should it see? How are you going to tackle user height?)
   - **Sara Gaber:** Create the interactive glove interface. The method of translating the information to the user will be up to Developer's discretion.

5. **Test Lead** - Sohan Chatterjee

   - **Responsibility:** Lead Testing Team

6. **Testers** - Neeti Mistry, Sohan Chatterjee

   - **Responsibility:** This team will focus on testing the applications developed by the Developer team. Testers will also find edge cases and apply stress testing, accounting for best and worst case scenarios. In addition, they will analyze the code to make it optimal and keep it as efficient as possible. Testers will coordinate with the Modification Request Board, appropriately.

7. **Documentation** - Neeti Mistry

   - **Responsibility:** This role is essential for ensuring that all aspects of the project are accurately recorded and communicated. This includes creating and maintaining comprehensive documentation for project requirements, design specifications, user feedback, and development processes. The team will be responsible for drafting diagrams, guides, and progress reports, facilitating clear communication among team members and stakeholders. They will also ensure that all documentation adheres to industry standards and is easily accessible to all project participants, thereby supporting project transparency and continuity.

8. **Documentation Editor** - Neeti Mistry

- **Responsibility:** Add substance, detail, and demonstrative diagrams to enhance project report quality.

9. **Designer** - Sara Gaber

   - **Responsibility:** The Designer will design the structure and design of the final product, focusing not only on functionality, but style and option variability. The Designer will also create a front-end website showcasing the product.

10. **User Advocate** - Neeti Mistry

    - **Responsibility:** Represent the end user and their desires and needs, find flaws with the product, and discover improvements.

11. **Risk Management** - Neeti Mistry

    - **Responsibility:** Identify risks with the project and risks with marketability and current competitors. Find how to make the product more appealing or unique.

12. **System Administrator** - Ahmad Shah

    - **Responsibility:** The System Administrator plays a crucial role in managing and maintaining the project's technical infrastructure, ensuring optimal performance, security, and reliability. Key responsibilities include configuring and managing servers and software applications, monitoring system performance, and troubleshooting issues to minimize downtime. They are also tasked with user access management, implementing backup and recovery processes, and providing technical support to team members. Additionally, the System Administrator collaborates closely with the development team to ensure system requirements align with project goals, while maintaining accurate documentation and implementing security measures.

13. **Modification Request Board** - Ahmad Shah, Sara Gaber

    - **Responsibility:** Oversee requests from all roles and will document requests and implications, as well as how it would affect project development from other roles. In decisions that may be sensitive or would affect the project as a whole, communicate with the Modification Request Board.

14. **Requirements Resource** - Neeti Mistry

    - **Responsibility:** Responsible for gathering, analyzing, and documenting the project's functional and non-functional requirements. This role involves collaborating with stakeholders, including users from the visually impaired community, to ensure that all needs are accurately captured and understood. The Requirements Resource will create detailed specifications that guide the development process, facilitating effective communication between team members and ensuring that the final product aligns with user expectations. Additionally, they will prioritize requirements based on project goals and constraints, helping to maintain a clear focus throughout the development lifecycle.

15. **Customer Representative** - Neeti Mistry, Sohan Chatterjee

    - **Responsibility:** The Customer Representative serves as the primary liaison between the project team and the visually impaired community, ensuring that user needs and expectations are effectively communicated throughout the development process. This role involves gathering feedback from users, advocating for their requirements, and providing insights on usability and accessibility. The Customer Representative will facilitate user testing sessions and focus groups to validate features and gather constructive feedback, helping the team make informed decisions that enhance the user experience. By maintaining open lines of communication with both users and the project team, they play a vital role in aligning the project's outcomes with the needs of the target audience.

16. **Customer Responsible for Acceptance Testing** - Neeti Mistry, Sohan Chatterjee

    - **Responsibility:** The Customer Responsible for Acceptance Testing plays a critical role in ensuring that the final product meets the expectations and requirements of the visually impaired community. They are tasked with developing and executing acceptance test plans based on user needs and project specifications. They will collaborate closely with the project team to define success criteria and identify key performance indicators for the product. During the testing phase, they will facilitate user testing sessions, gather feedback on functionality and usability, and document any issues or discrepancies. By validating that the product aligns with user requirements, the Customer Responsible for Acceptance Testing ensures that the final deliverable is not only functional but also user-friendly and accessible.

4.3 Method

The methodology for this project integrates both software and hardware development processes to create a robust and reliable solution. The following sections outline the software, hardware, backup plans, review processes, and build strategies used throughout the development.

4.3.1 Software

1. **Programming Languages**:

    - Swift (version 5.5) for iOS application development
    - Python (version 3.9) for back-end computations
    - Raspberry pi for microcontroller programming

2. **Operating Systems**:

    - iOS 15 for the mobile application
    - macOS Sequoia for development environment

3. **Software Packages/Libraries**:
    - GeoARKit framework for feature points
    - CoreLocation framework for compass data
    - CoreBluetooth framework for Bluetooth communication
4. **Code Conventions**:
    - Swift code will follow the Swift API Design Guidelines

4.3.2 Hardware

1. **Development Hardware**:
    - iPhone 12 Pro or newer with LiDAR capabilities
    - MacBook Pro for iOS development
    - Raspberry Pi
    - Battery
    - Servo motors for pointer control
    - Sensor Module for Raspberry pi
    - Limit switches
    - Breadboard kit
    - Wires
    - Prototype Materials (Glove)
2. **Test Hardware**:
    - Prototype glove with mounted device
    - Use TestFlight for testing
3. **Target/Deployment Hardware**:
    - Final glove-mounted device with optimized design
    - iPhone with the deployed application

4.3.3 Backup Plan

In case of challenges with data transfer speed and processing capabilities, we have devised two backup plans:

- **Cloud-Based Computations**: Utilize AWS cloud services to handle back-end computations, enabling faster data transfer and scalability.

- **On-Device Computations**: Perform all computations on the iPhone, leveraging its optimized architecture for handling 3D depth maps.

Individual team members will maintain local backups, and we will use GitHub for version control and collaboration.

4.3.4 Review Process

The review process is integral to ensuring the quality and effectiveness of our project deliverables. We will conduct several types of reviews throughout the project lifecycle, including architecture, usability, design, security, privacy, and code reviews. Each type of review will target specific aspects of the project to guarantee that all components align with our project goals and user needs.

1. Types of Reviews
   (a) Architecture Reviews: Assess the overall system architecture to ensure scalability and alignment with project objectives
   (b) Usability Reviews: Evaluate the user interface and experience to confirm that it meets the needs of the visually impaired community
   (c) Design Reviews: Examine design elements to ensure consistency and accessibility across the application
   (d) Security and Privacy Reviews: Analyze the system for vulnerabilities and compliance with data protection regulations to safeguard user information
   (e) Code Reviews: Conduct thorough evaluations of the codebase to identify issues, improve code quality, and ensure adherence to coding standards
2. Review Approach: We will adopt a formal review process for architecture, usability, design, security, and privacy assessments, utilizing established corporate standards where applicable. Code reviews will follow a combination of formal and informal approaches, incorporating peer reviews to foster collaboration and knowledge sharing among team members.
3. Responsibilities: The project manager will oversee the review process, ensuring that all reviews are scheduled and conducted as planned. Each team member will be responsible for their respective areas during the reviews, with designated leads for each type of review. Any issues uncovered during the reviews will be documented and assigned to appropriate team members for resolution, with follow-up reviews scheduled as necessary to confirm that issues have been addressed.
4. Code Readings: Regular code readings will be incorporated into the review process to promote transparency and collective understanding of the codebase among team members. These sessions will allow team members to discuss code implementation, share best practices, and identify potential improvements in a collaborative setting.

By implementing a comprehensive review process, we aim to enhance the quality of our project outcomes and ensure that we deliver a product that meets the highest standards of usability, security, and performance.

4.3.5 Build Plan

1. Revision Control System and Repository Used:
   We will use GitHub for version control and Jira for project tracking and issue management. All code changes will be committed to GitHub repositories, and Jira will be used to track progress, manage sprints, and handle any reported issues or bugs.

2. Continuous Integration:
   We will implement GitHub Actions for continuous integration (CI). Every time a pull request is made, the system will automatically run unit tests, build the project, and check for code quality issues. This ensures that every commit maintains the integrity of the build and passes automated tests before merging into the main branch.

3. Regularity of the Builds:
   Builds will occur **daily**, triggered automatically through the continuous integration system (CI) or manually if major changes have been committed.

4. Deadlines for the Builds (Deadline for Source Updates):
   First Iteration Deadline: December 13, 2024
   Completed Project Deadline: End of April 2025

5. Multiplicity of Builds:
   We will support multiple builds across different environments.
   Development builds: For testing ongoing changes in a local environment
   Staging builds: For pre-production testing, ensuring features are stable before moving to production
   Production builds: For deployment to the final product environment, ensuring the application is in a stable state

6. Regression Test Process (See Test Plan):
   The regression test process will be integrated into the continuous integration system, where a suite of automated tests will be executed after every build. These tests will ensure that new changes do not introduce any bugs or break existing functionality. If any regression issues are found, they will be reported and tracked in Jira, and the team will resolve them before the changes are merged into the main branch.

4.3.6 Modification Request Process

1. MR Tool: GitHub Pull Request Tool

2. Decision Process:
   The decision process will involve a Modification Request Board consisting of Ahmad Shah and Sara Gaber. When a team member submits a pull request, the board will review the changes, assess their impact on the overall project, and determine whether the modification aligns with project goals and architecture. The board will consider factors such as:
   - Compatibility with the current system architecture

- Potential risks introduced by the modification
- Code quality, maintainability, and scalability
- Alignment with user requirements and project constraints

3. State whether there will be two process streams one during development and one after development:
   Yes, there will be two process streams.
   During Development: The modification requests will follow the regular GitHub pull request review process, with the board conducting frequent checks to ensure development timelines are maintained. Urgent modifications will be prioritized to avoid blocking other tasks.
   After Development (Post-Launch): A separate process stream will be established for post-development modifications. In this phase, the Modification Request Board will convene less frequently (biweekly) to handle any bug fixes, feature enhancements, or patches. Priority will be given to critical bug fixes and security updates. Each modification will also undergo thorough regression testing to ensure that no unintended side effects occur in the stable release.

4.4 Virtual and Real Workspace

Our team utilizes both a physical and a virtual workspace to effectively collaborate on our Senior Design Project. The real workspace is the Software Engineering Lab, where our section meets twice a week on Tuesdays and Thursdays. This in-person collaboration allows us to discuss the project face-to-face, troubleshoot issues, and review our progress in real time. For our virtual workspace, we leverage multiple tools:

- Discord for virtual meetings and quick communication. It allows us to stay connected outside of class hours and hold discussions remotely.
- GitHub to manage and track our code and documents. Using GitHub, we maintain version control, ensure that everyone is working with the latest code, and review each other's contributions.
- We also use Jira for effective project management and to track our project's tasks and progress, assigning responsibilities, setting deadlines, and managing workflows. It helps us stay organized with clear visibility of project milestones, team contributions, and any blockers, ensuring effective collaboration and timely completion of deliverables.
- Google Drive to share presentations and other documents. While it doesn't offer the highest level of privacy, it's sufficient for our needs as a university project, enabling us to collaborate on slides and written documents efficiently.
- Overleaf and LaTeX for preparing our professional project reports and documentation. Overleaf allows us to collaborate on LaTeX-based documents, ensuring professional formatting and easy version tracking for all our reports and technical documents.

This combination of real and virtual workspaces ensures that we can collaborate effectively, whether we are meeting in person or working remotely.

4.5 Communication Plan

4.5.1 Heartbeat Meetings

The "heartbeat" meetings are a crucial component of our project management strategy, designed to regularly assess the project's status and foster team collaboration. These meetings are typically held on a weekly basis, ideally scheduled for the early part of the day to ensure that team members can engage without the distractions of their daily responsibilities.

Each meeting follows a structured agenda that includes brief updates from each team member, followed by a review of open issues and potential risks facing the project. To maintain efficiency and keep discussions on track, meetings are designed to be concise, lasting no longer than thirty minutes. After each session, notes summarizing key discussions, action items, and decisions are distributed to all relevant team members. Additionally, issues raised during the meetings are systematically tracked and revisited in subsequent sessions to ensure accountability and drive continuous improvement.

4.5.2 Status Meetings

Status meetings are strategically designed to provide the entire team with a clear overview of the project's progress and current status. Unlike the more frequent heartbeat meetings, status meetings are held less often, ideally on a biweekly basis, depending on the project's current needs. The primary focus of these meetings is to deliver concise updates on milestones achieved, resource utilization, and overall project progress.

These meetings are structured to be short and to the point, ensuring that everyone receives the necessary information without excessive detail. Each session typically includes brief overviews from each project lead who summarize their areas of responsibility, highlights achievements, and outlines any potential challenges. If any issues arise during the updates that require deeper discussion or problem-solving, these will be addressed in separate meetings specifically dedicated to issue resolution, ensuring that status meetings remain focused on high-level project insights.

By maintaining this format and periodicity, status meetings serve as an effective communication tool, keeping our team informed while allowing us to focus on our day-to-day operations and immediate challenges.

4.5.3 Issues Meetings

Issues meetings are important for addressing problems that arise during the project lifecycle, ensuring that the team is kept informed and involved in finding solutions. It is essential to avoid surprising the rest of the team with unexpected challenges; therefore, if a problem is identified, a meeting should be scheduled at the earliest convenience of the manager to discuss the issue in detail.

Alerts regarding potential issues will typically arise during the heartbeat meetings, where team members can present updates and highlight any challenges encountered. If a concern is deemed significant enough to warrant immediate attention, the team will collectively determine the need for an issues meeting. This decision will be based on the severity of the problem, its impact on project timelines or deliverables, and the urgency of the required resolution.

Once an alert is triggered, the issues meeting will be convened with relevant stakeholders, including key team members, to ensure a collaborative approach to problem-solving. During this meeting, the team will present the context of the issue, discuss possible solutions, and develop a clear action plan to mitigate the problem. By establishing this structure, we ensure that issues are addressed promptly and effectively, maintaining transparency and fostering a proactive project management environment.

4.6 Timeline and Milestones

September 3, 2024 to October 3, 2024: Determine project scope, potential solution, clients, and business objective. Define a problem and a plan to resolve it. Prepare a project proposal including mission statement, key drivers, and key constraints.

October 3, 2024 to October 31, 2024: Begin initial design of hardware and determine what needs to be bought/created. Weigh out options, alternate solutions, and devise multiple iterations to understand how to proceed according to the project mission. Select a hardware design to continue with and begin planning to build for initial prototype.

October 10, 2024 to October 31, 2024: Begin initial design of mobile application that uses LiDAR technology from iPhone. Explore how data is gathered and relayed, and how this can be used to accomplish the scope of the project. Determine features of the mobile application and what will be essential to show for best user experience.

October 31, 2024 to November 26, 2024: Finalize first prototypes of both hardware and software, separately. Implement necessary features to determine if solution is viable and should be continued on. Build a working piece of hardware and a full mobile application that meet significant requirements (version 1.0.0).

November 19, 2024 to December 13, 2024: Connect hardware to software. Use LiDAR from mobile application to control and direct hardware feedback. Establish communication between the two parts.

December 13, 2024 to January 21, 2025: Test initial prototype and gather data on results. Test all use cases, edge cases, and fail cases. Continue improving hardware and software according to test results. Begin designing second prototype and next big update for software.
Critical Participants: potential clients.

January 21, 2025 to February 4, 2025: Build second prototype and version 2.0.0 of software. Resolve any major issues found during testing period. Speed up communication between hardware and software.

February 4, 2025 to February 18, 2025: Test second prototype and version of software, gather data, and draw conclusions. Determine next steps for both hardware and software and plan to implement any missing key features.
Critical participants: potential clients

February 18, 2025 to March 4, 2025: Build final prototype and continue to improve software. Address all bugs and issues within hardware, software, and firmware. Refine product to be efficient and easy to use.

March 4, 2025 to April 25, 2025: Further refine final prototype and version of software. Finalize all documentation related to the project. Continue testing with potential client(s). Prepare for Innovation Expo and final submissions.
Critical participants: potential clients

4.7 Testing Policy/Plan

We will be conducting various tests in each phase of our development and employing a variety of testing methodologies. The three listed below will be our focus.

1. Unit Testing: We will conduct testing of our individual use cases. The use cases to be tested will represent various requirements outlined. We will then generate a set of unit tests for each use case to ensure that our requirements are met for each phase of the development.

2. Integration Testing: Due to the reliance of the communication between software and hardware components, we will be conducting constant integration testing. Each phase of our development will test how our different components interact.

3. Acceptance Testing: The product is meant to help improve the lives of visually impaired people and thus we must conduct acceptance testing with our prototype. We plan to show our prototype at several stages to a group of visually impaired individuals, and allow them to use it for themselves when applicable, to generate constant feedback on the acceptance of the product by the client base.

Our integration testing will begin as soon as we are working with multiple components of our project and will be vital. Unit testing will begin immediately and occur at the end of each phase of our build plan.

4.8 Risks

There is some risk to the project due to the development of new hardware. We do run some risks in the communication of the navigation hardware with our logic. To mitigate this we are using the LiDAR included within newer iPhone models since there are existing APIs and frameworks for it. The hardware we are developing for translating direction to sensory output is a large area of the risk since developing this hardware comes with its own firmware issues. Therefore we will start developing this first so if there are major problems with this idea they are found early on.

4.9 Assumptions

We are assuming that we will be only relying on our group members for all work. We also are assuming that the budget for our hardware components will be covered by Stevens Institute of

Technology. We are assuming that we will be able to work on this project from September through April. We assume that during December to January there will be a slow in productivity due to time off.

4.10 Distribution List

- Professor David Darian Muresan
- Stakeholders
  - Potential visually impaired clients
  - Hospitals and medical specialists
  - Insurance companies
  - Caregivers and family members
  - Nonprofit organizations
  - Device manufacturers
- Potential Investors
  - Government agencies
  - Lawmakers
  - Regulatory bodies
  - Medical technology companies

4.11 IRB Protocol

Our project does not require an IRB application. Any interviews or testing involving human subjects will be conducted in an educational setting, following protocol, and will not put subjects at risk of privacy issues, and their consent will be required to continue with their involvement. Participants will be given full transparency about the project and documentation.

U.S. Department of Health and Human Services (HHS) exemptions 45 CFR 46.104(d)(1), 45 CFR 46.104(d)(2)(ii), and 45 CFR 46.104(d)(3) apply.

4.12 Worry Beads

The current worry is being able to complete the entire project by April. This is a large task and we have been warned by two professors that it may be too big of a task considering the need for hardware and our lack of experience in that regard. We plan to have a functional piece of hardware for our first prototype in 4-6 weeks from now (November 1, 2024) to prove to ourselves that we can accomplish this difficult task and keep us on track of our timeline.

Another worry is the communication between our mobile application and our hardware. We do not doubt our programming abilities, but this project requires server communication and cloud integration which is new to us. We worry about the efficiency of our product and hope that there is little to no delay in the communication between the LiDAR sensor and hardware.

4.13 Documentation Plan

Our documentation plan begins with our Weekly Reports (Chapter A). Each week, we document everything related to our project from internal meetings to initial plans to implementation. These weekly reports contain the most up-to-date status of our project including designs, progress, and future plans. Beyond the weekly reports, we are to document any interviews with clients regarding our product. We will also document user stories and use cases as they come up, dependent on our design and future prototypes. As we continue with this project, we plan to document all of our steps to clearly outline our thought and build processes, including failures and reasoning for change of directions that may occur. We plan to document our various test cases and future improvements based on the results to visualize the timeline of our project.

4.14 Other

4.14.1 Stakeholder Engagement

Active engagement with stakeholders, including the visually impaired community, is crucial for gathering valuable feedback and ensuring that the final product meets their needs. Regular communication with stakeholders helps to build trust and keeps them informed about project developments. We will be in regular communication with our stakeholders as part of our acceptance testing and requirement and use case generation.

Chapter 5

Requirements

– Neeti Mistry, Sohan Chatterjee, Sara Gaber

5.1 Introduction

Our project focuses on addressing the challenges faced by visually impaired individuals when navigating public spaces. Despite advancements in assistive technologies, there is still a significant gap in providing efficient and accessible solutions for real-time navigation. Many existing tools lack precision, accessibility, or adaptability to different environments, leaving users dependent on external assistance or limited in their independence.

To solve this problem, we are developing an innovative navigation system that provides real-time guidance to visually impaired users. Imagine a visually impaired user who frequently commutes in busy urban environments. With our system, the user will be able to navigate through complex spaces such as train stations, shopping malls, or universities with ease. The system will offer haptic feedback and route suggestions to help the user avoid obstacles, follow safe paths, and reach their destination independently.

This chapter will outline the requirements of the system, ensuring that the needs of the visually impaired are met through a functional, reliable, and accessible solution that significantly enhances mobility.

5.2 Stakeholders

Stakeholders are individuals or groups that have an interest in the development and outcomes of the project. They are essential for gathering requirements, providing insights, and ensuring that the system meets the needs of those it impacts. The following are key stakeholders involved in our project:

- Visually Impaired Clients: The primary users of our product, their feedback and experiences are vital for shaping the design and functionality of our solution. Engaging with this community will help us understand their specific needs, preferences, and challenges, ensuring that our project effectively addresses their requirements.

- Healthcare Professionals: This group includes medical specialists, therapists, and caregivers who work directly with visually impaired individuals. Their insights will help us understand how our product can integrate into existing care frameworks and support the rehabilitation process for users.
- Nonprofit Organizations: Organizations that advocate for the visually impaired can provide valuable resources, information, and support in reaching our target user base. Collaborating with these groups will enhance our understanding of the community's needs and ensure that our product aligns with broader goals in accessibility and inclusivity.
- Insurance Companies: As potential stakeholders in the financial support and accessibility of our product, insurance companies can offer insights into the reimbursement process for assistive technologies. Understanding their policies can help us position our product effectively in the market.
- Device Manufacturers: Partnerships with manufacturers of assistive technology can facilitate access to resources and expertise necessary for hardware development. Their involvement can also ensure that our solution is compatible with existing devices and technologies.

5.2.1 Customers

The primary customers of this project are visually impaired individuals who seek to enhance their independence and improve their quality of life through technology. This demographic includes people of various ages and backgrounds who may rely on assistive devices to navigate their environment, access information, and engage with the world around them.

These users utilize our system to address specific challenges they face in daily life, such as:

- Navigation: Visually impaired users often struggle with spatial awareness and orientation. Our system provides real-time assistance and guidance to help them navigate safely and efficiently in both familiar and unfamiliar settings.
- Accessibility: The product aims to enhance the accessibility of information, which can empower them to participate more fully in educational, professional, and social environments.
- Independence: By offering solutions that promote self-sufficiency, users can perform tasks that might otherwise require assistance, thereby increasing their confidence and autonomy.
- Social Interaction: The system can also facilitate communication and social interactions, helping visually impaired individuals connect with peers and participate in community activities.

Overall, our customers are visually impaired individuals seeking innovative solutions to improve their daily experiences, fostering a more inclusive and supportive environment.

5.2.2 Sponsors

Sponsors are individuals or organizations that provide financial support, resources, or strategic guidance to the project. In the context of our product designed to assist visually impaired individuals, our primary sponsor is Stevens Institute of Technology.

As our main sponsor, Stevens provides essential funding and a designated budget for acquiring the necessary hardware and resources for the project. This support enables our team to access the technology and materials needed to develop a functional and effective solution for visually impaired users. The university also fosters a collaborative environment, offering mentorship and expertise from faculty members, which is crucial in guiding the project towards its objectives.

5.2.3 Engineering and Technical Persons

Engineering and technical personnel play a crucial role in the design, development, and implementation of the system aimed at assisting visually impaired individuals. Their expertise ensures that the product is not only functional but also aligns with user needs and industry standards. Key stakeholders in this category include:

- Software Engineers: Responsible for developing the software components of the system, software engineers will design user-friendly interfaces and ensure the application operates smoothly across various devices. Their skills in programming languages, frameworks, and best practices in accessibility will be vital in creating a solution that is intuitive for visually impaired users.
- Hardware Engineers: Focused on the physical aspects of the project, hardware engineers will design and test the hardware components necessary for the system. This includes selecting sensors, hardware materials, and other assistive technologies that enhance the user experience. Their expertise will ensure that the hardware is robust, reliable, and optimized for the intended environment.
- User Experience (UX) Designers: UX designers will work closely with the target audience to gather feedback and insights. They will create prototypes and conduct usability testing to ensure that the system is accessible and meets the specific needs of visually impaired users. Their focus on user-centered design will be crucial in enhancing the overall functionality and satisfaction of the end-users.
- Project Managers: Overseeing the entire project, project managers will coordinate between different engineering and technical teams, ensuring that the project stays on schedule and within budget. They will facilitate communication, manage risks, and ensure that all technical aspects align with the project's goals.

By collaborating effectively, our team can ensure that the project is not only innovative but also practical and responsive to the needs of visually impaired individuals.

5.2.4 Regulators

Regulators are essential stakeholders that ensure the project complies with relevant laws, standards, and guidelines. Their role is critical in maintaining safety, accessibility, and ethical considerations in the development of products for visually impaired individuals. Key regulatory considerations for this project include:

- Accessibility Standards: Compliance with national and international accessibility standards, such as the Americans with Disabilities Act (ADA) and the Web Content Accessibility Guidelines (WCAG), is important. These standards set the benchmark for designing products that are usable by individuals with disabilities, ensuring that the system meets their specific needs.
- Safety Regulations: Regulatory bodies may establish safety standards that must be adhered to in the design and implementation of hardware components. For instance, ensuring that devices used by visually impaired individuals do not pose any physical hazards is crucial. Compliance with safety regulations from organizations like the Occupational Safety and Health Administration (OSHA) may be necessary.
- Assistive Technology Guidelines: Various regulatory bodies may have specific guidelines for assistive technologies aimed at individuals with disabilities. Adhering to these guidelines ensures that the product is recognized as a legitimate assistive device, potentially opening avenues for funding and support from government or non-profit organizations focused on disability assistance.

By engaging with these regulators throughout the project lifecycle, the development team will ensure that the system not only meets legal requirements but also serves the best interests of visually impaired users, promoting trust and reliability in the final product.

5.2.5 Third Parties

Third parties are external entities that may influence or be influenced by the development and deployment of the project. They play significant roles in the ecosystem surrounding assistive technologies for visually impaired individuals. Key third-party stakeholders include:

- Hardware Suppliers: Companies that provide the components and devices necessary for building our system, such as the LiDAR sensor in our case. These suppliers are crucial for ensuring that the project has access to high-quality sensors and other hardware essential for effective functionality.
- Software Vendors: Providers of software solutions that may be integrated into the system, including operating systems, accessibility tools, and application frameworks. Collaborating with software vendors can enhance the user experience and ensure that the system is compatible with existing technologies.
- User Groups and Communities: Engaging with groups representing visually impaired individuals can provide firsthand insights into their needs and preferences. Their feedback can

be invaluable in shaping the product's features, usability, and overall design, ensuring it truly addresses the challenges faced by the target audience.

By actively involving these third parties throughout the project, the team can leverage their expertise, resources, and networks, ultimately enhancing the quality and impact of the system developed for visually impaired individuals.

5.2.6 Competitors

Competitors are systems and solutions that offer similar functionalities aimed at assisting visually impaired individuals. Understanding these competitors is essential for identifying market gaps and differentiating our product. Key competitors include:

- Smart Canes: Devices like the WeWALK smart cane utilize sensors and GPS technology to assist visually impaired users in navigation. These canes provide audio feedback and can connect to smartphones for enhanced functionality. While they effectively aid in mobility, our project aims to offer a more comprehensive solution that combines navigation with contextual information.
- Mobile Applications: Various smartphone applications, such as Seeing AI and Be My Eyes, offer features that assist visually impaired users by providing descriptions of their surroundings or connecting them with sighted volunteers for real-time assistance. While these applications are valuable tools, our product intends to provide a more integrated hardware-software solution that does not rely solely on mobile devices.
- Wearable Technology: Devices like Aira and OrCam offer wearable solutions that provide visual assistance through smart glasses or similar technologies. These systems often rely on remote agents or advanced image recognition technology. Our project differentiates itself by focusing on a user-friendly and standalone system that empowers users without ongoing reliance on external services.
- Home Assistance Systems: Smart home systems designed for accessibility, such as Amazon Alexa or Google Assistant, can assist visually impaired users in navigating their homes and performing daily tasks. However, these systems are primarily voice-controlled and may not cater specifically to outdoor navigation needs, an area where our product aims to excel.

By analyzing these competitors, we can identify unique features and advantages of our system, such as affordability, ease of use, and comprehensive support for outdoor navigation. This understanding will help refine our product development strategy and enhance its value proposition for visually impaired individuals.

5.3 Key Concepts

This section defines the key concepts and terminology that are essential to understanding the project. These terms will be used consistently throughout this document and in the project to ensure clarity and coherence. For further details, please refer to the Glossary chapter.

1. Visually Impaired: A term used to describe individuals who have partial or complete loss of vision, impacting their ability to navigate and interact with their environment. This project aims to enhance the mobility and independence of visually impaired individuals.

2. Assistive Technology: Devices or systems designed to support individuals with disabilities in performing tasks that might otherwise be difficult or impossible. In this project, assistive technology includes systems that help visually impaired users navigate their surroundings.

3. LiDAR (Light Detection and Ranging): A remote sensing technology that uses laser light to measure distances and create precise three-dimensional information about the surrounding environment. LiDAR sensors are a critical component of our navigation system.

4. Obstacle Detection: The process of identifying and locating objects or barriers in the environment that may pose a risk to visually impaired users. Effective obstacle detection is vital for ensuring safe navigation.

5. User Interface (UI): The means by which users interact with a system or device. A well-designed UI is crucial for ensuring that visually impaired users can easily understand and control the system.

6. Accessibility: The design and implementation of products, devices, services, or environments that ensure they can be used by all individuals, including those with disabilities. Our project prioritizes accessibility in both hardware and software components.

5.4 User Requirements

This section outlines the user requirements for our navigation system designed for visually impaired individuals. Each requirement is numbered and linked to specific use cases, ensuring clarity and traceability. The main use cases involved are navigating to a destination and avoiding obstacles. There will be other use cases surrounding the users maintenance the deployment of the product and fail safes detailed in the use cases section.

Table 5.1: User Requirements Table

Requirement	**Priority**	**Use Case(s)**
Functional Requirement 1 (reqfDirections) *The system shall give directions to a walkable Apple maps destination.*	MustHave	UC_1
Functional Requirement 2 (reqfObstacleAvoid) *The system shall give directions to avoid obstacles.*	MustHave	UC_2

Table 5.1: User Requirements Table

Requirement	Priority	Use Case(s)
Functional Requirement 3 (reqfCustomizable) *The system shall have directions given at a customizable intensity.*	CouldHave	UC_2,UC_1,UC_4, UC_5
Interface Requirement 1 (reqiReset) *The system shall have a reset button.*	ShouldHave	UC_3
Interface Requirement 2 (reqiPower) *The system shall have a power on and off button.*	MustHave	UC_4
Interface Requirement 3 (reqiPairing) *The system shall have a Bluetooth pairing button or automatically become discoverable when disconnected from a phone.*	MustHave	UC_3UC_4
Interface Requirement 4 (reqiAccessibility) *The mobile application shall have accessibility features integrated from the iPhone.*	MustHave	UC_1,UC_2,UC_4, UC_5

5.5 System (Constraints) Requirements

This section expands on the system constraint requirements that affect the production of C-ALL. These requirements are crucial to the development process as we intend to complete the project in a timely and efficient manner. Each requirement is numbered and linked to specific use cases, ensuring clarity and traceability.

Table 5.2: System Requirements Table

Requirement	Priority	Use Case(s)
Constraint Requirement 1 (reqcAppleRequirement) *The system shall only be accessible through Apple iOS applications.*	MustHave	UC_4

Table 5.2: System Requirements Table

Requirement	Priority	Use Case(s)
Constraint Requirement 2 (reqcBluetooth) *The system shall only communicate with the mobile application through Bluetooth.*	MustHave	UC_4

iOS Mobile Application

- Currently, the only mobile devices containing a LiDAR sensor are Apple's iPhones, beginning from the 12 Pro models and onward. Therefore, the mobile application must be coded in Swift, the official programming language for iOS applications. The application will be designed and optimized for iOS, requiring that the system may only be used with the iPhone. The mobile application shall not be accessible on non-iOS devices. The application will be made available only on the App Store where users can download it if their device meets the hardware requirements.

Bluetooth Connection

- The system will act as a Bluetooth device capable of connecting to the user's iPhone. Via Bluetooth, the mobile application will instruct the system on navigation and obstacle avoidance. The system will not contain WiFi modules or wired connection; it will only be Bluetooth compatible and will receive all information from the mobile application. The user will be able to begin Bluetooth pairing through the system and connect to it in their mobile device settings.

5.6 Non-functional (Quality) Requirements

This section emphasizes the quality priorities for C-ALL, as the system is designed to serve as an aid for the visually impaired and it is essential that the system works quickly and effectively. These requirements outline hardware necessities for final developments. Each requirement is numbered and linked to specific use cases, ensuring clarity and traceability.

Table 5.3: Non-functional Requirements Table

Requirement	Priority	Use Case(s)
Quality Requirement 1 (reqqCommunicationSpeed) *The system shall perform the task with minimal responsiveness and communication delay.*	MustHave	UC_1,UC_2, UC_5
Quality Requirement 2 (reqqBatteryLife) *The system shall perform the task with ample battery usage. The system will notify user when device is at 10 percent battery.*	ShouldHave	UC_1,UC_2

Communication Speed

- The communication between the mobile application and system must hae minimal delay, as the system is expected to guide a user in real-time. Any calculations done in the mobile application from data gathered by LiDAR must be efficient and the logic behind the decisions should be relayed quickly. The user should have information on incoming obstacles or upcoming directions with enough time to react comfortably.

Battery Life

- The system will be a standalone device powered through battery. To ensure that it remains useful, it should have a long enough battery life to instruct a user to their destination while avoiding obstacles in their path. The battery should be rechargeable or replaceable and consistently offer ample battery life to successfully aid a user without interruption.

5.7 Domain (Business) Requirements

This section outlines the business rules and regulations that impact the development and functionality of C-ALL for visually impaired individuals. Understanding these requirements is crucial for ensuring that our project aligns with industry standards and meets the needs of our stakeholders. Each requirement is numbered and linked to specific use cases, ensuring clarity and traceability.

Table 5.4: Domain Requirements Table

Requirement	Priority	Use Case(s)
Business Requirement 1 (reqbTutorial) *The system shall have a tutorial for stakeholder understanding.*	CouldHave	UC_1,UC_2,UC_4
Business Requirement 2 (reqbUpdates) *The mobile application shall offer software and firmware updates to the system for maintenance.*	ShouldHave	UC_4, UC_5

Regulatory Compliance

- The system must comply with the Americans with Disabilities Act (ADA), which mandates accessibility features for individuals with disabilities. This includes ensuring that the navigation system is usable by visually impaired users in public spaces.
- The product must adhere to relevant safety regulations, ensuring that the use of technology does not pose risks to users while navigating their environment.

User-Centric Design Principles

- The system should be designed with a user-centric approach, prioritizing the needs and preferences of visually impaired users. Feedback from potential users will be gathered throughout the development process to ensure the system is intuitive and effective.
- Accessibility features must be integrated into the user interface, ensuring that visually impaired users can easily interact with the system through audio feedback and tactile elements.

Market Considerations

- The system must be competitive in the assistive technology market, providing unique features that distinguish it from existing solutions.
- Pricing strategies should be developed to ensure that the navigation system is affordable for visually impaired individuals and organizations serving this demographic.
- The system should be made available to healthcare and insurance companies to offer to their clients and reach the target demographic.

Sustainability and Scalability

- The solution should be developed with sustainability in mind, considering the environmental impact of hardware and software components throughout the project lifecycle.
- The system must be scalable to accommodate future enhancements, including integration with new technologies or expansion to serve additional user needs.

Chapter 6

Use Cases

– Ahmad Shah, Neeti Mistry, Sohan Chatterjee, Sara Gaber

This chapter presents the use case diagrams designed to fulfill the specified requirements of the project. Use case diagrams are crucial in illustrating how various actors interact with the system, highlighting the functionalities that will be delivered. Each diagram serves as a visual representation of the system's behavior in response to external requests, capturing the essential interactions between users and the system components.

The use case diagrams are categorized based on the primary functionalities of the C-ALL (Cognitive Assistance with LiDAR Localization) project, ensuring that all critical user interactions are accounted for. By mapping out these interactions, we can better understand the requirements and expectations of end users, particularly those who are visually impaired.

The chapter will include the following elements:

1. **Description of Actors:** We will identify and describe the primary and secondary actors involved in the system. This includes users (the visually impaired individuals utilizing the C-ALL system), as well as secondary actors such as feedback devices and mobile applications that support navigation.

2. **Use Case Scenarios:** Each use case will outline specific scenarios depicting how users will engage with the system. These scenarios will encompass various functional requirements, such as navigating to a destination, avoiding obstacles, and receiving real-time feedback.

3. **Functional Requirements Mapping:** We will correlate the use cases with the project's functional requirements, demonstrating how each diagram addresses specific needs and functionalities outlined in the requirements analysis. This mapping will clarify which requirements are met by each use case and how they contribute to the overall functionality of the C-ALL system.

4. **Visual Representation:** Accompanying each use case description will be a corresponding diagram. These diagrams will visually represent the interactions, showing the relationships between actors and use cases. The use of standardized notation will facilitate understanding and communication among stakeholders.

5. **Implications for Development:** Lastly, we will discuss how these use case diagrams inform the development process. By establishing a clear understanding of user interactions and system responses, we can prioritize features, identify potential challenges, and ensure that the final product aligns with user needs and expectations.

Through these use case diagrams, this chapter aims to provide a comprehensive overview of the requirements and functionalities of the C-ALL project, laying the groundwork for successful development and implementation. The clarity and structure offered by these diagrams will be instrumental in guiding the design and ensuring that the project meets its intended objectives.

6.1 Table of Use Cases

Table 6.1: Use Cases Table

Use Case	Requirements	Name and Description
UC_1	$reqkFunctional_1$, $reqkInterface_4$, $reqkQuality_1$, $reqkQuality_2$, $reqkBusiness_1$,	ucNavigate Give directions to the User to navigate to their desired destination
UC_2	$reqkFunctional_2$, $reqkFunctional_3$, $reqkInterface_4$, $reqkQuality_1$, $reqkQuality_2$, $reqkBusiness_1$,	ucObstacleAvoid Give directions to the User to avoid obstacles
UC_3	$reqkFunctional_3$, $reqkInterface_1$, $reqkInterface_3$,	ucHandleErrors Notify the User of a system failure if directions can no longer be given reliably
UC_4	$reqkInterface_2$, $reqkInterface_3$, $reqkInterface_4$, $reqkConstraint_1$, $reqkConstraint_2$, $reqkBusiness_1$, $reqkBusiness_2$,	ucSetup Set up and calibrate the system for each user
UC_5	$reqkBusiness_2$, $reqkInterface_4$, $reqkQuality_1$, $reqkFunctional_3$,	ucUpdateSoftware Allow users to download and install software updates to improve performance, add new features, and fix potential issues

6.2 Use Case Diagram

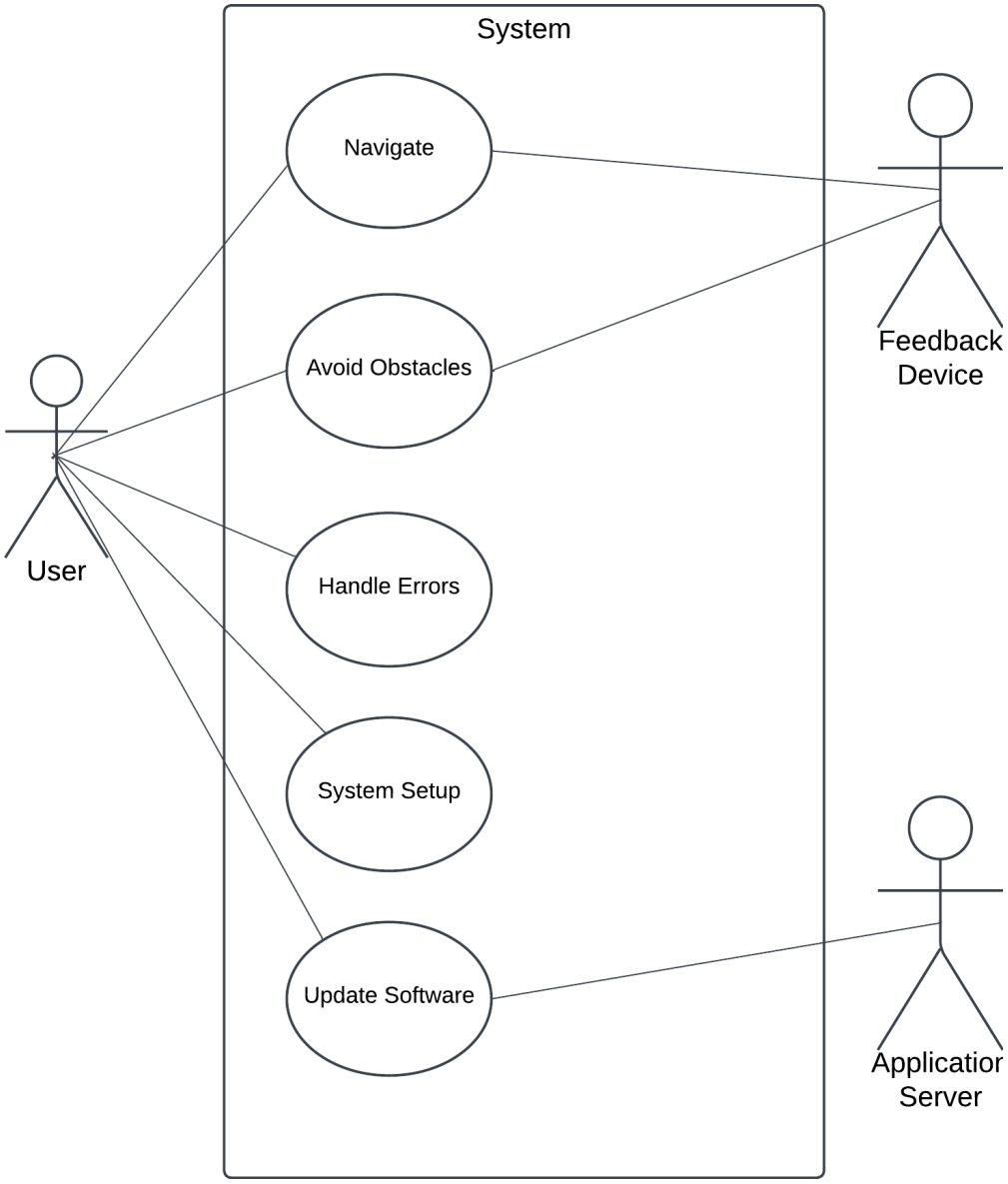

Figure 6.1: Complete Use Case Diagram, with reference to use cases UC_1,UC_2,UC_3,UC_4, UC_5

The use case diagram illustrates the interaction between the User, the System, and the Feedback Device within the C-ALL system. It highlights four primary functionalities that the system provides to assist visually impaired users in navigation:

1. Navigate – The user interacts with the system to move through an environment, receiving feedback and guidance for safe travel.
2. Avoid Obstacles – The system helps the user detect and steer clear of obstacles using LiDAR technology, ensuring a smooth path.

3. Handle Errors – The system is designed to identify and manage errors in real-time, such as incorrect obstacle detection or feedback delays.

4. System Setup – The user initializes and configures the system before use, ensuring proper functionality.

5. Making Updates to the Software – The system allows users to download and install software updates that enhance functionality, fix issues, and improve navigation performance.

Additionally, the Feedback Device interacts with the Navigate and Avoid Obstacles use cases, indicating that it provides real-time sensory feedback (haptic alerts) to assist the user in navigation.

This diagram effectively captures the core interactions necessary for the system to function and provides a clear representation of how users engage with the technology.

6.3 Use Cases

Table 6.2: Use Case Navigation

Use Case 1 (ucNavigate) *Navigate*
Requirements: reqfDirections, reqiAccessibility, reqqCommunicationSpeed, reqqBatteryLife, reqbTutorial,
Diagrams: Figure 6.1, Figure 8.3
Brief description: Give directions to the User and feedback device to navigate to the desired destination
Primary actors: User
Secondary actors: Feedback device
Preconditions: The system has been set up properly
Main flow: 1. The user inputs a destination 2. The system calculates the route to the destination 3. The system sends directions to the feedback device
Postconditions: None
Alternative flows: 1. The user changes destination and the system recalculates directions

Table 6.3: Use Case Obstacle Avoid

Use Case 2 (ucObstacleAvoid) *Avoid Obstacles*
Requirements: reqfObstacleAvoid, reqfCustomizable, reqiAccessibility, reqqCommunication-Speed, reqqBatteryLife, reqbTutorial,
Diagrams: Figure 6.1, Figure 8.4
Brief description: Give directions to the User and feedback device to avoid obstacles
Primary actors: User
Secondary actors: Feedback device
Preconditions: The system has been set up properly
Main flow: 1. The user turns on obstacle avoidance 2. The system scans the room using LiDAR 3. The system calculates the clearest path of motion 4. The system sends directions to the feedback device
Postconditions: None
Alternative flows: None

Table 6.4: Use Case Handle Errors

Use Case 3 (ucHandleErrors) *Handle Errors*
Requirements: reqfCustomizable, reqiReset, reqiPairing, reqbUpdates,
Diagrams: Figure 6.1, Figure 8.5
Brief description: Notify the User of a system failure if directions can no longer be given reliably
Primary actors: User
Secondary actors: Feedback device
Preconditions: None
Main flow: 1. The system detects an error 2. The system notifies the user 3. If offline system still available then: 3.1. The system continues to give directions to avoid obstacles 4. The user troubleshoots the error
Postconditions: None
Alternative flows: None

Table 6.5: Use Case System Setup

Use Case 4 (ucSetup) *System Setup*
Requirements: reqiPower, reqiPairing, reqiAccessibility, reqcAppleRequirement, reqcBluetooth, reqbTutorial, reqbUpdates
Diagrams: Figure 6.1, Figure 8.6
Brief description: Set up and calibrate the system for each user
Primary actors: User
Secondary actors: Feedback device Distributors
Preconditions: None
Main flow: 1. The user powers on all devices and connects the feedback device to their iPhone 2. The user launches an initial calibration mode 3. The system calibrates the feedback device 4. The system displays results of the calibration
Postconditions: Have a fully operational system
Alternative flows: None

Table 6.6: Use Case Update Software

Use Case 5 (ucUpdateSoftware) *Making Updates to the Software*
Requirements: reqfCustomizable, reqiAccessibility, reqqCommunicationSpeed, reqbUpdates,
Diagrams: Figure 6.1
Brief description: Allow users to update the system software to ensure the latest improvements, bug fixes, and new features are available
Primary actors: User
Secondary actors: System Application Server
Preconditions: User must have an internet connection and sufficient device storage
Main flow: 1. The system notifies the user of an available update 2. The user initiates the update process through the mobile application 3. The system downloads and verifies the update 4. The system installs the update and notifies the user upon completion 5. The user restarts the system to apply changes
Postconditions: The system runs on the latest software version with applied updates
Alternative flows: **Update Fails** 1. If the update fails due to connectivity issues, the user is prompted to retry 2. If the update is incompatible with the device, an error message is displayed, and a rollback is initiated

Chapter 7

User Interface Design

– Neeti Mistry, Sara Gaber

7.1 Introduction

The Cognitive Assistance with LiDAR Localization (C-ALL) project is designed to empower visually impaired individuals with enhanced mobility and independence by providing a real-time navigation solution. Utilizing LiDAR technology integrated into smartphones, the C-ALL system uses ARKit to collect depth information and constructs a structured depth map. The mobile applications and hardware device communicate with one another, which allows for navigation and obstacle avoidance instructions to be sent to the hardware device. This feedback assists users in avoiding obstacles, recognizing pathways, and navigating complex environments with minimal reliance on human assistance. By offering an affordable, user-friendly device, C-ALL aims to address the accessibility and cost challenges that often limit the availability of assistive technology for visually impaired individuals.

The scope of our preliminary design includes the core functional aspects of the user interface and the essential interaction flows required for basic navigation. This design covers key use cases, such as starting and stopping navigation, receiving real-time obstacle alerts, and managing user settings. Initial UI elements include the home, navigation, obstacle detection, settings, and support screens, each tailored to provide clear and accessible feedback that aligns with user needs. The design focuses on delivering essential functionality in an intuitive layout, establishing the foundation for further development and user testing.

7.2 UI Design

7.2.1 User Persona

To design a user-friendly navigation system for visually impaired individuals, we identified key User Personas to capture the primary needs, challenges, and goals of our diverse user base. Our personas include visually impaired individuals with varying levels of mobility independence, comfort with technology, and access to existing assistive technology.

Below are three primary personas that represent our target users:

Persona 1: Emily – Independent Commuter

- Age: 33
- Background: Emily is a full-time employee who is legally blind and has moderate experience using assistive technologies.
- Motivation: She wants to navigate urban spaces independently, particularly for commuting to work using public transportation.
- Challenges: She faces difficulties interpreting crowded and unfamiliar environments; relies on limited tools that are either costly or don't offer real-time feedback.
- Goals:
  - Navigate independently through complex areas such as public transportation stations and busy streets
  - Receive accurate, real-time feedback to safely avoid obstacles and stay in sync with the movement of other commuters
  - Receive clear, directional cues that are easy to interpret and do not overwhelm her
- Tasks:
  - Input her destination into the C-ALL navigation system before beginning her commute
  - Rely the feedback from the glove and app to follow directions in crowded areas, adjusting her pace as needed
  - Identify and respond to alerts for obstacles or sudden changes in the environment
- Scenario:
  - Emily is navigating a crowded bus station. She opens the C-ALL app on her iPhone, selects her work location as the destination, and receives immediate (obstacle avoidance) feedback guiding her to the correct bus. With the app's haptic cues, she is able to avoid obstacles, locate the correct boarding area, and safely boards the bus. During the ride, the app provides real-time updates, helping her prepare to disembark at the right stop and reach her destination.

Persona 2: Samir – Beginner with Assistive Tech

- Age: 55
- Background: Samir recently lost his vision and is new to assistive technology.
- Motivation: He wants a simple and affordable navigation tool to gain some independence in his local neighborhood.
- Challenges: He struggles with complex interfaces and cannot afford high-end assistive devices.
- Goals:
  - Use an easy-to-operate device to navigate familiar locations, like his neighborhood
  - Receive gentle and understandable guidance to avoid nearby objects and obstacles

  - Build confidence in independently moving within his community over time

- Tasks:
  - Launch the C-ALL system and select a pre-defined route in his neighborhood
  - Follow the prompts provided by the glove to navigate safely, while adjusting feedback settings to match his comfort level
  - Gradually explore more routes as he gains confidence and familiarity with the app's features

- Scenario:
  - Samir decides to take a walk to a nearby park. He opens the C-ALL app and chooses the simplest route. As he walks, he follows the motions felt on his hand from the glove that keep him on course and aware of any approaching obstacles. With each outing, Samir feels increasingly self-reliant and at ease with the app, empowering him to explore more of his neighborhood.

Persona 3: Ava – Student

- Age: 20
- Background: Ava is a college student with partial vision loss who frequently uses digital devices for academic and social activities.
- Motivation: She seeks a navigation tool compatible with her smartphone to help her move independently on campus and in nearby urban areas.
- Challenges: She needs a device that integrates with her digital lifestyle, offering connectivity and compatibility with smart environments.
- Goals:
  - Navigate her campus and nearby areas seamlessly using her smartphone and glove-based haptic feedback
  - Receive accurate navigation feedback while multitasking with other digital applications
- Tasks:
  - Activates the C-ALL app on her iPhone, with the glove connected for haptic navigation cues
  - Follows directional prompts through specific haptic patterns on the glove, such as vibrations or taps to indicate turns or hazards
  - Adjusts glove settings if needed, to personalize intensity or feedback patterns for different environments
- Scenario:
  - Ava is heading to her university library and begins by opening the C-ALL app on her phone and putting on the glove. As she walks across campus, the glove provides distinct patterns on her hand to guide her along the route. For example, a pulsing sensation on her right hand alerts her to an upcoming right turn, while a steady vibration indicates an obstacle ahead. With the glove's cues, Ava safely navigates around construction zones and crowded pathways without needing to check her phone, allowing her to walk confidently and independently.

7.2.2 User Interface Design

For the Cognitive Assistance with LiDAR Localization (C-ALL) project, we created a preliminary Paper Prototype of the user interface that emphasizes simplicity, ease of interpretation, and integration with smartphone capabilities. The design includes core screens and flows based on key user stories, goals, and scenarios, capturing the needs of each persona.

Key Features and Screens:

1. Home Screen
   (a) Elements: Large, accessible buttons for key actions (e.g., ”Start Navigation,” ”Settings”).
   (b) Functionality: Users can quickly begin navigation or access essential settings without needing to navigate through complex menus.
   (c) Rationale: Provides an intuitive starting point for users like Samir, who may be less familiar with digital interfaces.
2. Navigation Screen
   (a) Elements: Location entry with audio supported keyboard, play button.
   (b) Functionality: Enter desired locaton, given an option to play directions as well.
   (c) Rationale: For users like Emily, who require fast and accurate feedback in crowded or dynamic environments, this screen prioritizes straightforward and actionable guidance.
3. Settings Screen
   (a) Elements: Options for connecting to Bluetooth, offline mode, adjusting feedback intensity, language preferences, system status and reconnect.
   (b) Functionality: Users can personalize the experience to match their level of comfort and sensory needs and conduct system setup and troubleshooting.
   (c) Rationale: Allows customization for all personas, enabling them to tailor the feedback style and intensity for different environments.

Preliminary UI Prototype: Below are digital mock-ups of the main screens and interactions of the interface prototype of our app:

- Home Screen: Features large, high-contrast buttons for accessibility. The ”Start Navigation” button is prominently placed to encourage straightforward navigation initiation.
- Navigation Screen: The interface is minimalist, displaying only necessary navigation input, supported by audio and haptic feedback for real-time obstacle and directional guidance.
- Settings Screen: Simplified options for ease of access, with all necessary safety options.

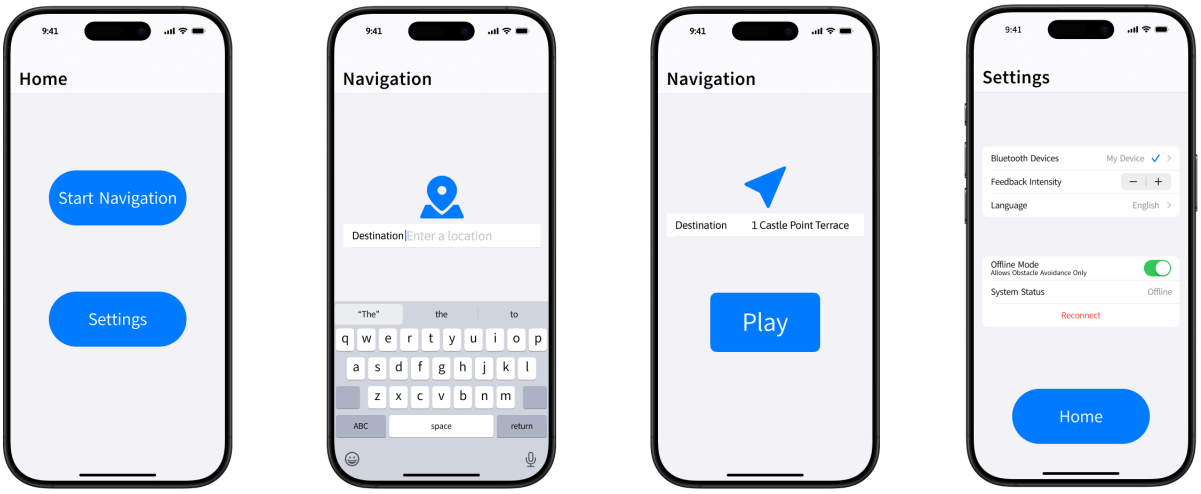

Figure 7.1: UI Prototype

Chapter 8

System Architecture

– Neeti Mistry, Sara Gaber, Sohan Chatterjee

The system architecture of the C-ALL project is described using the 4+1 View Model, which captures the system's design from multiple perspectives to ensure a comprehensive understanding. The Logical View defines the system's functionality, focusing on key components such as the LiDAR sensor, mobile application, and feedback device. The Development View outlines the organization of the codebase, highlighting modular structures and communication interfaces. The Process View illustrates runtime interactions, emphasizing real-time obstacle detection, data processing, and user feedback delivery. The Physical View details the deployment of system components, such as the wearable device and smartphone integration. The Scenarios View ties these perspectives together through user stories and use cases, demonstrating how the system addresses user needs effectively. This approach ensures a robust, user-centric, and technically sound system design.

8.1 Scenarios View: User Stories

The Cognitive Assistance with LiDAR Localization (C-ALL) project focuses on User Stories that define key functionalities, empowering visually impaired users to navigate their surroundings independently and confidently. These user stories outline specific needs and interactions from the user's perspective, ensuring that the system supports reliable navigation, obstacle avoidance, ease of setup, and accessible interface design. Each story emphasizes essential goals for user experience, guiding development to prioritize safety, ease of use, and adaptability for a wide range of environments and user preferences.

8.1.1 User Story 1: Independent Navigation

As a visually impaired user,
I want to receive real-time direction cues,
So that I can navigate to my destination independently and confidently.

Context: Users rely on timely and precise directions to navigate safely, so accuracy and minimal latency are critical.

Acceptance Criteria:

- Directions are delivered through haptic feedback, based on user preference.
- Directions update in real time as the user progresses along the route.
- The system recalculates and corrects the route if the user deviates or encounters obstacles.
- Upon reaching the destination, the system confirms arrival with a distinct sound or vibration pattern.

8.1.2 User Story 2: Obstacle Avoidance

As a user navigating unfamiliar areas,
I want the system to alert me of any obstacles in my path,
So that I can avoid them without needing additional assistance.

Context: Visually impaired users need reliable and early obstacle detection to make safe adjustments to their path.

Acceptance Criteria:

- The system detects obstacles within a predetermined range and alerts the user with sufficient time to respond.
- Alerts are distinct and consistent, either through specific notifications on the mobile app or movements on the glove that distinguish them from other cues.
- Users are provided with guidance on how to avoid the obstacle (e.g., motions on left glove indicates to move left).
- If the obstacle is temporary (e.g., a moving person), the system recalculates the route as needed.

8.1.3 User Story 3: System Reliability and Handling Errors

As a user dependent on navigation assistance,
I want to be informed immediately if the system encounters an error or failure,
So that I can take the appropriate actions to ensure my safety.

Context: Trust in the device's reliability is critical for users, especially in unexpected situations where system failures could pose risks.

Acceptance Criteria:

- The system performs self-checks periodically to detect any hardware or software failures.
- If a failure occurs, the user is notified through a distinct alert (e.g., a repeated vibration or specific sound).
- In case of a connectivity loss, the system provides stored information or directs the user to a safe location.
- The system logs errors for later diagnostics to help identify and fix recurring issues.

8.1.4 User Story 4: Easy Setup and Personalization

As a first-time user,
I want a straightforward setup and calibration process,
So that I can quickly configure the system to meet my preferences and begin using it effectively.

Context: First-time users need a simple, guided setup that minimizes frustration and ensures proper calibration for effective use.

Acceptance Criteria:

- The setup process includes an initial walk-through tutorial.
- The app offers a personalization menu for adjusting alert frequency, notification types, and navigation preferences.
- Settings are saved across sessions so users don't need to re-configure each time.

8.1.5 User Story 5: Continuous Feedback on Route Progress

As a user moving towards a set destination,
I want periodic feedback on my progress and any deviations from the route,
So that I feel reassured that I am on the correct path.

Context: For visually impaired users, regular feedback is essential to stay informed of their progress and prevent unnecessary detours.

Acceptance Criteria:

- If the user deviates from the route, the system promptly provides correctional feedback.
- Feedback is continuous but unobtrusive, providing enough information without overwhelming the user.

8.1.6 Summary

The scenarios view focuses on capturing and validating the system's behavior through key interactions and use cases. These detailed user stories provide a structured approach to understanding what the user wants and why, providing a high-level goal. User stories are commonly used in Agile development for quick, high-level understanding of user needs.

8.2 Logical View

To map out the Logical View, a class diagram was created to show the structure of the system for the functionality for users.

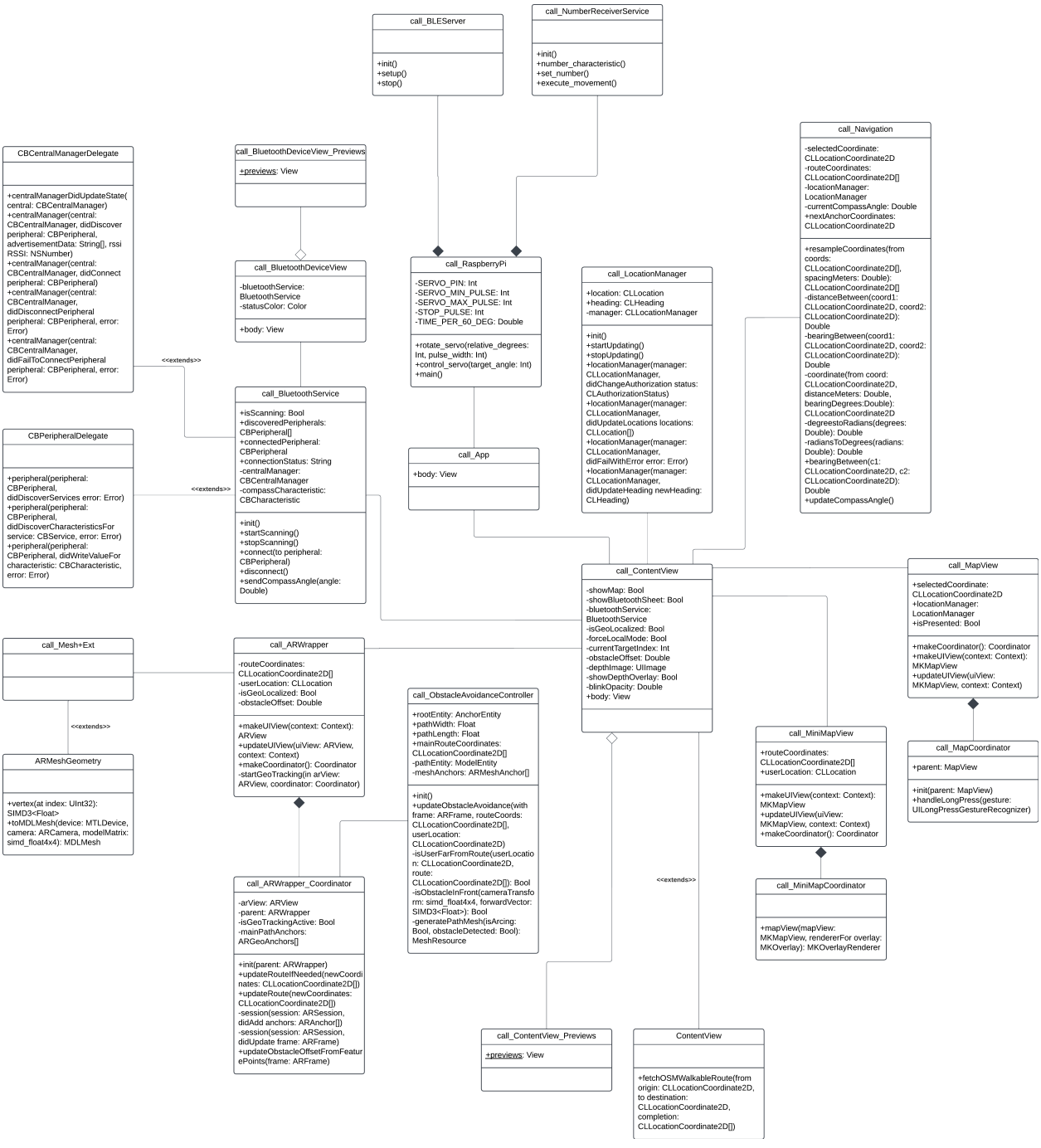

Figure 8.1: Class Diagram

The Class Diagram describes the core entities of the mobile application for providing LiDAR-based cognitive assistance to visually impaired users. It includes classes that represent the different components, such as user interface, location services, LiDAR usage, and connection to hardware. Each class in the diagram represents a key part of the system's functionality, such as interacting with the LiDAR sensor to collect depth information, processing the data to generate navigational guidance, and communicating it to the user.

Upon completion, the attributes and methods of these classes will indicate how the system handles data collection, processing, and user interaction. The main relationships between the classes should show how each entity relies on others to operate, such as the mobile application class using the processed navigation data to deliver haptic cues to the user. The current diagram provides insight into how different parts of the application collaborate to achieve the goal of providing real-time navigational assistance with obstacle avoidance.

** [2] *UML Class Diagram Tutorial*
In our project, we used the source from Visual Paradigm's UML Class Diagram tutorial to guide us in developing the class diagram for the system architecture. This source provided detailed insights into the structure and design principles of UML class diagrams, which are essential for organizing and representing the key components and their relationships within our system. By following the tutorial, we were able to better understand how to define classes, attributes, methods, and the relationships between various objects in our software, ensuring that our class diagram accurately reflects the core components of our application and their interactions. **

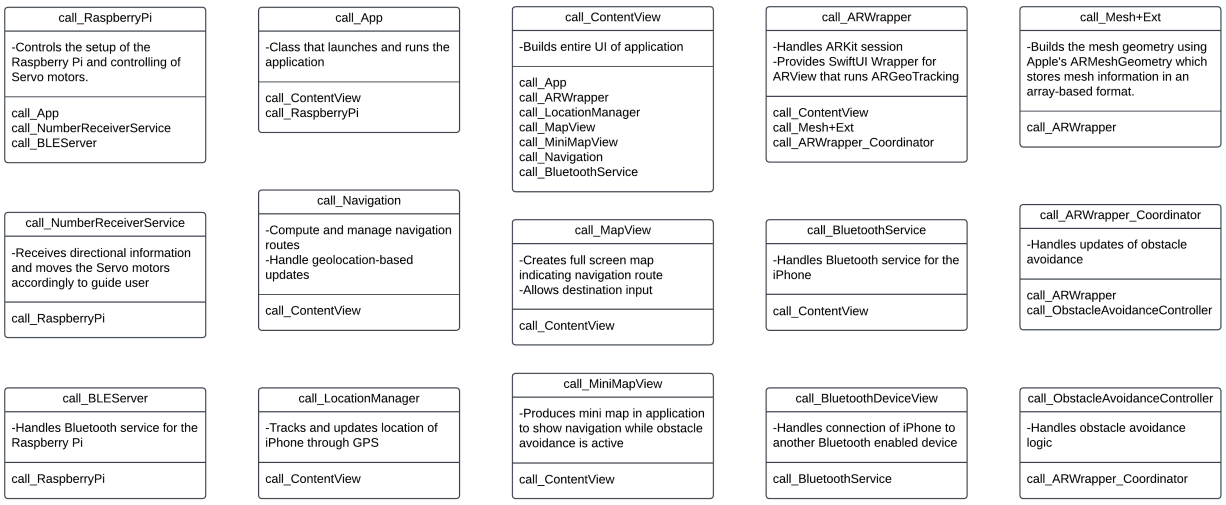

Figure 8.2: CRC Cards

The CRC Cards indicate the class-responsibility-collaboration between all classes involved in the system. The ContentView class is the driver of the program, where the entire UI is built to be given to the user in a digestible format. ARWrapper handles any obstacle avoidance logic and Navigation handles the directional route. The mobile application class also relays navigational and obstacle avoidance information to Raspberry Pi to initiate the hardware portion of the system. The Raspberry Pi class processes this information and directs the Servo motors through a MotorControl to move and ultimately guide the user.

8.3 Process View

To map out the Process View of each use case the following activity diagrams were created for each:

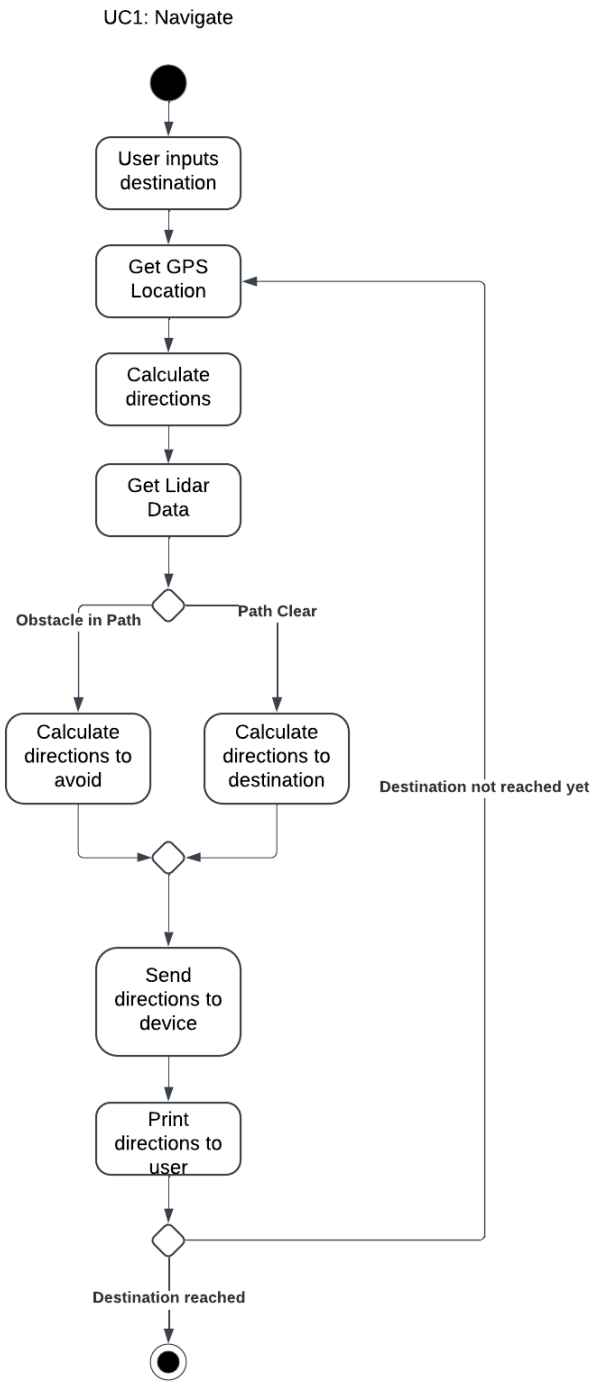

Figure 8.3: Activity Diagram, with reference to use case UC_1

This Activity Diagram shows the process of navigating around an obstacle detected in the user's path. It starts with the LiDAR sensor scanning the environment to identify objects and obstacles in real-time. If an obstacle is detected, the Navigation Processor calculates a new path that avoids the obstacle while keeping

the user on their intended route. This path is then communicated back to the Mobile Application, which in turn, sends commands to the glove.

The glove provides directional cues to the user, such as a gentle vibration to indicate a left or right turn. This process continues until the obstacle is cleared and the user is safely back on their original path. This use case is crucial for ensuring that the user can navigate independently without running into obstacles, and it leverages multiple system components (LiDAR, mobile app, glove) to provide seamless and safe guidance.

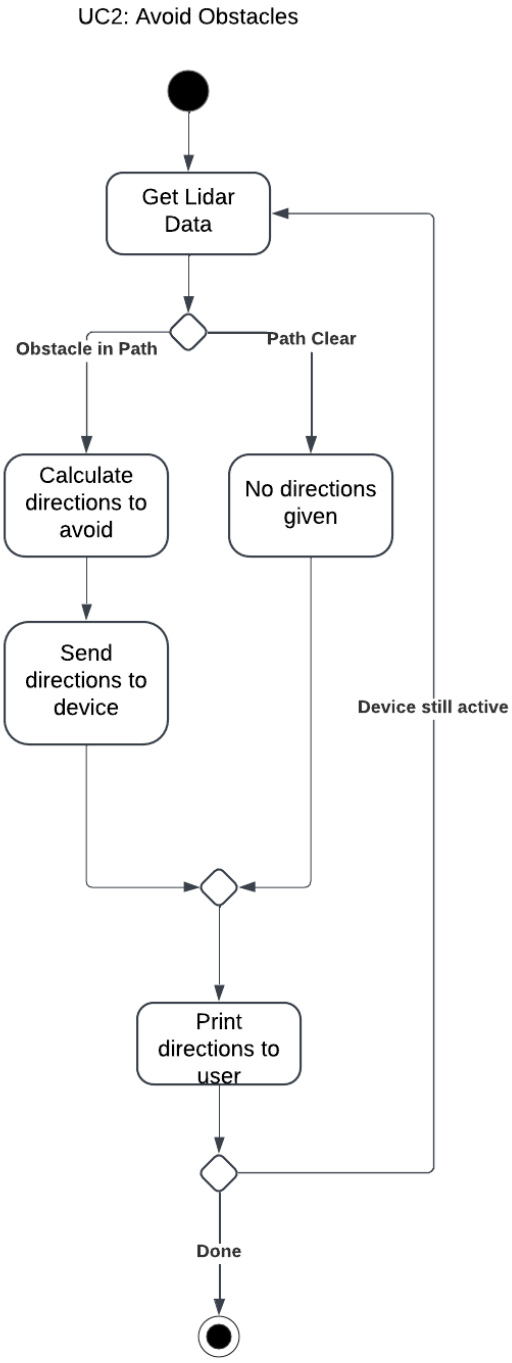

Figure 8.4: Activity Diagram, with reference to use case UC_2

This activity diagram demonstrates how the system proactively manages obstacle detection and avoidance using real-time LiDAR data. By calculating new directions when an obstacle is detected and providing

clear feedback through connected devices, the system ensures user safety and navigation efficiency. The diagram emphasizes the system's adaptability and responsiveness, showing how it continually monitors and recalculates paths to provide seamless guidance in dynamic environments. This process helps users maintain mobility and confidence while navigating complex areas.

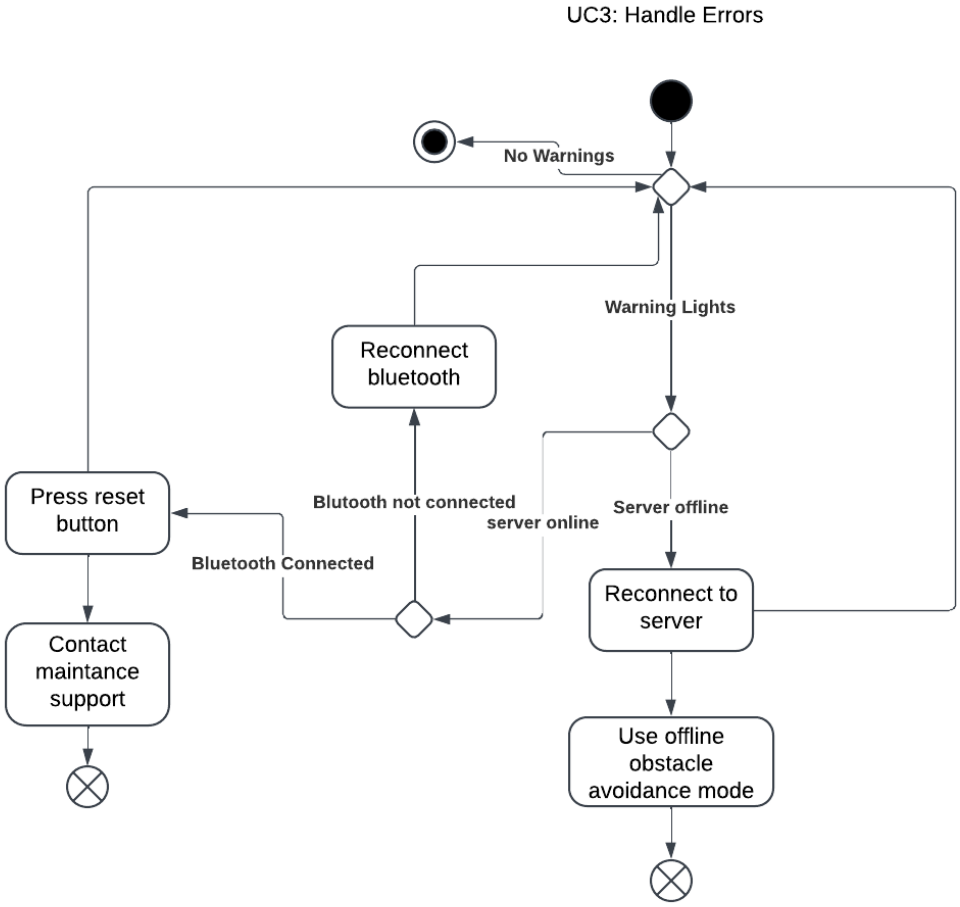

Figure 8.5: Activity Diagram, with reference to use case UC_3

This activity diagram highlights the steps taken to handle system failures, prioritizing user safety and system functionality. It incorporates multiple fallback mechanisms, including automatic reconnection, manual reset, and switching to offline modes, ensuring continued functionality during disconnection events. When all attempts fail, the system escalates the issue to maintenance support, ensuring users receive assistance for unresolved issues. The diagram illustrates the process for maintaining system reliability and user trust in critical scenarios.

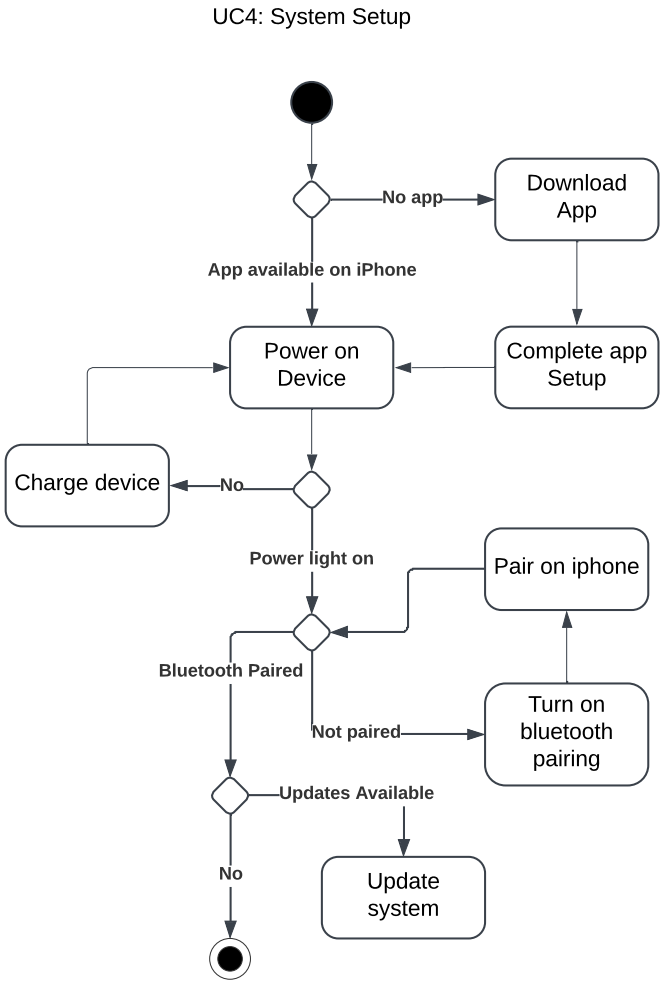

Figure 8.6: Activity Diagram, with reference to use case UC_4

This activity diagram provides a clear workflow for initializing the system, ensuring all components are prepared for proper functionality. It emphasizes essential tasks like powering on the device, pairing via Bluetooth, charging, updating the system, and setting up the app on the user's smartphone. By guiding the user through each step, this process minimizes setup errors and ensures a smooth and efficient onboarding experience. The diagram demonstrates the system's user-centric design, focusing on accessibility and reliability during the setup phase.

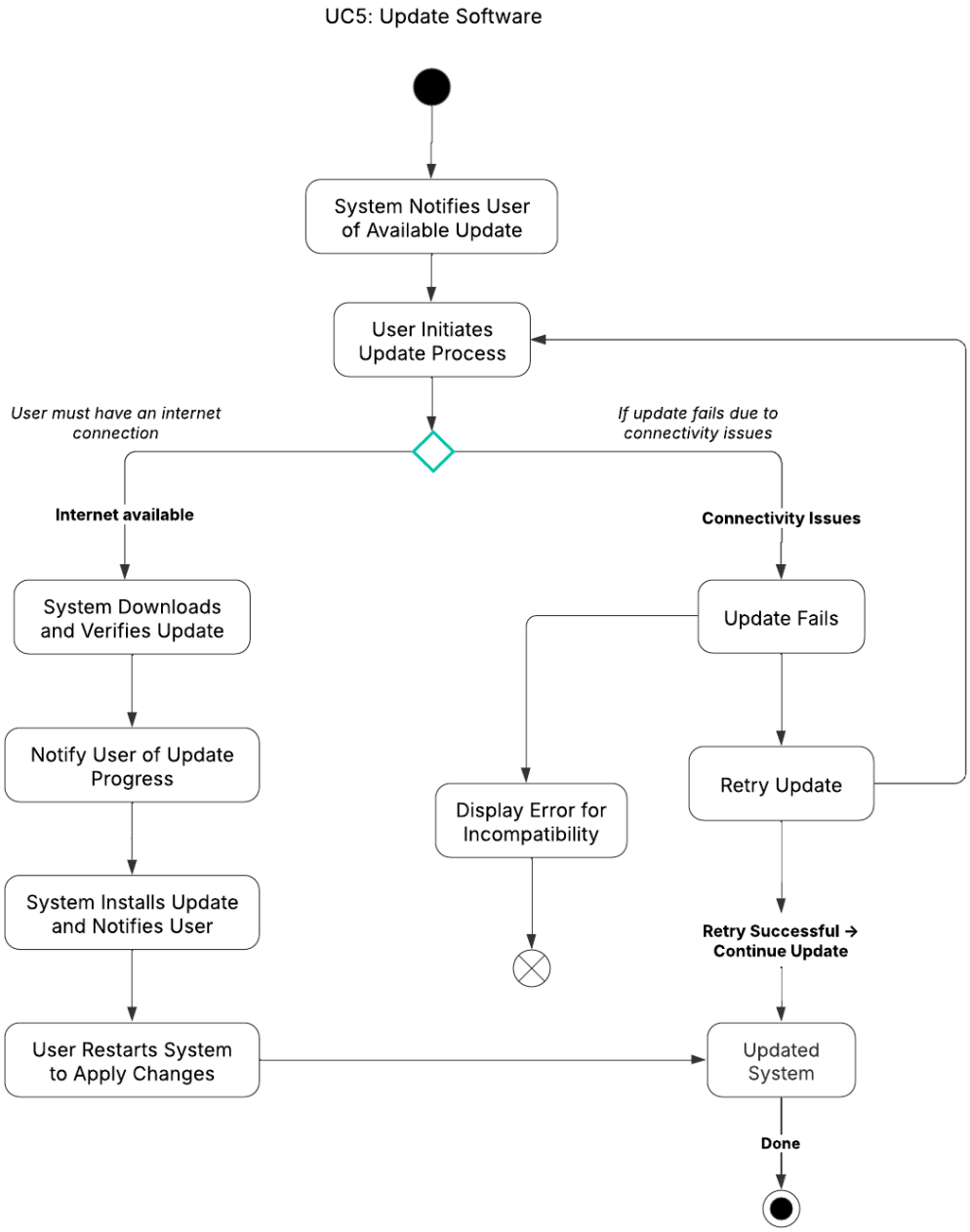

Figure 8.7: Activity Diagram, with reference to use case UC_5

The UC5: Update Software activity diagram illustrates the process of updating software within a system, covering both successful and unsuccessful update scenarios.

1. Initiation
   - The process begins when the system notifies the user about an available update.
   - The user then initiates the update process.

2. Internet Connectivity Check
    - If an internet connection is available, the system proceeds with downloading and verifying the update.
    - If there are connectivity issues, the update fails, and the system allows the user to retry.
3. Successful Update Path
    - The system downloads and verifies the update.
    - It notifies the user of update progress.
    - The system installs the update and prompts the user to restart the system to apply changes.
    - After restarting, the system is updated successfully, marking the end of the process.
4. Failure Handling
    - If an update fails due to incompatibility, an error message is displayed, and the process terminates.
    - If the failure is caused by connectivity issues, the system provides an option to retry.
    - Upon a successful retry, the update process continues from the verification step until completion.

This activity diagram ensures a structured, user-friendly software update process, handling common errors such as connectivity failures and incompatibility.

To further demonstrate the process across components the following Sequence Diagram was created:

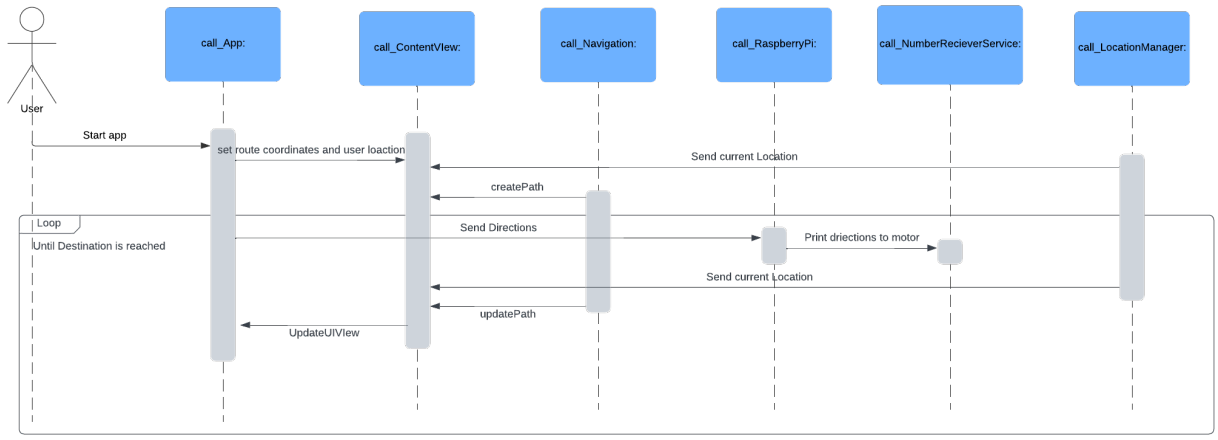

Figure 8.8: Sequence Diagram

This sequence diagram illustrates the interaction between the user, the C-ALL mobile application, the navigation and location services, and the feedback device to facilitate real-time navigation for visually impaired users.

Participants:

1. User: Initiates the system and interacts with the wearable feedback device.
2. C-ALL Mobile App: Manages the user interface, receives location data, and processes navigation information.
3. ContentView Module: Handles UI updates and user input within the app.
4. Navigation Module: Calculates routes and provides step-by-step navigation.
5. Raspberry Pi: Interfaces with the haptic feedback system and processes movement commands.
6. Number Receiver Service: Helps process directional input for guiding the user.
7. Location Manager: Tracks the user's current location and updates the navigation system accordingly.

Workflow:

- Initialization:
  - The user launches the C-ALL mobile app and inputs the destination.
  - The app sets the route coordinates and the user's initial location.
- Path Calculation:
  - The Navigation module computes the optimal route and sends directional data to the app.

- Continuous Navigation Loop (Until Destination is Reached):
  - The app sends updated location data to the Location Manager.
  - The Raspberry Pi processes the navigation directions and translates them into haptic feedback.
  - If necessary, the Number Receiver Service refines the directional cues.
- Obstacle Detection Guidance:
  - If an obstacle is detected, the Navigation module recalculates the path and updates the instructions.
  - The updated directions are relayed through haptic feedback to the user.
- User Feedback UI Updates:
  - The system continuously updates the ContentView module to reflect real-time path changes.
  - The app dynamically adjusts the UI for accessibility, ensuring clear guidance.
- Completion:
  - Once the user reaches the destination, the system stops updating directions and ends the navigation process.

This sequence diagram highlights how C-ALL ensures a safe, efficient, and user-friendly navigation experience by integrating real-time sensor data, intelligent path recalculations, and wearable haptic feedback technology.

8.4 Development View

Below is a Component Diagram to illustrate the Development View of the system's interaction.

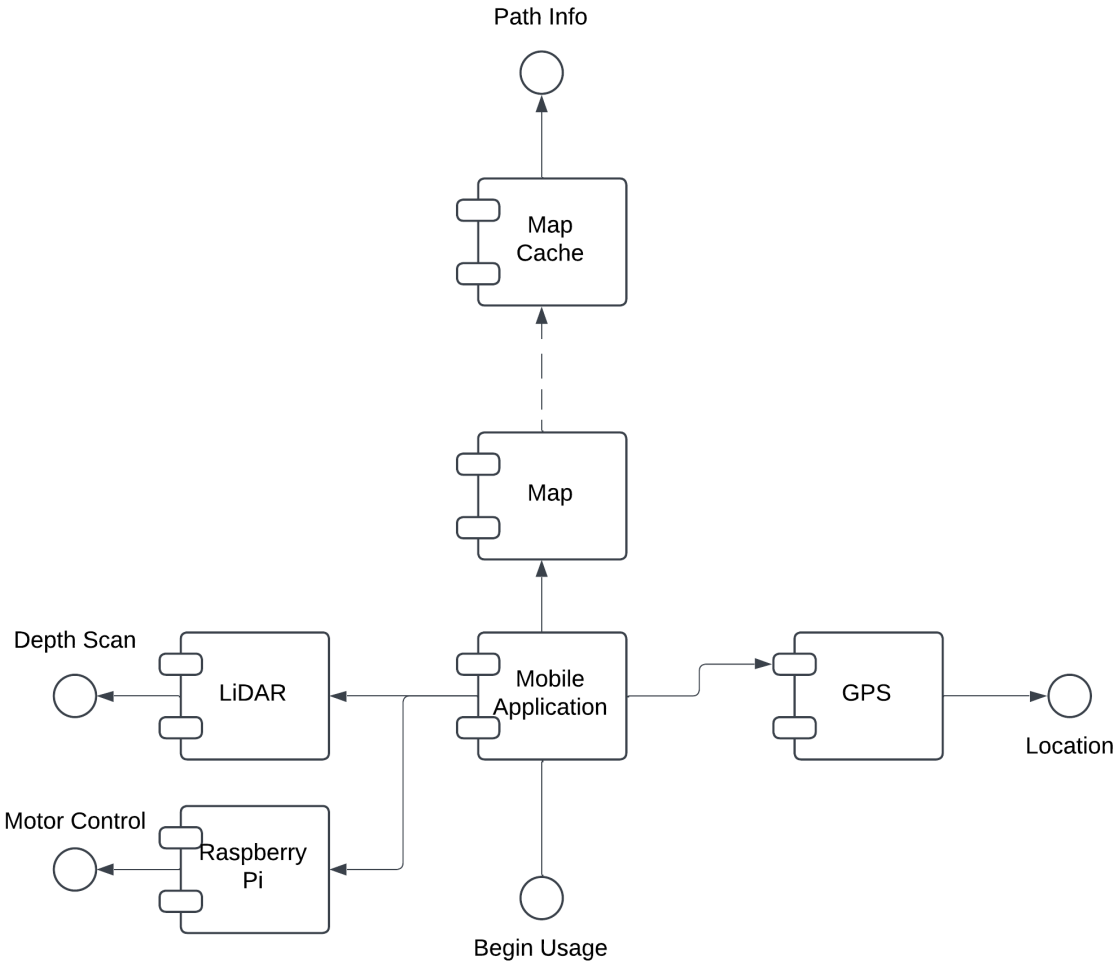

Figure 8.9: Component Diagram for C-ALL

The main components of the system can be broken up into four areas: the mobile application, LiDAR localization, GPS navigation, and Raspberry Pi motor control. Team members will collaborate by ensuring continuity between all four areas to prevent delays, inefficiency, and conflicting code. The mobile application should provide communication from the LiDAR and GPS navigation to the Raspberry Pi, which then moves the motors. The mobile application should also be the point of contact for the user of the system. Collaboration ensues when confirming that each area functions as per requirements so they can be integrated into the whole system.

8.5 Physical View

The deployment architecture is divided into two primary components: the mobile application running on an iPhone device and the feedback system embedded in a physical glove. To show the Physical View of our system we created the Deployment Architecture Diagram below:

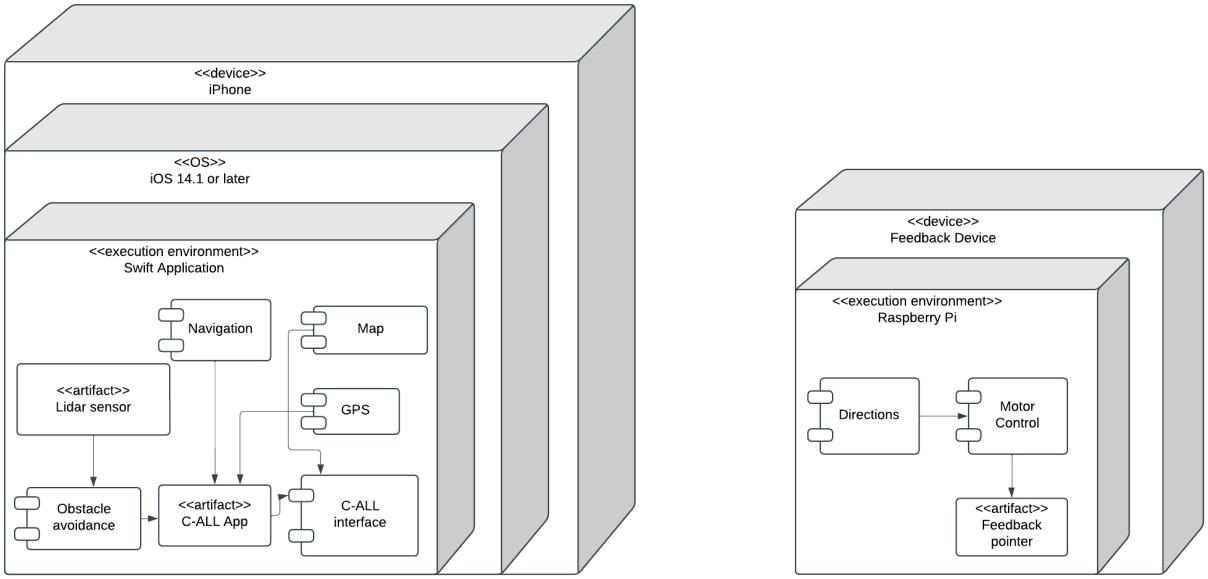

Figure 8.10: Deployment Architecture Diagram

The deployment diagram outlines the physical architecture as follows:

1. Mobile Application
   (a) Device: iPhone running iOS 14.1 or later
   (b) Execution Environment: The application consists of modules for navigation, mapping, and communication with hardware.
      i. LiDAR Sensor: Captures spatial depth data to facilitate obstacle detection and pathfinding
      ii. Navigation Module: Utilizes ARKit to dynamically generate paths in the augmented reality (AR) view
      iii. Map Module: Stores and processes geographical data
      iv. C-ALL Interface: Manages communication between the mobile app and external hardware via Bluetooth
      v. Obstacle Avoidance: Computes safe navigation paths by integrating LiDAR data
2. Feedback System
   (a) Device: Raspberry Pi based feedback device integrated into a wearable glove
   (b) Execution Environment:
      i. Directions Module: Processes navigation instructions from the mobile application
      ii. Motor Control Module: Translates instructions into motor vibrations to provide haptic feedback to the user
      iii. Feedback Pointer: Outputs vibrations to guide the user based on navigation and obstacle avoidance computations

3. Integration Between Components

   (a) Communication: The mobile application sends navigation instructions to the feedback device via Bluetooth (ideally Bluetooth 5.0 for low-latency communication).

   (b) Synchronization: The mobile app ensures real-time processing to synchronize motor responses with navigation updates.

This configuration facilitates seamless interaction between the iPhone running the C-ALL app, the server handling navigation and obstacle avoidance processing, and the Raspberry Pi feedback device, which provides user guidance through directional cues and feedback pointers.

Chapter 9

Preliminary Implementation

– Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee

9.1 Introduction

The Preliminary Implementation Demo showcases the early development stages of the Cognitive Assistance with LiDAR Localization (C-ALL) system. The demo highlights the project's progress by presenting a working prototype (in the initial stages) of the core functionality, focusing on the integration of LiDAR-based obstacle detection. This stage serves as a foundation for further refinements, Usability Testing, and feature expansions in preparation for the final submission.

9.2 Demo Objectives

The Preliminary Implementation Demo focuses on achieving key milestones in the early development of the C-ALL system. The specific goals of the prototype demo include:

1. Validating Key Functionalities
    (a) Demonstrate the system's ability to detect obstacles using LiDAR technology and process this data in real time
2. Establishing Feasibility
    (a) Prove the viability of the system's architecture and design through an initial working prototype
    (b) Lay the groundwork for refining features and addressing identified challenges
3. Gathering Initial Feedback
    (a) Use the prototype demo to collect insights from stakeholders, including team members, sponsors, and potential users, to inform future development
    (b) By achieving these objectives, the demo serves as a critical step toward building a robust, user-centered system that addresses the mobility needs of visually impaired individuals

9.3 Prototype Description and Demo Results

The initial prototype of the C-ALL system demonstrates the core functionality of the navigation and obstacle detection system. The hardware component includes a 3D-printed housing designed to integrate all electronic components, while the software leverages LiDAR technology through a mobile application to process real-time depth data. Haptic feedback mechanisms are used to guide users safely around obstacles. The prototype focuses on showcasing the system's ability to provide real-time, actionable feedback in a simple and user-friendly manner.

The preliminary implementation demo successfully validated key functionalities of the system. LiDAR-based obstacle detection performed effectively in controlled environments. The demo highlighted areas for further refinement, including optimizing feedback responsiveness and improving hardware. Overall, the results confirmed the feasibility of the system and provided valuable insights for the next phase of development.

9.3.1 Hardware Update

The hardware component of the C-ALL system is progressing with the development of the glove-based feedback device. The following updates outline the current status and next steps:

1. Glove Component Construction
   (a) The construction of the glove component is scheduled to begin once the project budget is approved and the necessary materials are procured. This step is critical to integrating the hardware components, including motors and modules, into a functional prototype.
2. Housing Design and Fabrication
   (a) Before the start of the spring semester, a newer version of the hardware housing will be designed using SolidWorks. The updated design will ensure improved compatibility and ergonomics for the glove.
   (b) The housing will be 3D-printed to provide a lightweight and durable solution that accommodates all hardware components securely.
3. Integration with Glove
   (a) The 3D-printed housing, which hosts the motors and modules, will be sewn onto the backside of the glove. This configuration is intended to leave the user's palm and fingers free, ensuring comfort and usability during navigation tasks.

Figure 9.1: Hardware Prototype Drawing

The initial hardware prototype for the C-ALL system was created using Solidworks and a 3D-printer. This prototype serves as a foundational step in the development process, allowing the team to test basic functionality and ensure compatibility with the intended design. Future iterations and refinements of the hardware will be developed once the required budget materials, such as sensors, wires, and other components, are acquired. These additional resources will enable the integration of advanced features and optimizations to enhance the prototype's performance and usability.

These updates reflect the team's commitment to developing a functional and user-friendly hardware solution that meets the needs of visually impaired users. Further refinements will continue as materials arrive and construction begins.

9.4 Implementation Details

The implementation of the C-ALL system combines advanced LiDAR processing, effective communication between hardware and software, and seamless integration to ensure optimal performance. Below are the key technical details:

LiDAR Processing Information

- The system utilizes ARKit to collect depth information using LiDAR sensors, constructing a structured Depth Map.
- Floors are prioritized during segmentation, with specific constraints clarifying navigable spaces.
- Floor meshes are identified as safe, navigable areas, while objects of significant size are flagged as obstacles and anchored to the mesh scene for user awareness.
- A dynamically updating path is generated within the AR view, allowing the system to avoid obstacles in real-time.

Communication Between Hardware and Software

- The mobile application sends navigation and obstacle avoidance instructions to the hardware via Bluetooth.
- Instructions are transmitted as vectors, prompting the motors controlled by an Arduino board to move accordingly.
- This setup ensures real-time responsiveness to navigation and obstacle detection computations.

Integration Between Hardware and Software

- Strong Bluetooth connectivity with low latency is prioritized, ideally leveraging Bluetooth 5.0 for reliability.
- Quick data processing in both the mobile application and the Arduino board ensures efficient communication and feedback delivery, minimizing delays during operation.

These implementation details highlight the system's technical foundation and its capability to provide visually impaired users with reliable and responsive navigation assistance. Further optimizations will enhance performance and usability as the project progresses.

** [3] *Tracking Geographic Locations in AR*
We utilized this source from Apple's ARKit documentation, which provided us with detailed guidance on implementing geographic location tracking in augmented reality (AR) applications. The documentation offered valuable insights into how ARKit's capabilities can be used to track and anchor real-world geographic locations, a crucial component for our project. By integrating spatial awareness and location-based navigation features, we were able to leverage ARKit's tracking abilities to enhance our product's performance. This resource has been instrumental in ensuring that our product provides accurate, real-time location data to visually impaired users, helping them navigate urban environments more effectively and safely. **

** [4] *Understanding World Tracking*
The source "Understanding World Tracking" from Apple's ARKit documentation helps our research by providing a comprehensive overview of how ARKit's world-tracking technology works. This is highly relevant since our project involves integrating spatial awareness or location-based navigation features. The documentation outlines how ARKit uses device sensors and visual input to map and track the physical environment, which is essential for developing an application that aids visually impaired users in navigating urban spaces. By leveraging ARKit's world-tracking capabilities, your system can better detect obstacles, guide users through real-time feedback, and enhance overall navigation accuracy. This helps ensure that our product integrates precise environmental awareness, a critical feature for safe and effective navigation in real-world settings. **

9.5 Challenges Encountered and Solutions

During the development of the C-ALL prototype, several challenges arose that tested the team's problem-solving capabilities. Below is an outline of the primary obstacles faced and the corresponding solutions:

1. Combining Navigation with Pathfinding
   (a) Challenge: Difficulty in translating geo-coordinates into relative positions for an augmented reality (AR) view, complicating the integration of navigation and pathfinding functionalities.

   (b) Solution: Conducted research into existing applications and codebases to identify best practices and integration techniques. Continued iterative testing and refinement of the codebase to achieve accurate and reliable results.

2. Integration Between Hardware and Software

   (a) Challenge: Determining risks and identifying potential bugs in the integration process, particularly between the hardware components (LiDAR sensors) and the software application.

   (b) Solution: Initiated mock integration tests with available hardware parts to simulate real-world conditions. Conducted rigorous testing of software functionality in tandem with hardware simulation to identify and resolve issues early.

3. Evaluating Viability for Visually Impaired Users

   (a) Challenge: Ensuring that the system effectively meets the needs of visually impaired users and provides a viable, practical solution.

   (b) Solution: Scheduled additional interviews and feedback sessions with visually impaired individuals and relevant stakeholders. Gathered detailed requirements to refine the system's design and functionality, ensuring alignment with user expectations.

By addressing these challenges with effective strategies, the team ensured steady progress in prototype development. These experiences also provided valuable insights for future development phases, including refinement, testing, and feature expansion.

9.6 Future Work and Next Steps

As the project progresses, the team will focus on further development and refinement of the C-ALL system to meet the goals outlined for the final submission. Key areas of focus include:

Refining the Prototype

- Improve the accuracy and efficiency of the LiDAR-based obstacle detection algorithms to ensure reliable performance in real-world scenarios
- Enhance the responsiveness of the system, minimizing delays and optimizing user experience
- Develop the mobile app, incorporating user feedback

Expanding Features for Final Submission

- Add additional functionality to the mobile application, including:
  - Improved user interface design for accessibility and ease of use
  - Enhanced navigation features such as route customization and obstacle classification
- Implement logging and analytics capabilities for better debugging and system performance monitoring

Conducting Comprehensive Usability Testing

- Collaborate with visually impaired users to test the system in diverse real-world environments, such as urban streets, public transit stations, and structured indoor spaces
- Gather detailed feedback on system usability, accessibility, and effectiveness, and use these insights to make necessary improvements

Updating System Documentation

- Revise and finalize UML diagrams to reflect the updated system architecture and design
- Prepare detailed user and technical documentation to support the final prototype and its deployment

Preparing for Final Submission and Innovation Expo

- Finalize the system prototype to ensure it is robust, reliable, and ready for presentation
- Conduct rigorous testing to identify and resolve any remaining issues
- Complete all project deliverables, including the final report, presentation materials, and user guides, in preparation for the Innovation Expo

By addressing these focus areas, the team aims to deliver a fully functional, user-centered system that meets the needs of visually impaired individuals and achieves the project's goals.

9.7 Conclusion

The Preliminary Implementation Demo marks a significant milestone in the development of our project. Through this demo, we successfully showcased key functionalities, including LiDAR-based obstacle detection. While challenges such as limited access to LiDAR-equipped devices presented obstacles, the team addressed these issues effectively through iterative testing, debugging, and collaboration. The initial prototype validated the core system design and provided valuable insights into areas requiring further refinement, such as optimizing the obstacle detection algorithms and improving the responsiveness of the feedback system. Moving forward, the focus will be on refining the prototype, expanding features, developing both the hardware and mobile app, conducting comprehensive usability testing with visually impaired users, and preparing the system for final submission. This chapter has outlined the progress made to date and established a clear roadmap for the next steps, ensuring the project will continue to meet its goals and deliver a reliable, User-Friendly Design that enhances the mobility and independence of visually impaired individuals.

Chapter 10

Software Design and Implementation

– Neeti Mistry

10.1 Introduction

This chapter provides a detailed overview of the software design and implementation that powered our system. The purpose of this chapter is to explain how our mobile application and software components were structured, developed, and integrated with the hardware to create a seamless assistive experience for visually impaired users.

The mobile application plays a central role in the overall system. It processes real-time spatial data captured by the iPhone's LiDAR sensor, determines the proximity and direction of obstacles, and transmits navigation cues to the wearable gloves via Bluetooth. The software was designed to be responsive, efficient, and user-centric, with accessibility and real-time feedback as core priorities.

10.2 Software Architecture and Design

Our system architecture follows a modular design pattern that separates concerns across key functional components, enabling flexibility, scalability, and efficient debugging. The core layers of the architecture include:

- **Presentation Layer:** This is the user-facing interface built using SwiftUI. It includes views like `ContentView`, `CompassView`, and `MapView`, all of which interact with observable objects to render dynamic content based on real-time data.
- **Logic and Control Layer:** This layer manages app behavior and user interactions, encapsulated in files such as `LocationManager.swift`, `ARWrapper.swift`, and `ObstacleAvoidance.swift`. It processes LiDAR input, geolocation updates, and AR scene understanding, while managing communication between UI and hardware.
- **Hardware Communication Layer:** Facilitates Bluetooth interactions between the iOS app and Raspberry Pi hardware, allowing for bidirectional data exchange. CoreBluetooth is used to manage low-latency, reliable transmissions.

- **External APIs and Services Layer:** Integrates third-party services like OpenRouteService (ORS) to obtain real-time walking routes and geospatial data. This enables the system to dynamically adapt navigation paths based on user position.
- **Hardware Integration:** The Raspberry Pi processes external sensor inputs and delivers haptic feedback based on instructions from the mobile app. This interaction is optimized for quick reaction times and obstacle response.

This modular, layered approach allowed each component to focus on a specific responsibility, making the system easier to test, modify, and expand without disrupting the overall architecture.

10.3 Technology Stack

Our project integrates a comprehensive and modern technology stack to deliver an effective and user-friendly navigation assistant. The core components include:

- LiDAR: Utilized for real-time depth sensing and obstacle detection. The LiDAR scanner enables accurate mapping of the surrounding environment, essential for safe navigation and spatial awareness.
- Swift: The primary programming language used to develop the iOS mobile application. Swift provides the necessary tools and frameworks to interact with device hardware, implement ARKit features, and ensure a smooth user experience.
- Geographical ARKit: Apple's Augmented Reality framework is employed to integrate geographical data and real-world coordinate tracking. This allows the system to guide users visually using virtual waypoints in their physical surroundings.
- Bluetooth: Enables wireless communication between the iOS mobile application and external hardware devices, such as the Raspberry Pi, for data exchange and control signals.
- Raspberry Pi: Serves as a compact computing module to process external sensor inputs and relay signals to the mobile application. It acts as a bridge between hardware components and the app.
- Mobile Application (iOS): The user-facing application that provides real-time navigation cues. It integrates LiDAR data, AR visualization, and Bluetooth communication to guide users seamlessly.
- SolidWorks: Used for designing and prototyping custom hardware mounts or enclosures. These designs support the physical integration of sensors and ensure ergonomic placement for practical use.

This diverse technology stack enables the creation of a robust, interactive system that merges hardware and software components to support intuitive and accessible navigation for users in real-world environments.

10.4 Key Features and Functionalities

The primary objective of our project is to develop an affordable, real-time navigation system specifically designed to assist visually impaired individuals in navigating their environments safely and independently. By leveraging the advanced LiDAR technology integrated into iPhone 12 Pro and newer models, the system provides precise spatial awareness and live guidance.

10.4.1 Key Features

- Real-Time Obstacle Detection and Path Guidance: The system continuously scans the environment using the iPhone's LiDAR sensor to detect obstacles and guide the user through safe paths. It provides immediate feedback through haptic cues to assist in real-time decision-making.
- Seamless Hardware and Software Integration: The mobile application communicates with external devices, such as the Raspberry Pi, via Bluetooth. This allows for synchronized input from additional sensors or controls, enhancing the system's reliability and functionality.
- Sidewalk-Based Mapping and Navigation: Unlike many commercial navigation systems that focus on roadways, our solution is optimized for sidewalks and pedestrian-friendly routes. This ensures safer and more practical navigation tailored to the needs of foot traffic.

10.4.2 Impact

By combining real-time obstacle detection, precise mapping, and accessible design, this system aims to enhance mobility, promote independence, and improve the overall quality of life for visually impaired users. It represents a step forward in making modern navigation technology more inclusive and adaptive to all users' needs.

10.5 Code Walkthrough

** [5] *Swift Programming Language*
Our mobile application is developed using Swift, Apple's powerful and intuitive programming language designed for iOS, macOS, watchOS, and tvOS development. Swift offers modern language features such as type safety, memory management, and expressive syntax, which make it an ideal choice for developing a robust and efficient mobile application. We referred to Apple's official Swift documentation throughout the development process to ensure best practices, maintain code reliability, and fully leverage the capabilities of the iOS development environment. Swift's seamless integration with Apple's frameworks, including ARKit and CoreBluetooth, enabled us to implement real-time obstacle detection, device connectivity, and voice-activated user controls efficiently. **

10.5.1 Overview

Our application is built using Swift and integrates ARKit, RealityKit, and CoreBluetooth to deliver a seamless navigation experience for visually impaired users. The following code walkthrough highlights the structure, purpose, and functionality of key files and components within the app.

10.5.2 App Entry Point: `ContentView.swift`

The application starts at `ContentView.swift`, which serves as the main interface for the app.

- **State Initialization:** The file initializes several important variables and state objects such as map rendering, Bluetooth communication, geolocation tracking, target coordinates, and user-selected destinations.

- **Permissions:** On launch, the app requests permissions for location services, microphone usage, and Bluetooth access. These are essential for the app's accessibility and AR capabilities.
- **User Interface:** `ContentView` acts as the front-end and conditionally renders elements based on permission states. It provides users with access to select destinations, initiate navigation, and manage Bluetooth devices.
- **Location and ARKit Integration:** The `ContentView` leverages `LocationManager` to obtain real-time geolocation data, which is vital for route generation and navigation. ARKit is used to manage depth sensing, ensuring that the app dynamically adapts to the user's environment.
- **Depth Sensing and LiDAR Controls:** The app allows users to control and interact with depth sensing features through a series of on-screen toggles and sliders. These controls allow the user to manage settings such as the maximum range and the minimum clearance distance, which influences the depth visualization and AR experience.
- **Route Calculation and Map Integration:** The app integrates OpenRouteService (ORS) API to fetch walking routes from the user's location to a selected destination. The route is displayed on the map, with resampled coordinates for better accuracy and smoother navigation.
- **Real-time Navigation:** As the user moves, the app constantly updates their position and compares it to the next target on the route. Once the user reaches a designated point, the app automatically advances to the next step in the navigation process.
- **AR and UI Interactivity:** Depending on whether LiDAR is available or not, the app can toggle between using LiDAR and feature point-based AR capabilities. This allows the user to switch between a more precise 3D depth sensing model (via LiDAR) and a less precise but still effective feature-point model.
- **Dynamic UI Updates:** The UI elements, such as the distance controls, update dynamically based on the user's interaction with the app. These elements are conditionally visible and can be customized, providing an adaptive user experience that accommodates different environments and user preferences.
- **Map and Target Updates:** The `ContentView` reacts to changes in user location by continuously updating the map and the route. The target index is updated when the user reaches a waypoint, ensuring smooth and uninterrupted navigation.

In summary, `ContentView.swift` is the central hub of the application, seamlessly integrating geolocation, AR features, and user controls to provide a comprehensive and interactive navigation experience. It ensures that users have a responsive, adaptable interface that enhances the usability of the app, while incorporating advanced features like real-time route calculation and AR depth sensing.

10.5.3 AR Integration: `ARWrapper.swift`

The `ARWrapper` acts as a bridge between the SwiftUI interface and the ARKit/RealityKit systems that drive our app's core navigation and obstacle avoidance.

- **AR Setup and GeoTracking:** The AR view is initialized with geo-tracking enabled, allowing the placement of virtual anchors based on GPS coordinates. This is essential for route mapping in outdoor environments.

- **Depth and Obstacle Detection:** Using LiDAR and ARKit's scene depth data, the app detects nearby obstacles. Taking into account the location of the obstacle, the program calculates the safest direction in which to proceed towards.
- **Scene Visualization:** Optional mesh overlays can be displayed, helping developers visualize how ARKit interprets the surrounding environment.
- **Real-Time Feedback:** The app overlays directional cues in AR (e.g., arrow indicators) and calculates path angles, which are then sent to the Raspberry Pi for haptic feedback via Bluetooth.
- **Memory Management:** To ensure optimal performance, the app clears old anchors and unused mesh data during runtime.

  Simplified Logic for Obstacle Avoidance:

```
if leftClear && !rightClear {
    instruction = "Go Left"
} else if rightClear && !leftClear {
    instruction = "Go Right"
} else if !leftClear && !rightClear {
    instruction = "Stop"
} else {
    instruction = "Proceed Forward"
}
```

  This is a simplified overview of the program, in actuality the pointer is capable of rotating at varied angles to guide the user precisely towards the safest path.

10.5.4 Bluetooth Communication with Raspberry Pi

- The app establishes a connection with a Raspberry Pi device via CoreBluetooth.
- Commands such as target angles and obstacle status are sent to the Raspberry Pi, which translates these into haptic feedback signals in the wearable glove devices.
- Swift code ensures consistent, low-latency data transmission, with retries built in for reliability.

10.5.5 `CompassView.swift` – Directional Feedback

The `CompassView` and `AvoidanceCompassView` SwiftUI components are responsible for visually communicating navigation instructions to the user through a dynamic compass-style arrow. These views provide real-time directional guidance based on the user's route and surrounding obstacles.

CompassView

- **Purpose:** Displays a rotating arrow that indicates the direction of the next waypoint
- **Implementation Details:**
  - Uses the SF Symbols `arrow.up` as the base image

  - Rotates the arrow to match the calculated direction angle using `.rotationEffect(.degrees(angle))`
  - Adds styling such as padding, background, a circular clip shape, and shadow for improved visibility

AvoidanceCompassView

- **Purpose:** Enhances `CompassView` by providing visual feedback on obstacle detection
- **Key Feature:** Dynamically changes background color to indicate the safety of the path:
  - **Red:** Path is blocked on the Right
  - **Blue:** Path is blocked on the Left
  - **Yellow:** Path forwards is blocked (e.g., turn left or right)
- **How it Works:**
  - Receives an angle and a Boolean `isPathClear`
  - Applies color logic based on those values
  - Uses the same visual style as `CompassView` for consistency

Example Use in App

```
AvoidanceCompassView(angle: navigationAngle, isPathClear: !obstacleDetected)
```

- `navigationAngle` is calculated based on the ARKit scene and user's current location
- `obstacleDetected` is derived from LiDAR depth analysis in `ARWrapper.swift`

These views ensure that users receive clear, intuitive direction guidance at a glance. This is especially useful for visually impaired users when paired with additional feedback mechanisms such as audio or haptics.

10.5.6 Location Services: `LocationManager.swift`

The `LocationManager.swift` file defines a custom class responsible for handling location and heading updates throughout the application. It leverages Apple's `CoreLocation` framework to provide accurate, real-time geographic data necessary for navigation and AR interactions.

- **Observable Class:** `LocationManager` conforms to `ObservableObject`, allowing SwiftUI views to automatically react to changes in location or heading data via the `@Published` properties `location` and `heading`.
- **Initialization and Configuration:** During initialization, the class sets itself as the delegate for `CLLocationManager`, requests location authorization, and starts updating both the user's location and compass heading. The desired accuracy is set to `kCLLocationAccuracyBestForNavigation` to ensure precision for navigational use cases.
- **Start/Stop Updates:** The class includes methods `startUpdating()` and `stopUpdating()` that allow external views or components to toggle location and heading updates on demand.

- **Location Updates:** The delegate method `didUpdateLocations` retrieves the most recent `CLLocation` object and assigns it to the `location` property on the main thread, ensuring UI responsiveness.
- **Heading Updates:** The `didUpdateHeading` delegate method updates the `heading` property with the device's latest compass heading, also dispatched to the main thread for smooth UI binding.
- **Authorization Handling:** The method `didChangeAuthorization` checks if location permissions were granted (`authorizedWhenInUse` or `authorizedAlways`) and resumes updates accordingly.
- **Error Handling:** If the location manager encounters an error, the `didFailWithError` delegate method logs it for debugging purposes.

Overall, `LocationManager.swift` abstracts complex location services into a reactive, SwiftUI-compatible class that efficiently powers navigation and AR features within the app. Its design ensures real-time updates and seamless integration with other components through observable bindings.

10.5.7 Map Interface: `MapView.swift`

The `MapView.swift` file implements a custom map interface using Apple's `MapKit` framework. It enables users to view their current location and select a destination via a long-press gesture on the map.

- **UIViewRepresentable Integration:** The `MapView` struct conforms to `UIViewRepresentable`, allowing integration of UIKit's `MKMapView` into the SwiftUI environment. This approach provides full control over map behavior while maintaining compatibility with the reactive SwiftUI paradigm.
- **Bindings and Observed Objects:** The view accepts three key parameters:
  - `@Binding var selectedCoordinate`: Holds the user-selected destination coordinate
  - `@ObservedObject var locationManager`: Tracks real-time user location
  - `@Binding var isPresented`: Manages modal or sheet presentation state
- **Initial Map Configuration:** In `makeUIView`, the map view is configured to show the user's location. It centers the map either on the user's current coordinates or a default fallback (San Francisco) if the location is unavailable.
- **Long-Press Gesture:** A `UILongPressGestureRecognizer` is added to the map, enabling users to drop a pin and select a destination. When triggered, it:
  - Converts the tap location to geographic coordinates
  - Clears existing annotations
  - Adds a new annotation at the selected point
  - Updates the `selectedCoordinate` binding
  - Dismisses the map view by setting `isPresented` to `false`
- **Coordinator Pattern:** The inner `Coordinator` class acts as the map view's delegate and handles interaction logic, such as responding to long-press gestures and managing annotations.
- **Dynamic Updates:** The `updateUIView` method continuously updates the map's region based on real-time changes to the user's location, ensuring the view remains centered during navigation.

In summary, `MapView.swift` provides a responsive and interactive map interface that connects user input with navigation logic. Its UIKit-SwiftUI bridging and gesture recognition features enhance the user experience while maintaining modular code structure.

10.5.8 Mini Map Display: `MiniMapView.swift`

The `MiniMapView.swift` file defines a compact, embedded map interface using UIKit's `MKMapView` via the `UIViewRepresentable` protocol. This view is designed to present the user's current location and a route preview without interactive navigation.

- **View Purpose:**
  - Displays a small, static map showing the user's location and route path.
  - Enhances spatial awareness by offering a contextual overview during navigation or activity tracking.
- **Input Parameters:**
  - `routeCoordinates`: A list of `CLLocationCoordinate2D` representing a navigation route
  - `userLocation`: The current location of the user, used for centering and bounding logic
- **Map Configuration:**
  - Interaction is disabled (`isZoomEnabled` and `isScrollEnabled` set to `false`) to create a read-only experience.
  - `showsUserLocation` ensures the user's position is always marked.
- **Dynamic Map Updates:**
  - In `updateUIView`, any previous overlays are cleared before redrawing the route.
  - If a route is available, a green polyline is drawn connecting all coordinates.
  - The map bounds are dynamically adjusted to include both the route and the user location, ensuring everything is visible with slight padding.
  - If no route is provided, the view centers around the user's location with a fixed zoom level.
- **Overlay Rendering:**
  - A nested `Coordinator` class implements `MKMapViewDelegate` to customize rendering behavior.
  - The route is rendered as a green line using `MKPolylineRenderer`, with a width of 3.0 points for clarity.

In summary, `MiniMapView.swift` provides a visually informative yet non-interactive map widget that serves as a route preview component. It balances simplicity and functionality, offering clear user feedback in navigation-related contexts.

10.5.9 Obstacle Avoidance Controller: `ObstacleAvoidance.swift`

The `ObstacleAvoidance.swift` file defines the `ObstacleAvoidanceController` class, which manages real-time obstacle detection and visual path generation in an AR environment using RealityKit and ARKit. It serves as a visual guidance mechanism by rendering navigational cues that dynamically adapt to the user's surroundings and position.

- **Core Functionality:**
  - Monitors mesh data in the AR scene to detect obstacles
  - Compares user location to a predefined route to determine deviation
  - Generates 3D visual guidance paths using `ModelEntity` that adapt color and shape based on detected obstacles or user deviation
- **Key Components:**
  - `rootEntity`: An `AnchorEntity` that anchors the obstacle-avoidance visuals in AR space
  - `pathEntity`: A `ModelEntity` used to visualize the path ahead
  - `mainRouteCoordinates`: Stores the intended navigation route as GPS coordinates
  - `meshAnchors`: Cached `ARMeshAnchor` data representing the physical environment captured by LiDAR or depth sensing
- **Update Mechanism:**
  - The `updateObstacleAvoidance` method is called once per frame.
  - It calculates the camera's forward vector and position, checks if the user is off the intended route, and determines if an obstacle is in front of the user.
  - Based on these conditions, it generates a colored and shaped path: red if an obstacle is present, blue if clear; box if on-route and clear, cylinder if off-route or obstructed.
- **Distance and Obstacle Detection:**
  - `isUserFarFromRoute`: Computes the closest distance from the user's current GPS location to the route coordinates and flags deviation if beyond 5 meters.
  - `isObstacleInFront`: Iterates through ARMeshAnchor vertices to check if any intersect with a predefined forward corridor in front of the user.
- **Path Geometry Generation:**
  - `generatePathMesh`: Constructs either a rectangular box or a cylinder to represent the navigation path, depending on whether the user is off-route or an obstacle is present.

`ObstacleAvoidanceController` plays a critical role in merging environmental awareness with spatial navigation, offering a dynamic, adaptive path visualization that enhances user safety and orientation in AR navigation experiences.

10.6 User Interaction and Accessibility Features

Our navigation system has been designed with a strong focus on accessibility to ensure it meets the needs of visually impaired individuals, utilizing both existing and custom-built accessibility features for ease of use.

10.6.1 Apple's Current Accessibility Features

Leveraging Apple's powerful accessibility tools, such as VoiceOver, users can interact with the app through voice feedback, ensuring an intuitive and inclusive experience. Additionally, features like magnification, screen contrast adjustments, and haptic feedback are integrated for greater usability.

Siri Integration: To simplify user interaction, Siri is integrated into the app, allowing users to easily open the application and set their desired destination via voice commands. This hands-free functionality enhances ease of use, especially in situations where physical interaction is not possible or convenient.

Device Pairing with Raspberry Pi: The system allows for seamless pairing of the iPhone with the Raspberry Pi via Bluetooth, enabling real-time communication between the mobile app and external hardware components. This ensures smooth interaction and data flow between the phone and the navigation system's sensors.

Fully Integrated Voice Controls: The app features full voice control integration, allowing users to navigate through different settings, adjust preferences, and receive real-time guidance without the need for visual input. This voice-first approach ensures greater autonomy and flexibility for users.

Night and Evening Functionality: Understanding the importance of accessibility during all times of day, our system is designed to work effectively at night and in low-light environments. The app utilizes enhanced depth sensing and LiDAR data to ensure that obstacle detection and guidance remain reliable, even in dimly lit conditions, ensuring continuous safety and usability.

10.7 Integration with Hardware and External Systems

Initially, we considered using an Arduino for the hardware integration of our navigation system. However, after evaluating its performance and considering the need for reliability, cost-effectiveness, and ease of development, we transitioned to using a Raspberry Pi. The Raspberry Pi offers several advantages over the Arduino, making it a better choice for our project.

10.7.1 Switch from Arduino to Raspberry Pi

The Arduino, while useful for basic applications, proved to be less reliable for our requirements, especially in terms of real-time obstacle detection and system responsiveness. The Raspberry Pi, being more powerful and cost-effective, provided a significant improvement in performance and flexibility. The increased processing speed of the Raspberry Pi allows for quicker data processing and more efficient obstacle avoidance algorithms.

10.7.2 Bluetooth Integration

For seamless communication between the Raspberry Pi and the iPhone, we implemented a Bluetooth connection using a specialized library. This library enables the Raspberry Pi to identify itself as a connectable Bluetooth device, making it easy to pair with the mobile application. This Bluetooth setup allows for real-time transmission of sensor data and navigation instructions.

10.7.3 Efficient Obstacle Avoidance

To ensure effective and efficient obstacle avoidance, we used an additional library on the Raspberry Pi that allows us to receive multiple sensor transmissions in rapid succession. This feature is crucial for real-time adjustments as the system identifies and avoids obstacles, providing a smooth and continuous navigation experience.

10.7.4 Why Raspberry Pi Over Arduino

Ultimately, the Raspberry Pi was chosen over the Arduino due to its faster processing capabilities, ease of use, and better support for testing and debugging. The Raspberry Pi is also more adaptable to new challenges, offering greater flexibility as the project evolves. Its robust ecosystem of libraries and community support further reinforced our decision to use it as the core hardware component of the system.

10.8 Challenges Encountered and Solutions Implemented

Throughout the development of our navigation system, we faced several technical challenges. One of the toughest challenges was integrating obstacle avoidance with navigation, as Apple provides two separate libraries for these functionalities. These libraries cannot be used simultaneously, which was the main problem for us.

10.8.1 Obstacle Avoidance vs. Navigation

The challenge arose because we had to choose between two critical features: one library offered better obstacle avoidance, while the other provided better navigation capabilities. Given the importance of providing accurate and reliable navigation for visually impaired users, we determined that navigation was the more crucial feature for our application.

10.8.2 Leveraging LiDAR Data

With the focus on navigation, it became essential to find a way to effectively harness the raw LiDAR data provided by the iPhone. We needed to process this data in a way that allowed us to offer smooth, real-time navigation without compromising on the system's overall responsiveness. Our solution involved experimenting with various data processing methods to extract the most useful information for the navigation algorithm while ensuring the system could still handle obstacles effectively.

10.8.3 Custom Solution Development

To address this challenge, we implemented custom software that processed the LiDAR data in real time, enabling the system to identify obstacles and navigate safely. While the obstacle avoidance feature was not

as advanced as initially planned, our solution allowed us to strike a balance between the two features and prioritize navigation, which was our primary goal.

In summary, the most significant challenge we encountered was determining how to integrate the competing libraries for navigation and obstacle avoidance. By focusing on the raw LiDAR data and processing it efficiently, we were able to implement a navigation system that met the needs of visually impaired users while still considering obstacle detection.

10.9 Future Improvements and Features

While our current system provides a functional and responsive navigation tool for visually impaired users, we envision several enhancements to increase its effectiveness and usability:

- **Offline Navigation Support:** Currently, our system relies on real-time API calls for route guidance. Implementing offline caching of common routes and maps would improve reliability in areas with limited connectivity.
- **Expanded Sensor Fusion:** Incorporating additional sensors such as ultrasonic range finders or infrared sensors could improve obstacle detection in conditions where LiDAR performance is limited (e.g., glass or reflective surfaces).
- **Customizable Feedback Profiles:** Future iterations could allow users to adjust the intensity, frequency, or type of haptic feedback according to their preferences or needs.
- **Emergency Location Sharing:** To improve safety and peace of mind, we plan to implement a feature that automatically shares the user's live location with pre-selected emergency contacts if the app detects a failure or if the user issues a distress signal.
- **Advanced AI-Based Path Prediction:** Integrating machine learning to predict user walking patterns or avoid frequently encountered obstacles could improve the overall adaptability and safety of the system.
- **Multi-Language Voice Commands:** Expanding voice control to support multiple languages and dialects would enhance accessibility for non-English speakers.

These proposed improvements represent the next step in transforming the system from a functional prototype into a fully scalable, market-ready product.

10.10 Conclusion

This chapter has outlined the software design and implementation of our assistive navigation system, highlighting the seamless integration between software and hardware components. By leveraging Apple's LiDAR technology, ARKit, and the computational power of the Raspberry Pi, we created a mobile application that offers real-time guidance and spatial awareness to visually impaired users.

From architectural planning to practical code implementation, we prioritized accessibility, reliability, and user-centered design. Despite technical limitations, especially in balancing navigation with obstacle avoidance, our team devised effective custom solutions that meet the core goals of the project.

Ultimately, this system reflects our commitment to inclusive technology, offering a blueprint for future innovations in assistive navigation tools. The foundation laid by this project positions it for continued refinement and broader deployment in real-world environments.

Chapter 11

Hardware Device Implementation

– Neeti Mistry

11.1 Introduction

The hardware implementation of the C-ALL system is a critical component in ensuring seamless real-time navigation and obstacle detection for visually impaired users. This chapter outlines the design, development, and integration of the hardware device with the mobile application, highlighting its role in delivering haptic feedback to assist users in navigating safely.

The primary purpose of integrating hardware with the C-ALL system is to create a wearable, responsive, and intuitive assistive device that enhances spatial awareness without relying solely on audio cues. The hardware component, equipped with haptic feedback motors, works in conjunction with the LiDAR-powered mobile application, enabling users to detect obstacles, receive directional cues, and navigate their environment more independently.

This chapter will detail the hardware components, assembly process, and software-hardware communication that form the foundation of the C-ALL system. Additionally, it will explore the challenges encountered, solutions implemented, and future enhancements planned for optimizing the device's usability and performance.

11.2 Hardware Design and Components

11.2.1 List of Components

To build our hardware device, we utilized the following components:

Hardware:

- iPhone 12 Pro (for testing)
- Battery (to power the Raspberry Pi)
- Triple Axis Compass Magnetometer Sensor Module 3.3V 5V for Arduino and Raspberry Pi (for directional input)

- Raspberry Pi (processing unit)
- Limit Switches (to prevent over-rotation of the servo motor)
- Breadboard Kit (for prototyping circuit connections)
- Wires (to connect components)
- Prototype Materials (glove for potential wearable design)
- Servo Motors (to control pointer movement)
- 3D printed Solidworks model (casing and structure for hardware components)

11.2.2 Design Considerations

When designing the hardware component of our system, we aimed to provide a sensory-based navigation method that does not rely on visual feedback. Given our use case, traditional visual or auditory directions were not ideal, leading us to explore tactile and mechanical feedback solutions. Our design needed to be:

- Compact and Wearable – The hardware should be lightweight and small enough for practical use
- Accurate and Responsive – The movement of the pointer should reliably reflect directional guidance from the software
- Durable and Secure – The materials should ensure that the components remain intact during movement
- Seamlessly Integrated – The hardware must communicate effectively with the software to provide real-time directional feedback

Taking these considerations into account, we developed a 3D-printed prototype with a servo motor-based pointer system to physically indicate directional changes.

11.2.3 SolidWorks 3D Design and Prototyping

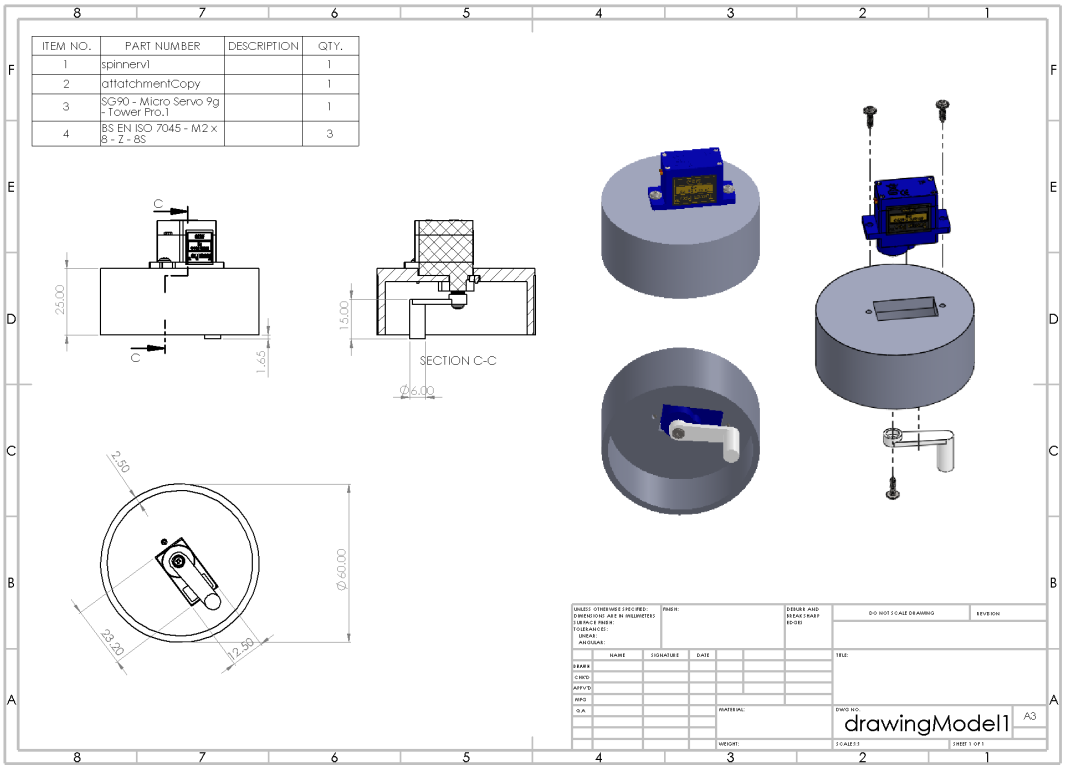

Figure 11.1: Solidworks Model Drawing First Iteration

To create a functional and efficient design, we used SolidWorks to model the hardware casing and pointer mechanism. The 3D model was designed to house the servo motor and allow for a 360-degree rotating pointer that corresponds to the navigation directions provided by the software application.

After finalizing the model, we 3D printed the casing and assembled the components inside it. The servo motor was mounted on top, with a rotating arm inside the casing to act as the pointer. The compact design resulted in a device about the size of a palm, making it portable and easy to integrate into a wearable system.

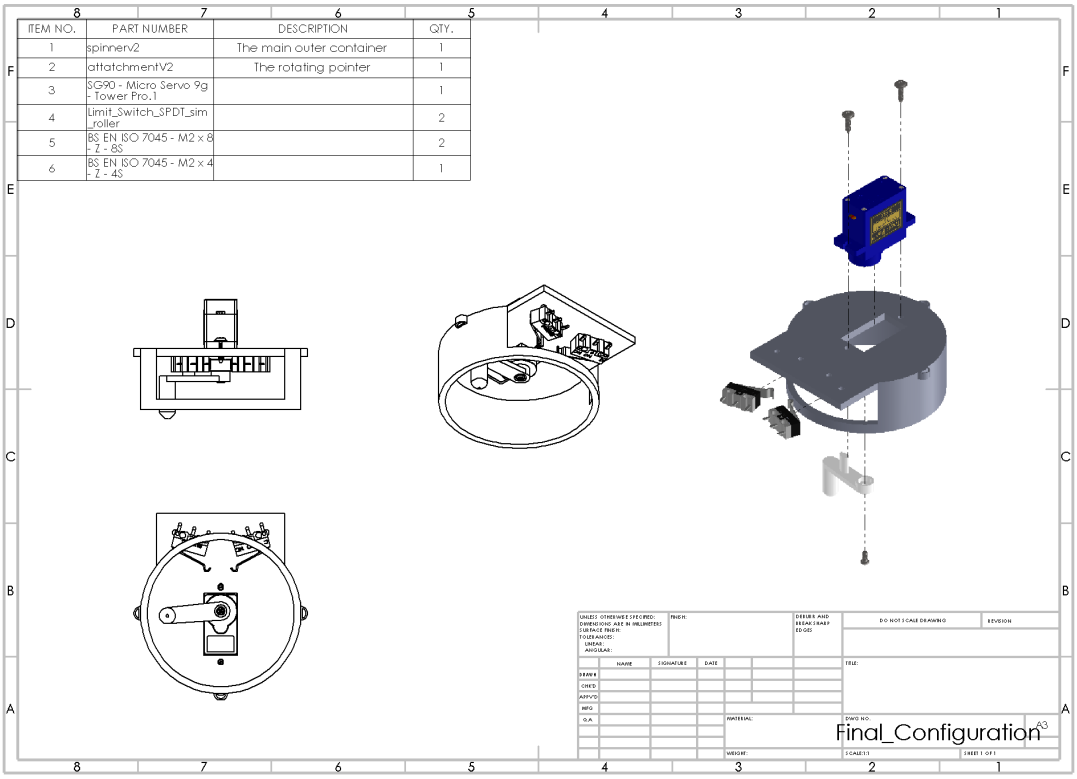

Figure 11.2: SolidWorks Model Drawing Final Iteration

The updated SolidWorks design reflects two key improvements from the initial iteration:

1. Limit Switch Compartment: We added a dedicated compartment to house the limit switches securely. This structural addition ensures better alignment and protection for the switches, improving both the reliability and maintainability of the system.

2. Refined Pointer Design: The pointer has been reshaped to feature a sharper, more defined tip. This change enhances directional clarity during navigation and makes the device's feedback cues more intuitive for the user.

These revisions were guided by testing insights and user feedback, helping us move closer to a functional final design.

11.3 Assembly and Hardware Integration

The hardware assembly involved securing the servo motor inside the 3D-printed casing using screws. The rotating arm (pointer) was attached to the motor, which moves based on directional input from the software.

The Raspberry Pi controls the servo motor, receiving movement commands and translating them into real-time directional changes.

We ensured:

- Proper alignment of the servo motor within the casing
- Stable attachment of the pointer to prevent misalignment
- Secure wiring to connect the Raspberry Pi, motor, and sensors

11.4 Hardware-Software Interaction

The hardware interacts with the software through a Raspberry Pi-controlled servo motor. The app determines the desired direction and sends corresponding commands to the Raspberry Pi, which then rotates the pointer to indicate the next movement.

- The Triple Axis Compass Magnetometer provides real-time orientation data
- The software calculates the correct pointer position based on navigation algorithms
- The servo motor adjusts the pointer's direction accordingly

This seamless interaction allows users to receive tactile feedback rather than relying on a visual interface, aligning with the project's accessibility goals.

11.5 Challenges and Solutions

Throughout the hardware implementation process, we encountered some challenges:

1. Pointer Stability and Accuracy
    - Challenge: The pointer sometimes overshot or undershot its intended position
    - Solution: We fine-tuned the servo motor's angle calculations and added limit switches to restrict movement
2. Hardware Size Constraints
    - Challenge: The original model was too bulky for practical use
    - Solution: We optimized the SolidWorks design to reduce material use and improve compactness
3. Hardware-Software Sync Issues
    - Challenge: Delays in communication between the Raspberry Pi and servo motor caused lag
    - Solution: We optimized the data transmission process, reducing latency between software commands and hardware response

11.6 Conclusion

The hardware implementation phase successfully resulted in a functional directional feedback device that provides non-visual navigation assistance. Using a 3D-printed casing, a servo motor, and Raspberry Pi, we developed a compact and responsive system that integrates seamlessly with our software. Future iterations may include miniaturization, enhanced durability, and improved response times to further refine the user experience.

Chapter 12

Usability Testing

– Neeti Mistry

12.1 Introduction

The chapter focuses on evaluating the effectiveness and user-friendliness of our prototype by gathering feedback from individuals who are representative of our target audience. In this case, our primary users are visually impaired individuals who rely on assistive technologies to navigate their environments. By conducting usability testing, we aim to identify any challenges, obstacles, or areas of improvement in both the hardware and software aspects of our prototype.

Usability testing is crucial to ensure that our device meets the practical needs of the visually impaired community, providing a seamless and efficient solution for safe and independent mobility. In this chapter, we will discuss the methodologies used in the testing process, the type of feedback gathered, and the resulting improvements made to the product. The goal is to enhance the user experience by ensuring that the device is intuitive, accessible, and capable of addressing real-world challenges faced by our users.

Through Iterative Testing and user-centered design, we strive to refine our product to ensure that it delivers maximum benefit to those who need it the most.

12.2 Testing Methodology

To ensure that our prototype effectively meets the needs of visually impaired individuals, we conducted a series of usability tests aimed at evaluating both the hardware and software components of our system. Our approach focused on real-world testing scenarios, iterative feedback collection, and direct engagement with our target users.

12.2.1 User-Centered Testing Approach

Given the critical role of accessibility in our project, we adopted a user-centered testing methodology. This involved gathering feedback from individuals who rely on assistive technologies in their daily lives, allowing us to assess how well our prototype aligns with their needs. Our primary goal was to identify usability challenges, accessibility barriers, and areas for improvement in both the design and functionality of the system.

12.2.2 Test Participants

Our usability testing involved visually impaired individuals with varying levels of experience using assistive navigation tools such as canes, guide dogs, and mobile accessibility applications. We conducted a detailed use case interview with Jules Jaworowski, a visually impaired student majoring in Vision and Special Education. Her insights helped shape our testing process early on by highlighting real-world navigation challenges and accessibility considerations.

12.2.3 Testing Environment and Scenarios

To evaluate the prototype in realistic conditions, we conducted usability tests in different environments:

- Outdoor Spaces: Assessing the prototype's effectiveness in open areas, sidewalks, and crowded public spaces
- Unfamiliar Routes: Observing how users adapt to the device when navigating new locations

Each test scenario aimed to simulate real-world use cases, focusing on obstacle detection, route guidance, and user comfort.

12.2.4 Data Collection Methods

To ensure comprehensive evaluation, we used multiple data collection methods:

- Observation: Watching users interact with the prototype and noting any difficulties they encountered
- Think-Aloud Protocol: Encouraging users to verbalize their thoughts and experiences while using the system
- Questionnaires and Interviews: Gathering structured feedback on the device's usability, accessibility, and effectiveness
- Performance Metrics: Measuring key factors such as reaction time, navigation accuracy, and ease of use

12.2.5 Iterative Testing and Refinement

Usability testing was conducted in multiple phases to allow for iterative refinement:

- Initial Prototype Testing: Early tests to identify fundamental usability issues and software bugs
- User Feedback Integration: Adjustments made based on feedback from visually impaired users
- Final Usability Evaluation: Comprehensive testing to validate improvements before deployment

Through this iterative process, we continuously improved our system to better address user needs, ensuring a more intuitive and reliable navigation aid.

12.3 Use Case Interview - Initial Feedback

As part of our usability testing process, we conducted an in-depth initial use case interview with Jules Jaworowski, a visually impaired individual who is pursuing a dual major in Vision and Special Education at Kutztown University. Jules provided valuable insights into her daily navigation challenges, which informed several improvements to our project very early on in the development stage. Her feedback allowed us to shift our focus to think from the perspective of our key users right from the beginning.

As developers, it's easy to get caught up in creating something highly technical, pushing the limits of our skills, and prioritizing complex features. However, this experience highlighted the importance of User-Centered Design, especially when developing technologies that directly impact the daily lives of individuals. We are grateful that we were able to interview Jules early in the process, as it helped drive our focus throughout the entire project. By incorporating her insights, we ensured that our efforts aligned with the real-world needs of visually impaired users, making their experiences safer, easier, and more empowering.

Friday, September 20, 2024

12.3.1 Background Information

- Name: Jules Jaworowski
- School: Kutztown University, PA
- Major: Dual Major in Vision and Special Education
- Age: 20
- Gender: Female

12.3.2 Key Insights and Feedback

Comparison of Seeing Eye Dog vs. Cane
Jules highlighted significant differences between her seeing eye dog, Ruby, and a cane. She explained, "A cane is great, but a dog has 360-degree awareness and can track objects at face level. A cane only tracks 2-3 feet in front of you and the ground." Jules noted that while a cane is limited to detecting obstacles directly in front, Ruby provides more comprehensive situational awareness by detecting objects around her and even preventing collisions with obstacles. This feedback underscored the importance of improving our prototype's ability to detect obstacles in the user's environment and provide more intuitive guidance, similar to the full surroundings awareness Ruby offers.

Routine Navigation in Familiar and Unfamiliar Environments
Jules shared her routine for navigating familiar and unfamiliar spaces. "Ruby picks up a familiar route really fast, but when it's unfamiliar, we stop more often for better navigation." This insight suggested the need for our prototype to include more adaptive and flexible obstacle detection and navigation features to handle both familiar and new routes effectively. It also indicated that dynamic, exploratory navigation should be considered to enhance the user's experience when facing unfamiliar paths.

Training the Dog on New Routes
Jules mentioned that it typically takes 2-3 trips for Ruby to fully acclimate to a new route. This insight

emphasized the importance of making our prototype adaptable to new environments and ensuring that it can learn and improve over time to provide consistent support during navigation.

Use of Other Navigation Aids
Jules described using a sighted guide in crowded areas where Ruby's ability to navigate may be hindered. She explained, "In large crowds, Ruby's ability to guide sometimes diminishes, so I may switch to a cane or ask for help." This feedback highlighted the necessity of integrating a system that works alongside existing navigation aids, such as a cane or guide, to provide additional support when other methods fail.

Technological Tools and Apps
Jules currently uses several technological tools, including Apple's built-in zoom and VoiceOver features, and a free app called Seeing AI that helps identify colors and read text aloud. However, she encountered significant challenges with some accessibility features, such as with the Pearson platform (for school), which she found to be poorly accessible. She also mentioned that PDFs do not work well with VoiceOver. This insight pointed to the importance of ensuring that our navigation aid is fully accessible, compatible with Apple's accessibility features.

Challenges in Public Spaces
When navigating public spaces, Jules mentioned the challenge of relying on others for assistance. "When I ask people for directions, they often freeze up or point in a direction that I can't see." She also noted difficulties reading Braille signs and room labels that are outdated or incorrect. This feedback emphasized the need for our prototype to provide clear and actionable guidance without requiring dependence on others, especially in public spaces where help may not always be available.

Desired Features in a Navigation Aid
Jules outlined several important features she would find valuable in a navigation aid for visually impaired individuals. She emphasized the need for the device to be "descriptive but not overwhelmingly descriptive," focusing only on major obstacles or hazards that could pose a risk. Jules explained that providing information about relevant, significant features, such as potential hazards in the environment, would be most beneficial without overwhelming the user with unnecessary details.

Additionally, Jules highlighted the challenge that many visually impaired individuals face when using mobility aids, noting that some feel embarrassed to use canes or other assistive devices. "Having another option would be game-changing. It would remove the shame factor and provide more choices." This insight confirmed the potential value of our navigation aid, offering users a more discreet and customizable solution.

Feedback on Technology Advancements
Jules expressed a strong interest in advancements in assistive technology, especially if it could provide a non-invasive, efficient alternative to traditional aids like canes or dogs. She noted that many visually impaired students feel ashamed of using mobility aids and would be willing to adopt new technologies as long as they could personalize and customize the experience. This feedback reinforced the importance of designing a solution that empowers users and allows them to feel in control of their navigation experience.

12.3.3 Impact on Prototype Development

The insights gathered from Jules' interview were invaluable in shaping the next steps for our prototype development. We plan to refine the device's obstacle detection capabilities to offer more intuitive and adaptive navigation, particularly in unfamiliar environments. We will also focus on improving accessibility features, and ensuring that our device provides clear, concise guidance without overwhelming the user with unnecessary information. Lastly, the feedback on personalization and choice will guide our efforts to make the navigation aid a customizable solution that empowers visually impaired individuals to navigate independently and confidently.

12.4 Further Interviews and Usability Testing

While we encountered challenges in deploying the app through TestFlight due to time constraints and the delayed permissions from Stevens, we were still able to gather valuable feedback from Jules Jaworowski and other visually impaired users. Unfortunately, we were unable to meet with participants in-person to conduct hands-on usability testing, but we continued to engage with them through remote interviews, where we presented the product and asked key questions to gather their thoughts and experiences.

After our initial interview with Jules, we reached out to her again for follow-up conversations as the development of the prototype progressed. During these interviews, we were able to present updates on the product, ask more in-depth questions, and solicit feedback on specific features we had been refining. Jules provided further insights into how the product could be made more adaptive and responsive to the specific needs of visually impaired users in varying environments.

Some of the key topics covered in these follow-up interviews included:

- Response to Updated Obstacle Detection: Jules shared how our improvements in detecting nearby obstacles were beneficial, but she emphasized the importance of clear and non-overwhelming feedback when multiple obstacles are present.
- Adaptability in Different Environments: Jules noted that while our product worked well in familiar spaces, she suggested adding features to dynamically adjust based on the complexity of new or unfamiliar routes.
- Accessibility of Interface: Jules emphasized that we should make the app as accessible as possible. She recommended ensuring that all voice-guided prompts were clear and easy to understand, while also making sure the app's design catered to users with varying levels of technological familiarity.

These follow-up discussions helped solidify our understanding of the real-world challenges that visually impaired individuals face when using navigation aids and deepened our commitment to refining the usability and accessibility of the system.

Despite the challenges, our continued interaction with Jules and other visually impaired users ensured that our product was refined in line with their needs, and we are optimistic about the future iterations that will come from ongoing feedback and improvements.

12.5 TestFlight

To support external testing and broader accessibility, we planned to deploy the C-ALL mobile application through TestFlight, Apple's official beta testing platform. TestFlight provides a secure environment for distributing apps to testers without requiring full App Store publishing permissions, enabling real-time feedback and iterative development.

12.5.1 Certificate Management and App Access

Although we were not granted the necessary permissions in time to fully use TestFlight for external testing, we were able to generate a development certificate for the C-ALL app. This certificate linked the app to a specific Apple Developer account, allowing installation on authorized devices for internal use. It also enabled team members to run the app on their own computers for localized testing, ensuring some consistency in early-stage feedback.

12.5.2 Testing and Feedback Integration

If access had been granted, TestFlight would have allowed us to distribute the app to a controlled group of internal testers. Testers would have evaluated usability, accessibility, performance, and real-world functionality across different environments. TestFlight's built-in features—such as recording tester activity and collecting usage analytics—would have provided critical insights to refine navigation input, pointer responsiveness, and overall interface clarity.

12.5.3 Future Automation with CI/CD Pipeline

We also prepared for future integration of a CI/CD (Continuous Integration and Continuous Deployment) pipeline alongside TestFlight. This system would automate app builds, testing, and deployment, accelerating updates, streamlining bug fixes, and supporting scalable development for future iterations of the C-ALL system. While limitations in access restricted full use of TestFlight during this phase, the work done laid the foundation for a more structured and scalable testing process in future development cycles.

12.6 Conclusion

Usability testing has been an integral part of ensuring that our navigation aid effectively meets the needs of visually impaired users. By engaging directly with individuals like Jules Jaworowski, we gained invaluable insights into real-world navigation challenges and accessibility concerns. These insights allowed us to refine our approach, focusing on user-centered design and iterative improvements.

Through structured testing methodologies, including observations, interviews, and performance evaluations, we identified key areas where our prototype required enhancements. The feedback led to significant improvements in obstacle detection, adaptive navigation, and accessibility features, ensuring that the device provides a seamless and intuitive experience for users.

As we move forward, our commitment to refining the prototype remains strong. By continuously engaging with users, implementing iterative testing cycles, and incorporating feedback-driven refinements, we aim to create a navigation aid that is both effective and empowering. The ultimate goal is to enhance

independent mobility for visually impaired individuals, offering a reliable and customizable solution that integrates seamlessly into their daily lives.

By prioritizing accessibility and user experience, we hope to contribute to a future where assistive technologies offer greater freedom, confidence, and independence to those who rely on them the most.

Chapter 13

Conclusion

– Neeti Mistry

13.1 Introduction

As we conclude our Senior Design journey, we reflect on the progress made from our initial idea to a fully functional prototype of our assistive navigation system for the visually impaired. This project challenged us to apply both our technical knowledge and our ability to empathize with end users in order to design a meaningful solution. It allowed us to bridge the gap between concept and execution, while reinforcing the importance of accessibility, user-centered design, and collaboration.

13.2 Original Project Concept vs. Delivered Product

Our original project concept aimed to develop a wearable navigation system designed specifically to assist visually impaired individuals in detecting and avoiding obstacles in real time. The initial vision included a much larger 3D-printed casing mounted on a single glove, which would house the hardware components.

Initially, we planned to use an Arduino microcontroller for data processing and feedback control. However, as the project progressed, we transitioned to a Raspberry Pi due to its superior processing power, built-in Bluetooth support, and ability to run Python-based scripts for real-time data handling and communication with the mobile app.

In the final stages of the project, we expanded the hardware system to include two gloves: one dedicated to navigation and another focused on obstacle avoidance. Since the Swift code had already been designed to handle both obstacle detection and navigation functionality, adapting the system to support two gloves was a manageable adjustment late in the project timeline.

The final product preserved the core goals of the original vision while adapting to the project constraints and user feedback. We successfully integrated LiDAR technology from iPhone Pro models into our wearable system, providing directional haptic feedback through the gloves. Rather than using real-time audio output, we shifted to fully haptic feedback based on usability concerns and the potential for sensory overload. A mobile application was developed to allow users to configure basic accesibility settings and communicate with the glove hardware via Bluetooth, although its functionality was more limited than originally envisioned due to time constraints.

Ultimately, the delivered product stayed true to the mission: creating a functional, accessible, and user-centered mobility aid for the visually impaired. The design changes made throughout the process reflect thoughtful prioritization, user feedback, and a commitment to delivering a polished and usable final system.

13.3 Final System Design

The final design of the C-ALL system consists of two wearable gloves and a mobile application. One glove focuses on real-time obstacle avoidance, while the second glove provides directional guidance for navigation. Together, they offer a comprehensive mobility solution for visually impaired users.

The wearable gloves house compact hardware modules built around Raspberry Pi boards. These modules process data received either from the mobile application or onboard sensors. The navigation glove interprets destination input from the mobile app and uses moving mechanical pointers to guide the user toward their selected location. The obstacle avoidance glove receives real-time LiDAR data from the user's iPhone and uses a similar pointer system to indicate nearby obstacles, providing intuitive and immediate environmental feedback.

The mobile application, developed in Swift, connects to the gloves via Bluetooth. It allows users to input destinations, customize feedback settings, and initiate navigation sessions. Although initial plans included broader features, such as caretaker monitoring and obstacle classification, the final app version prioritizes core usability and stability for individual users.

Communication between the app and gloves was designed with scalability in mind, incorporating a foundation for a CI/CD pipeline and potential TestFlight deployment. While full TestFlight testing was not possible due to permission and access challenges, the system is prepared for streamlined future updates and wider distribution.

Overall, the final C-ALL system emphasizes real-time performance, user comfort, intuitive feedback, and affordability—staying aligned with the original mission while adapting intelligently to technical, time, and resource constraints.

13.4 High-Level Illustration of the Final Product

Figure 13.1: The C-ALL system in use. The user wears both gloves while the iPhone app processes LiDAR data to guide navigation and detect obstacles in real time.

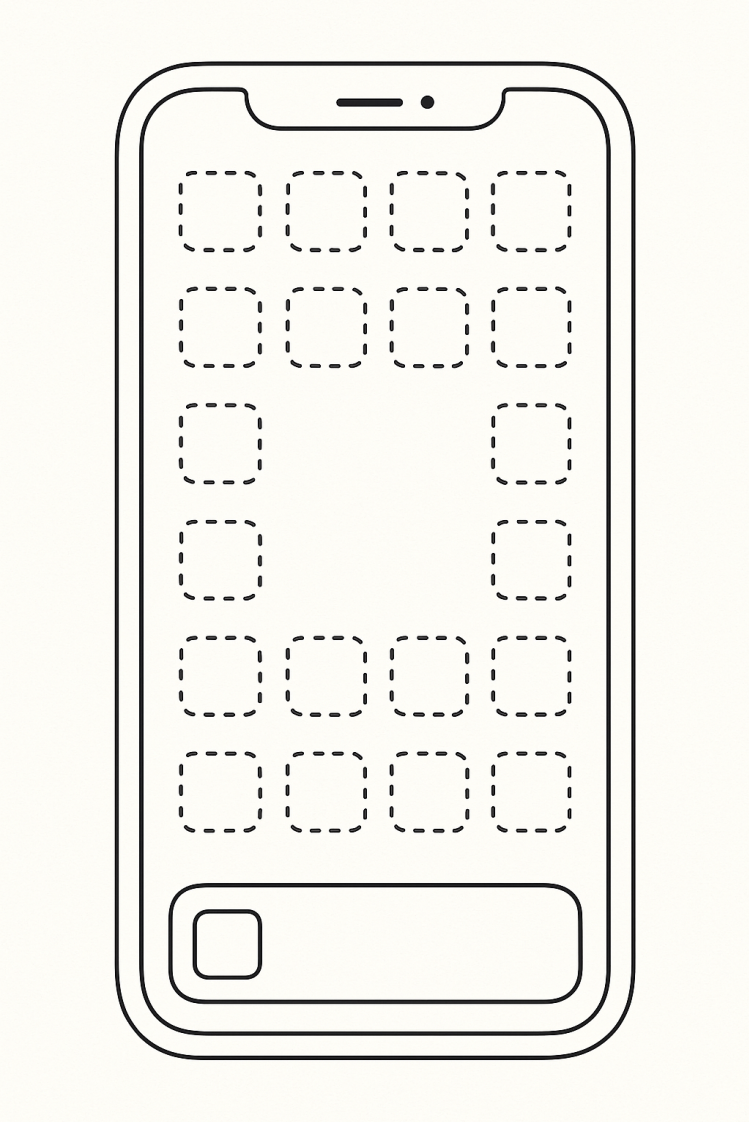

Figure 13.2: Locate the C-ALL app on your iPhone home screen after installation

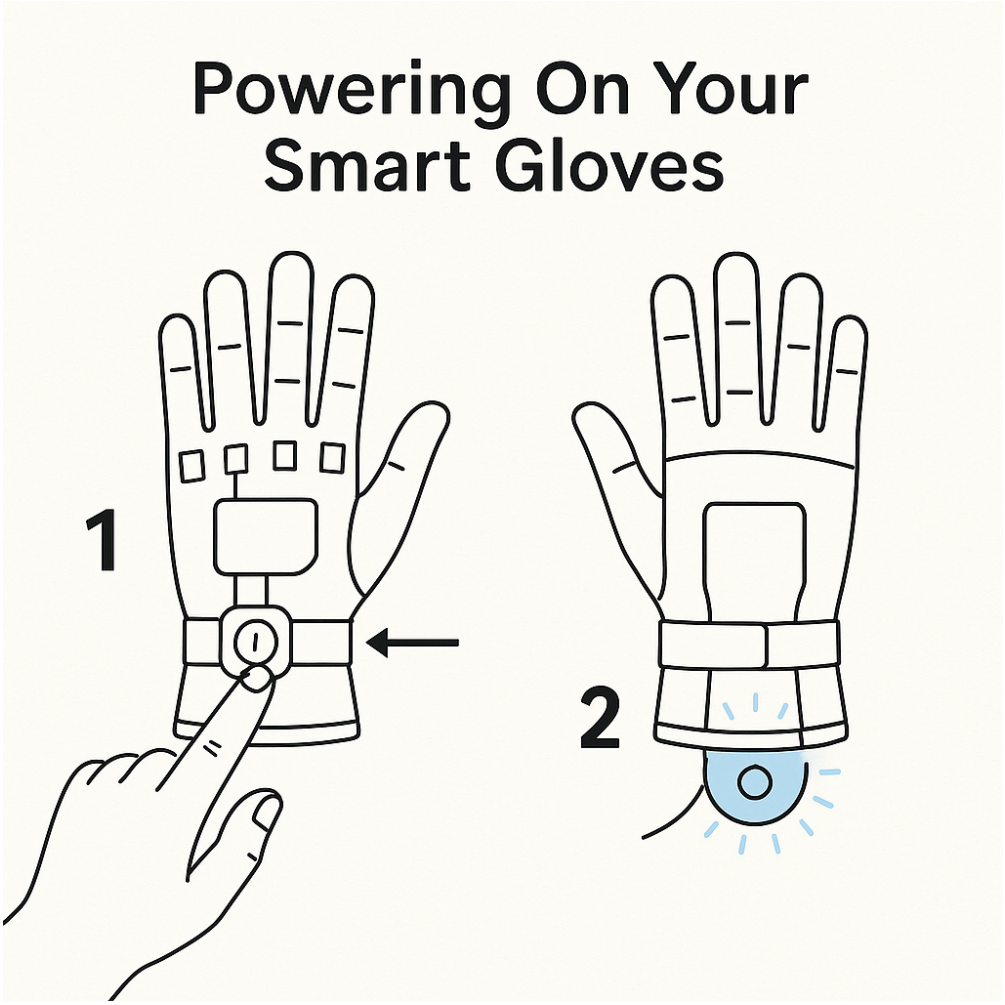

Figure 13.3: Powering on the smart gloves

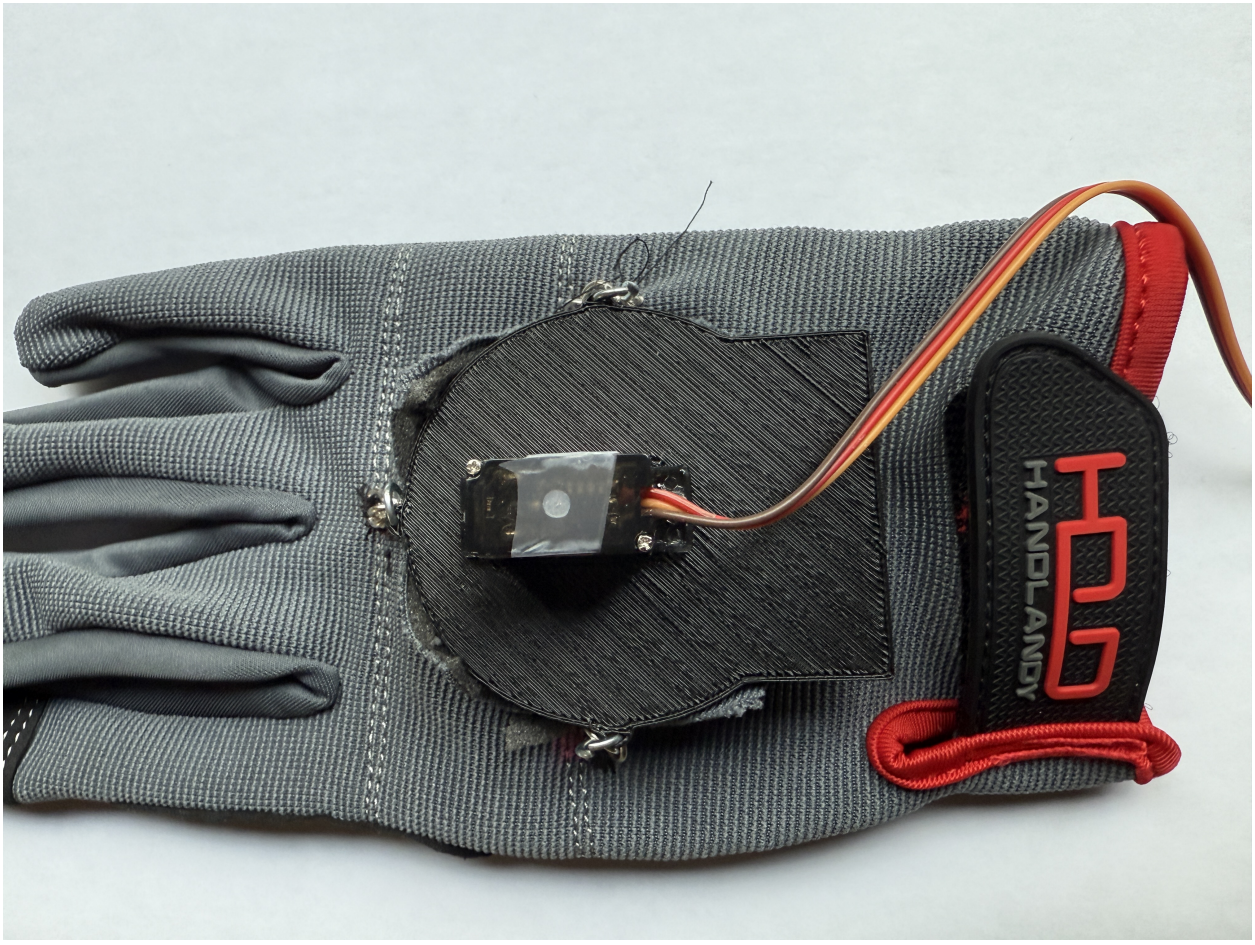

Figure 13.4: Final glove design with integrated servo motor and 3D-printed enclosure for directional haptic feedback.

The final gloves (there are 2 gloves) prototype features a durable work glove base with a centrally mounted servo motor housed in a custom 3D-printed enclosure. This motor controls a rotating pointer used to convey directional cues to the user. The glove is wired to connect with the Raspberry Pi, allowing it to receive real-time navigation instructions from the mobile app. The design balances comfort, robustness, and precise mechanical feedback, making it well-suited for daily assistive use.

13.5 Challenges and Obstacles

Throughout the development of our assistive navigation system, we faced a variety of challenges that tested our technical skills, resourcefulness, and team coordination.

Project Approval and Feasibility:
One of the earliest challenges was securing approval for our project concept. Given the ambitious nature of the system — combining wearable hardware with real-time LiDAR processing and a custom mobile application — there were concerns about feasibility and scope. We engaged in multiple discussions with faculty

and among ourselves to assess whether the system could be realistically completed within the academic year and in time for the Stevens Innovation Expo.

Lack of Existing Solutions to Reference:
Another hurdle was the absence of any directly comparable products on the market. There were no readily available hardware devices or mobile applications offering similar real-time obstacle detection and feedback for visually impaired users. As a result, we had to build our system from the ground up, including both the hardware and software components. This required extensive research, prototyping, and a flexible design process.

Access to Development Resources:
Because our system relied on iOS-specific features — particularly the LiDAR capabilities available only on newer iPhones models — development required access to MacOS. However, not all team members owned Mac devices, which created a bottleneck during app development and testing. We had to coordinate closely to share access and divide responsibilities in a way that maximized our available resources.

System Integration:
The most significant technical challenge we encountered was integrating all the components into a cohesive, functioning system. While we had previous experience with hardware programming and mobile development individually, combining them at this scale introduced many unexpected issues. Communication between the Raspberry Pi and iPhone, Bluetooth reliability, LiDAR data processing, and real-time obstacle avoidance all required considerable trial and error. Debugging across platforms and ensuring consistent performance pushed us to improve our troubleshooting and system design skills.

Despite these obstacles, our team remained resilient and adaptable, ultimately delivering a functional, innovative product that aligned with our initial goals.

13.6 Project Post-Mortem

The Senior Design project was a valuable learning experience that tested our technical skills, teamwork, and problem-solving abilities. Throughout the project, we encountered various challenges but also made significant progress in developing a new assistive navigation system for visually impaired individuals. This section reflects on what we did correctly, areas for improvement, and whether we would choose to pursue this project again with the knowledge we have now.

What We Did Correctly

1. Clear Vision and User-Centered Design: From the beginning, we maintained a strong focus on user needs by conducting interviews and gathering feedback from visually impaired individuals. This ensured our solution was designed with real-world usability in mind.
2. Effective Team Collaboration and Communication: Our team successfully divided tasks based on strengths, allowing us to work efficiently while keeping each other updated through regular meetings. Utilizing project management tools like Jira helped us stay organized and track progress.
3. Iterative Development and Testing: We followed an iterative approach, refining our software and hardware through multiple prototype cycles. Early testing allowed us to identify and fix issues before they became major roadblocks.

4. Integration of LiDAR Technology and Haptic Feedback: Successfully implementing LiDAR for obstacle detection and haptic feedback for user navigation was a major technical achievement. These features significantly enhanced the usability and effectiveness of our system.

What We Would Do Differently

1. Start Hardware Development Earlier: We initially focused more on the software side of the project, leading to delays in hardware integration. Starting hardware prototyping earlier would have given us more time for refinement and testing.

2. Improve Testing and Validation Process: Although we conducted some usability testing, we could have scheduled more structured testing sessions with visually impaired users earlier in the project. More real-world testing would have provided deeper insights into potential design improvements.

3. Better Contingency Planning for External Dependencies: We encountered delays due to reliance on hardware components and institutional approvals (e.g., TestFlight access issues). Anticipating these bottlenecks and having backup plans would have mitigated some of these delays.

Would We Do This Project Again?
With the knowledge we have now, we would still choose to pursue this project, but with adjustments to our approach. The project addresses a meaningful problem and has significant potential for real-world impact. Despite the challenges, the experience was incredibly rewarding, and we gained valuable insights into software-hardware integration, accessibility design, and real-world problem-solving.

If we were to do it again, we would focus on refining our development timeline, streamlining feature implementation, and ensuring earlier stakeholder involvement to maximize the effectiveness of our final product. Ultimately, the experience reinforced our ability to tackle complex engineering challenges and develop a solution that has the potential to improve lives.

13.7 Future Iterations and Expanded Use Cases

While C-ALL was designed to empower individuals who are blind or visually impaired, its underlying technology has far-reaching potential in visually challenging environments where enhanced situational awareness is essential.

Our core innovation—a haptic polar-coordinate feedback system powered by LiDAR and real-time localization—offers 360-degree spatial understanding in the absence of vision. This makes it highly adaptable beyond accessibility contexts.

Future applications include:

- Firefighting, where thick smoke can obscure vision and threaten navigation
- Military special operations, where stealth and low-visibility conditions are frequent
- Cave and underwater exploration, where GPS fails and line of sight is limited

By augmenting traditional tools—such as canes, night vision goggles, or breathing apparatus—with intuitive, tactile navigation, C-ALL can serve as a silent, real-time guide across high-risk environments.

Whether assisting a firefighter through smoke or helping a soldier maintain spatial orientation in darkness, future iterations of C-ALL can multiply human capability through sensory extension.

This vision reflects our commitment to inclusive design—technology that begins with accessibility, but scales to benefit everyone.

Chapter 14

Github Repository

– Neeti Mistry

The GitHub repository for the **Cognitive Assistance with LiDAR Localization (C-ALL)** project serves as the central hub for code, documentation, and updates related to the system's development. As seen in the repository [6], the development has progressed steadily with key milestones and components being implemented in phases.

The repository includes:

- The source code for both the mobile application and the hardware control system
- Documentation on system architecture, setup, and usage
- A detailed list of issues, tasks, and development progress
- A version control system to track changes and allow for collaboration

This repository will continue to be updated as the project evolves, with new features, bug fixes, and optimizations being implemented throughout the development process. It serves as both a tool for development and a resource for the team and any external contributors.

For further details, the GitHub repository can be accessed at: [6] *Cognitive Assistance with LiDAR Localization (C-ALL) GitHub Repository*.

Appendix A

Weekly Reports

– Neeti Mistry

A.1 Week Report 26 (04/25/2025)

What We Did This Past Week
This past week, we focused on refining our documentation report and making final improvements to the hardware. We enhanced the calibration process to ensure better accuracy during obstacle detection and finalized key adjustments to prepare for the class demo. Additionally, we reviewed and incorporated feedback to strengthen both the written and functional components of our project.

What We Will Do Next Week
Next week, we plan to complete and present our final demo, finalize the EXPO poster, and complete the full project report.

List of Action Items:

- Finalize and refine project documentation
- Polish user manual, if time allows
- Continue refining hardware prototype
- Prepare and complete the final demo presentation slides
- Make final updates to the EXPO poster

Issues and Risks:
No issues or risks at the moment.

A.2 Week Report 25 (04/18/2025)

What We Did This Past Week
This week, we focused on refining the calibration and obstacle avoidance code to enhance system accuracy. We also made significant progress on the hardware prototype by expanding from one to two glove devices to

improve functionality and user experience. Additionally, we began developing the final presentation slides for our in-class demo scheduled for April 29, 2025.

What We Will Do Next Week

Next week, we plan to continue refining both the hardware and code to enhance overall performance. We will also keep working on the demo and presentation slides for the upcoming class presentation. Additionally, we aim to improve our EXPO poster by incorporating more visuals and enhance the user manual with additional images. If time permits, we hope to begin working on the front-end user interface.

List of Action Items:

- Refine project documentation
- Finalize user manual
- Design user interface
- Refine hardware prototype
- Further usability testing
- Update EXPO poster with additional images
- Prepare and polish final demo presentation slides

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

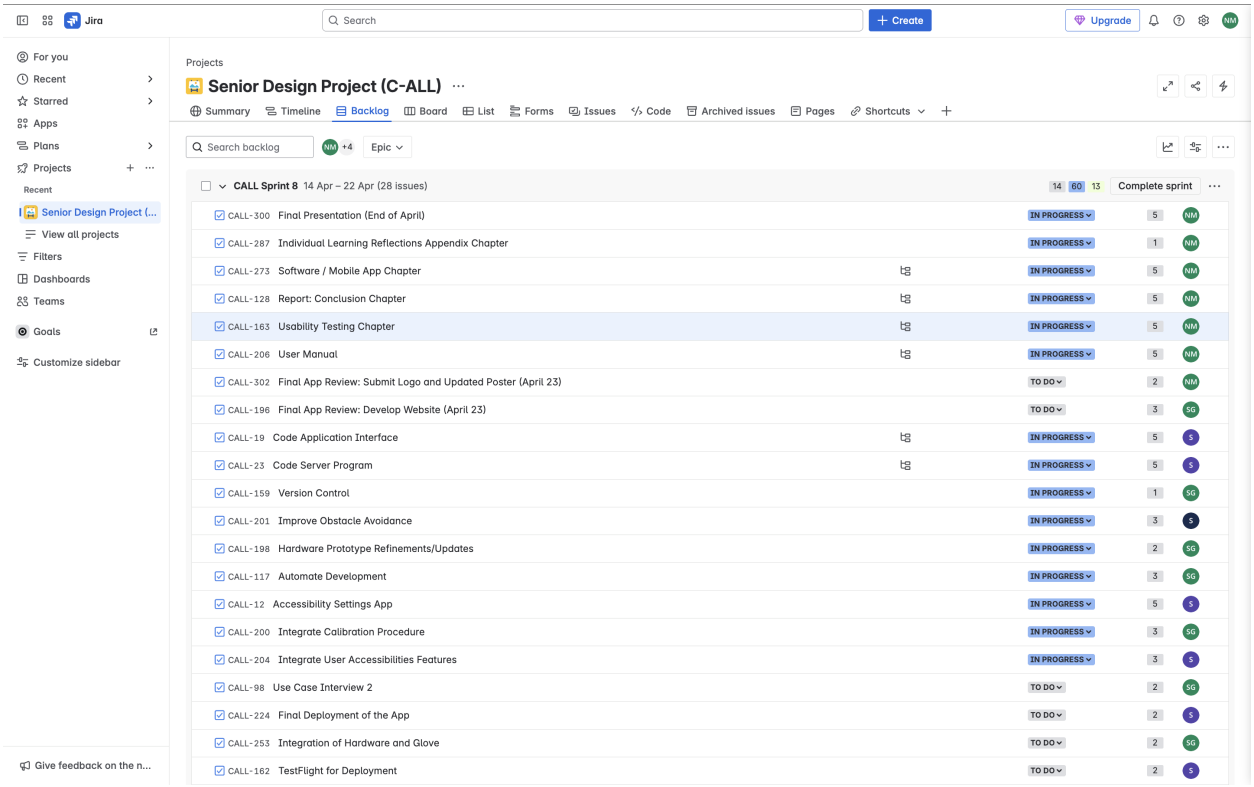

Figure A.1: Jira Sprint Update April 18

UML Diagram Updates

No UML Diagram updates were made at this time.

A.3 Week Report 24 (04/11/2025)

What We Did This Past Week

This past week, our team finalized and developed our official team logo, which is shown below. Additionally, we successfully presented our final Project Release in-class demo on Tuesday, April 8, 2025. During the demo, we showcased the key functionalities of our system, highlighted improvements made since the previous release, and received valuable feedback. Other updates include:

- Developed a new prototype for the calibration system, giving us a second version to compare and test alongside our existing glove prototype.
- Worked on integrating calibration functionality using the built-in limit switches, which we aim to complete by the end of the week.
- Continued refining the obstacle avoidance logic. We are transitioning from a fixed 45-degree offset approach to a more dynamic pathing method that allows the system to navigate around obstacles more intelligently and efficiently.

Figure A.2: C-ALL Official Logo

What We Will Do Next Week

Next week, we aim to accomplish the following tasks:

- Complete the calibration process for our system to ensure accurate real-time navigation.
- Begin designing the user interface, focusing on accessibility settings to make the app more user-friendly for visually impaired individuals.

List of Action Items:

- Refine Project Documentation
- Finalize User Manual
- Complete Calibration
- Design User Interface

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

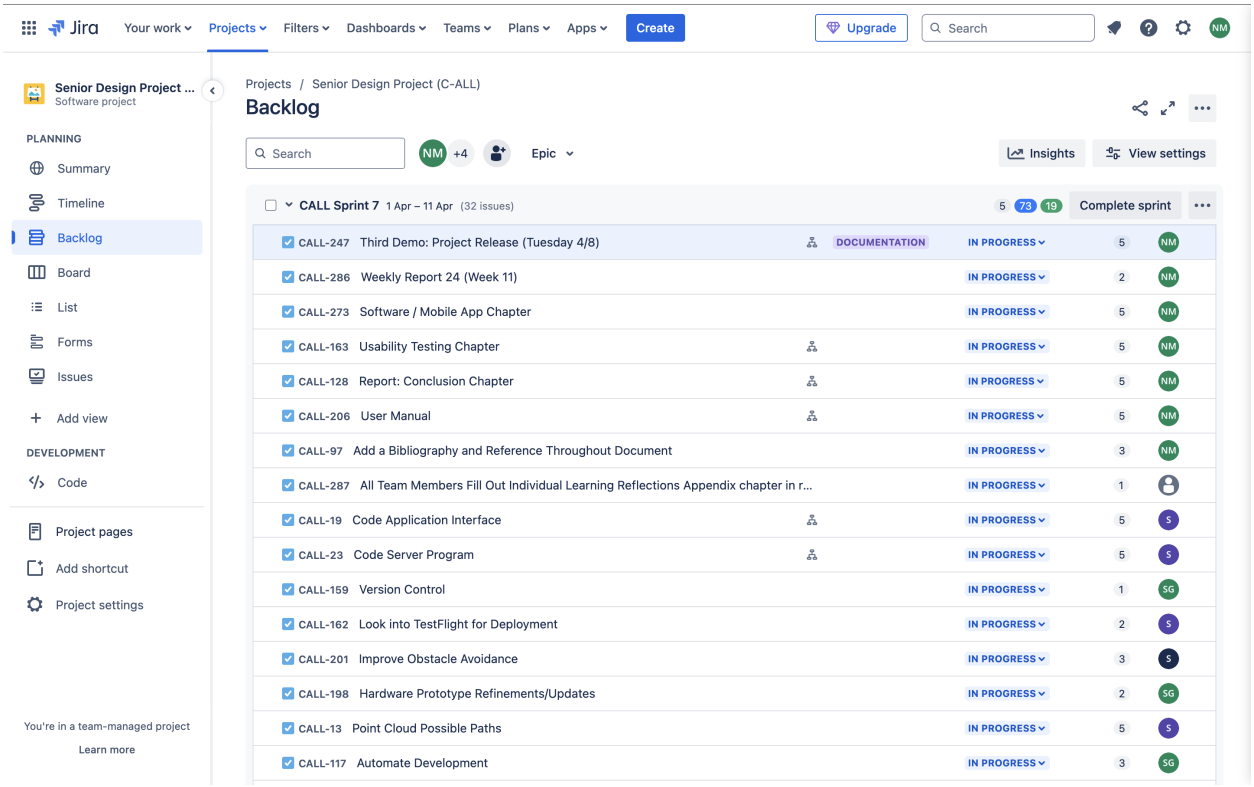

Figure A.3: Jira Sprint Update April 11

UML Diagram Updates

No UML Diagram updates were made at this time.

A.4 Week Report 23 (04/04/2025)

What We Did This Past Week

This past week we continued work on the class demo presentation and added figures to the user manual document. We also developed a team logo and completed our EXPO poster. On the technical side, we were able to get the LiDAR to change the user direction to go right or left based on obstacles. It adds a defined offset of 45 degrees left or right, but it's more reliable, faster, and can detect objects much further away compared to the features. We also tested the best range to detect obstacles and found that 4 meters works best.

What We Will Do Next Week

Next week, we aim to enhance our system by refining the navigation and obstacle detection functionalities. Specifically, we will focus on the following improvements:

- Dynamic Offset Adjustments: The current offset is fixed at 45 degrees, but we plan to modify it so that it dynamically changes based on the user's heading and position relative to an obstacle. This will create a more adaptive and natural navigation experience.

- Improved Obstacle Detection for Thin Objects: Our system currently processes every 8th LiDAR reading to optimize computational efficiency. However, this approach struggles to detect thin obstacles, such as poles. To improve this, we plan to process every 3rd pixel while optimizing performance to ensure the system remains responsive and efficient.
- Path Retention and Boundary Constraints:
  - We now have two pointers: one directing the user towards the next target location and another guiding them through clear pathways. The challenge is ensuring that the user stays on the intended path rather than continuously re-routing around obstacles in ways that could lead them into the street.
  - Our initial idea is to define a 1-meter boundary zone on either side of the target path, marking the safe sidewalk region. However, integrating this directly into obstacle avoidance could be computationally intensive.
  - To maintain efficiency, our proposed solution is to keep navigation and obstacle avoidance in separate threads while using vibration feedback to alert the user when they deviate too far from the intended path. This method allows the system to guide users back without explicitly detecting curbs, aligning with our initial design goals while being faster to implement.

List of Action Items:

- Complete class presentation
- Continue updating codebase and prototype
- Continue adding figures to the user manual
- Continue updating the bibliography

Issues and Risks:
No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

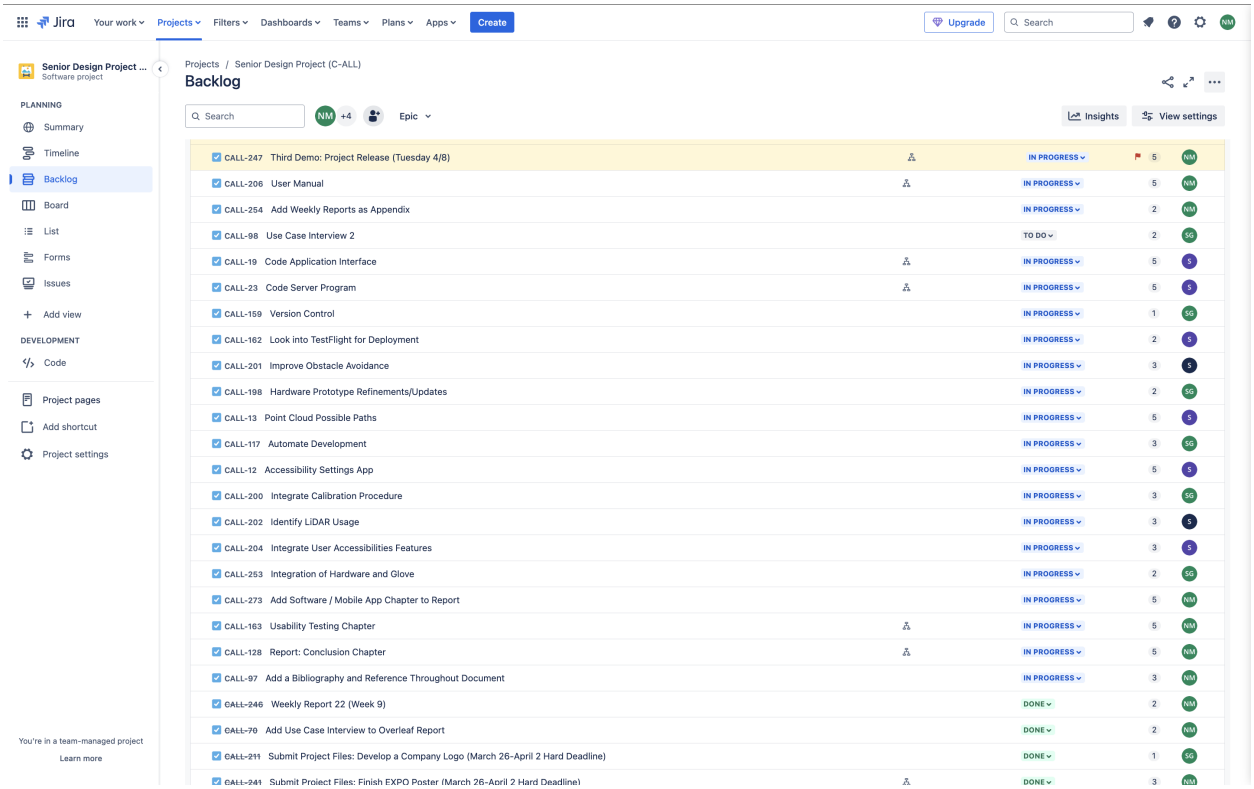

Figure A.4: Jira Sprint Update April 4

UML Diagram Updates

Updated Sequence Diagram (Figure 8.8)

A.5 Week Report 22 (03/28/2025)

What We Did This Past Week

Over the past week, we finalized the design of our EXPO poster and are developing a team logo. We also revised the user manual, incorporating relevant figures and images to enhance clarity. Additionally, we began preparing for the Project Release in-class demo presentation. On the documentation side, we created two new chapters in our report: Usability Testing and Conclusion. For hardware, we are focusing on adding noise to the motor to improve feedback. On the software side, we made further progress on depth avoidance and are continuing efforts to refine and enhance its performance.

What We Will Do Next Week

In the coming week, we plan to continue refining the user manual and continue preparing for the demo presentation. We will also work on expanding the Usability Testing and Conclusion chapters in our report while also completing the team logo. A key focus will be on conducting additional usability testing and connecting with visually impaired individuals to test our prototype and gather valuable feedback. Additionally, we aim to integrate user accessibility features, implement the calibration process, and enhance obstacle avoidance to improve overall system performance.

List of Action Items:

- Continue editing the user manual
- Continue working on demo presentation
- Develop the team logo
- Update the sequence diagram
- Conduct additional usability testing with visually impaired individuals
- Integrate user accessibility features
- Implement the calibration process
- Improve obstacle avoidance functionality

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

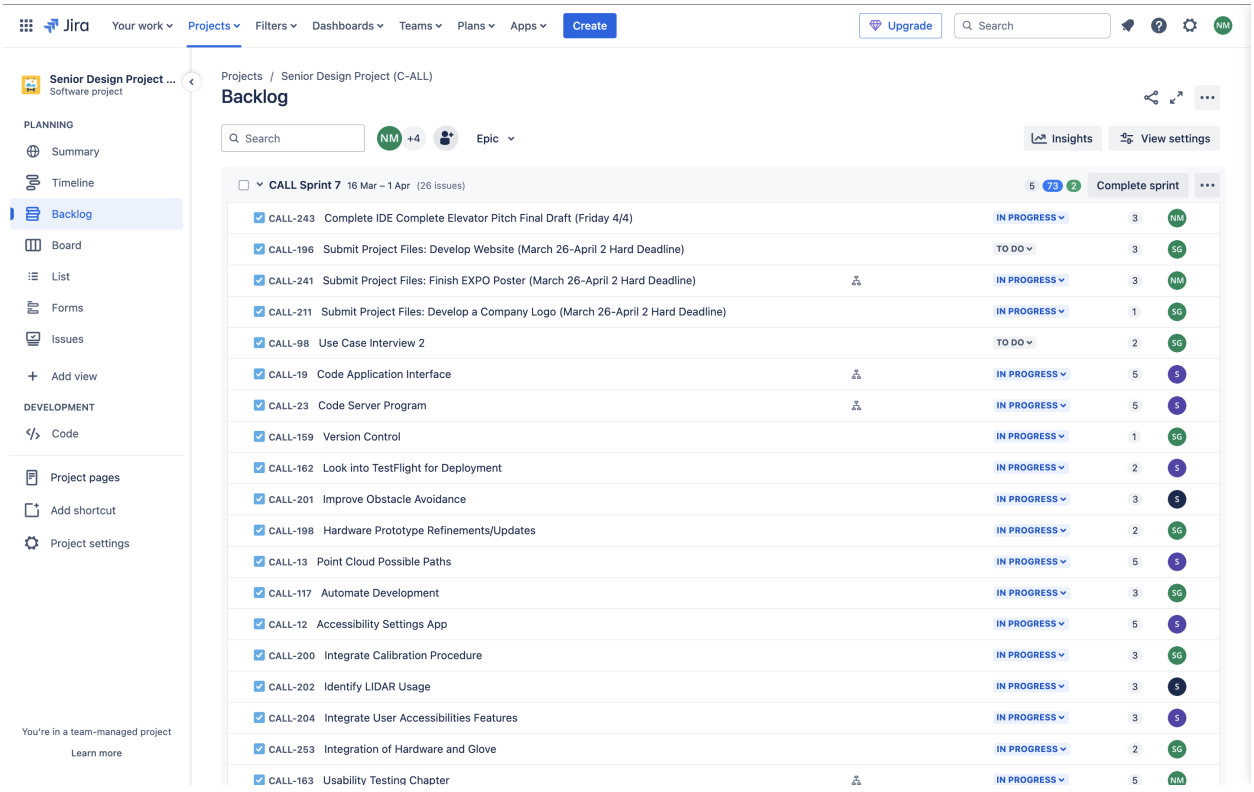

Figure A.5: Jira Sprint Update March 28

UML Diagram Updates

Working on Sequence Diagram but not completed yet.

A.6 Week Report 21 (03/14/2025)

What We Did This Past Week

This past week, we presented the next milestone demo in-class on Tuesday March 12, 2025. We also added a new use case for making software updates (Table 6.6), updated the use case diagram (Figure 6.1), updated requirements (Chapter 5), updated the class diagram (Figure 8.1) and CRC cards (Figure 8.2), and added a new activity diagram to represent this use case (Figure 8.7). We also began developing a user manual for further documentation.

What We Will Do Next Week

Next week, we hope to continue working on the user manual and refining our versioning for the overall project. Additionally, we will continue improving the codebase while exploring TestFlight for deployment and integration of a CI/CD pipeline. We will also work on refining the obstacle avoidance and making necessary prototype improvements. Lastly, we will begin discussions on final deployment strategies for our project.

List of Action Items:

- Continue working on the user manual
- Continue versioning
- Look into TestFlight for deployment and CI/CD system
- Improve obstacle avoidance features
- Hardware prototype refinements and improvements
- Begin considering how we will deploy our project

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

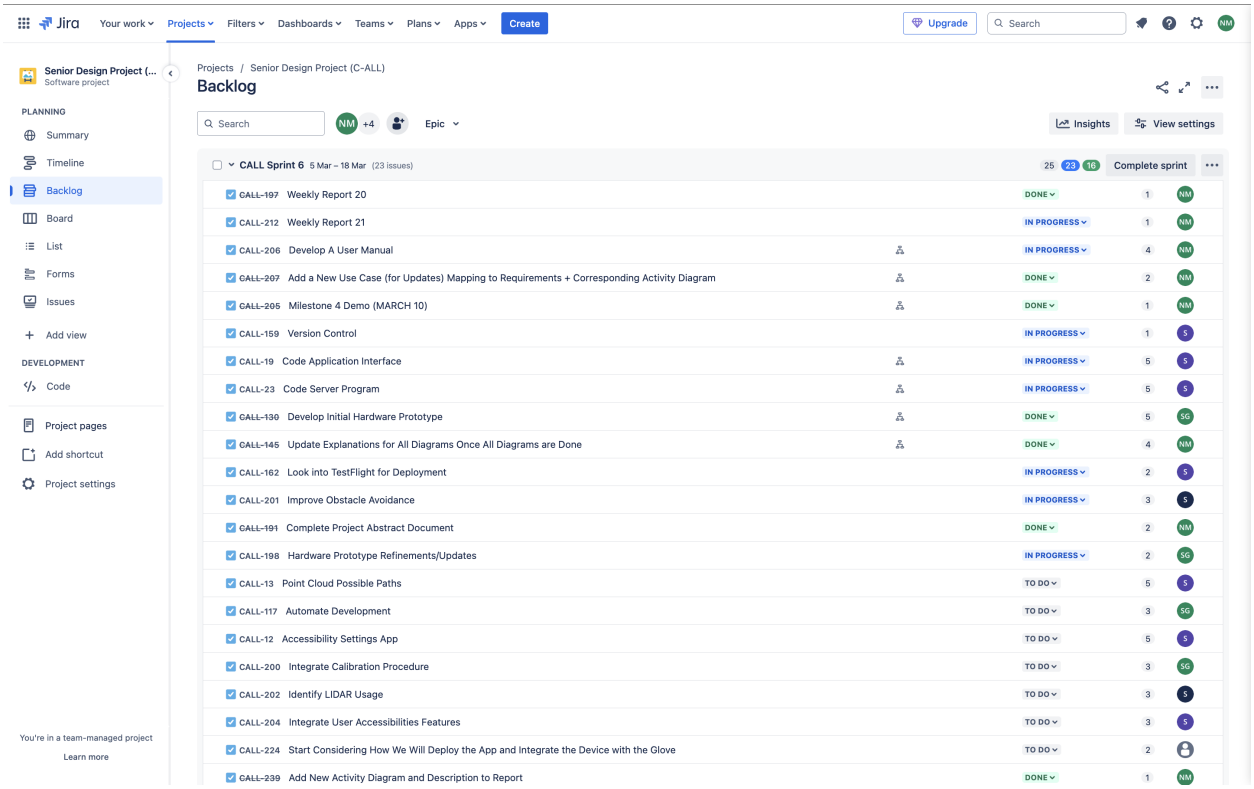

Figure A.6: Jira Sprint Update March 14

UML Diagram Updates

Added a new activity diagram and description for use case 5 (Figure 8.7)
Updated class diagram (Figure 8.1) and CRC cards (Figure 8.2)
Updated Use Case Diagram (Figure 6.1) to include new use case that was added

A.7 Week Report 20 (03/07/2025)

What We Did This Past Week

This past week, we focused on preparing for the next in-class presentation. Additionally, we explored the implementation of TestFlight for testing our application and created new issues to track further improvements. However, TestFlight could not be fully set up due to the lack of an access code from the school, so we plan to schedule a meeting with the professor to resolve this issue. We also started working on a User Manual for our device, incorporating key instructions and guidelines based on the professor's suggestion.

What We Will Do Next Week

Next week, the team aims to finalize the preparation of our upcoming presentation, ensuring it is ready for submission by March 11, 2025. Additionally, as recommended by the professor, we plan to continue developing a user manual for our device to improve ease of use and enhance documentation. Furthermore, we intend to explore versioning strategies and incorporate a new use case and activity diagram to show the process of making updates to the software.

List of Action Items:

- Complete next milestone demo presentation
- Add new use case (for making updates) and complete activity diagram
- Continue developing a comprehensive user manual
- Schedule a meeting to resolve TestFlight access issues
- Refine hardware prototype based on recent developments
- Begin initial usability testing and gather feedback

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

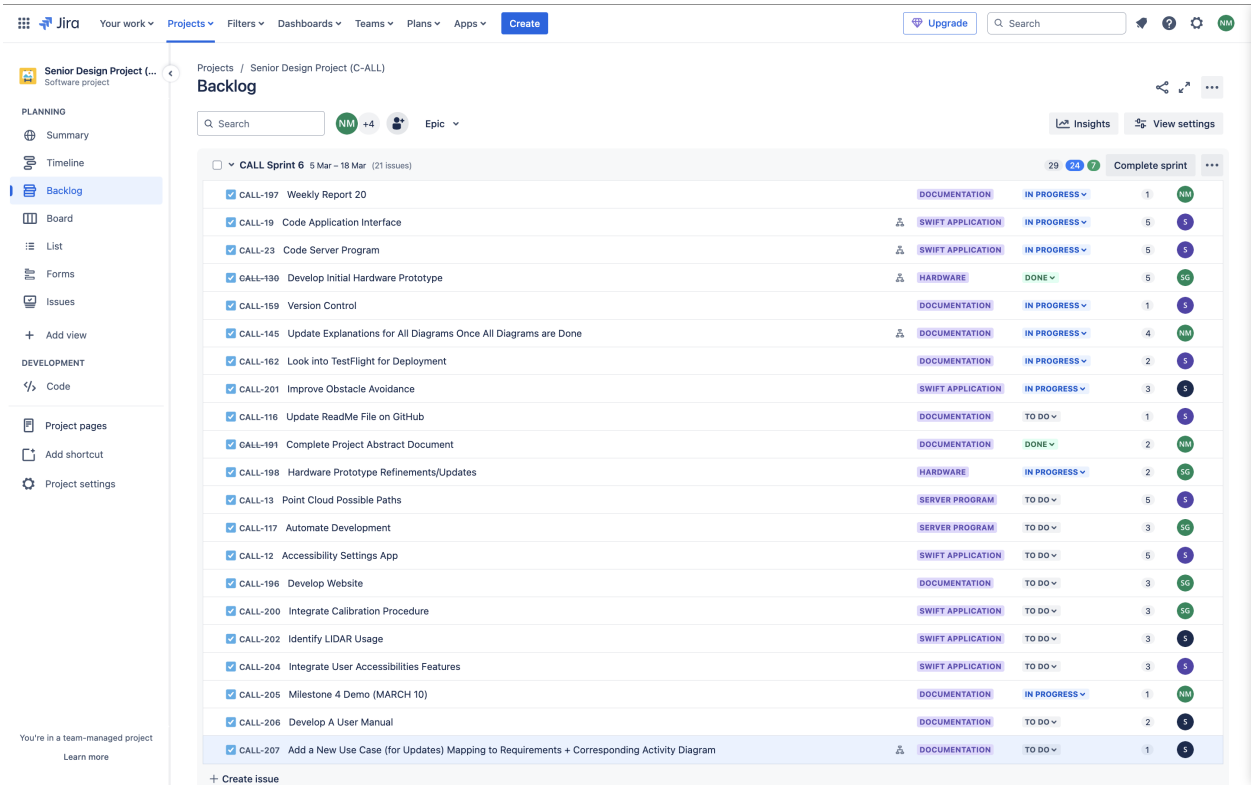

Figure A.7: Jira Sprint Update March 7

UML Diagram Updates

Updated Deployment Diagram (Figure 8.10)

A.8 Week Report 19 (02/28/2025)

What We Did This Past Week
This past week we continued making refinements to the hardware design and software code base. We also presented a Preliminary Implementation Demo in class.

What We Will Do Next Week
Next week we plan on getting started on usability testing and working on interviewing visually impaired clients to test our prototype and/or gather feedback on any modifications we should make.

List of Action Items:

- Finalize hardware refinements and ensure all components are functioning properly
- Conduct initial usability testing with the prototype
- Schedule interviews with visually impaired clients
- Gather feedback on hardware and software usability
- Identify and document potential modifications based on user feedback
- Continue refining the software code base for improved performance
- Update project documentation to reflect recent changes and testing outcomes

Issues and Risks:
No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

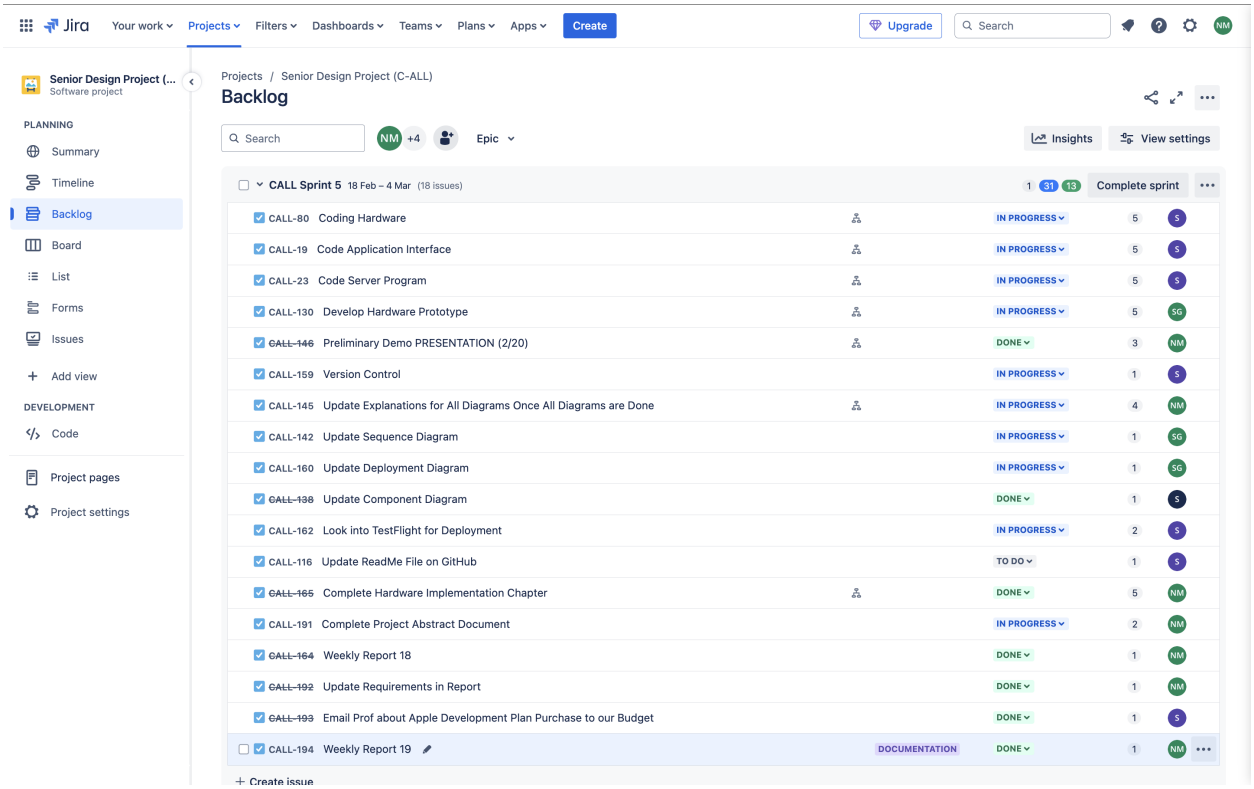

Figure A.8: Jira Sprint Update February 28

UML Diagram Updates

No updates at this time.

A.9 Week Report 18 (02/21/2025)

What We Did This Past Week

This past week, we completed and presented our Preliminary Implementation Demo in class on Thursday, February 20. The presentation included updates on our use case diagrams, activity diagram, class diagram and CRC cards, requirements, versioning logic, hardware prototype, and working software demos. Additionally, we created a new chapter in our report dedicated to the Hardware Device Implementation (Chapter 11), documenting the integration process and technical details. On the hardware side, we continued working on the prototype and 3D printed various iterations of our SolidWorks model. We were also able to combine our hardware materials and integrate them with the Raspberry Pi, resulting in a working hardware device that aligns with our system's functionality.

What We Will Do Next Week

Next week, we will continue refining the hardware prototype to improve its design and functionality while further developing the software to enhance system performance and integration. We also hope to begin a usability testing phase to gather feedback and evaluate the effectiveness of our current working design.

List of Action Items:

- Complete Hardware Implementation chapter
- Begin Usability Testing chapter
- Continue working on hardware device
- Refine and test software integration with hardware
- Update report with recent hardware and software developments

Issues and Risks:

No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

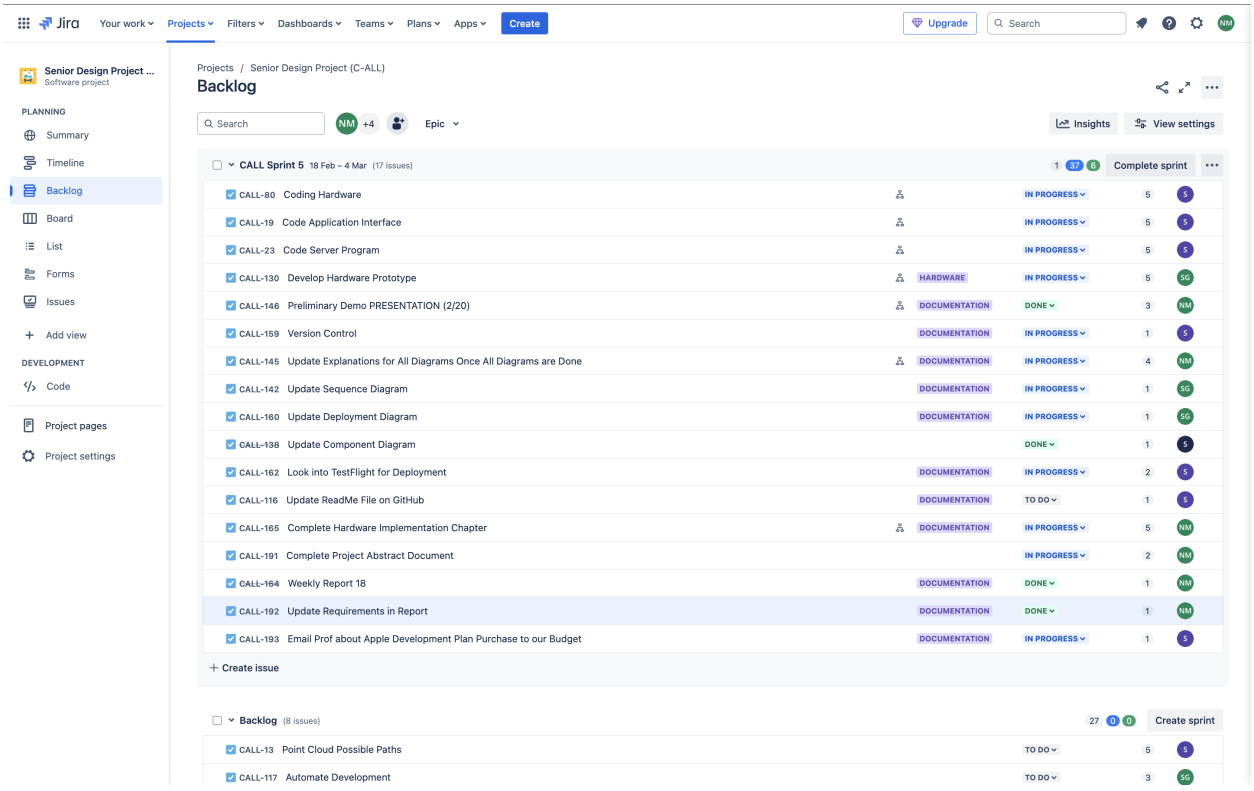

Figure A.9: Jira Sprint Update February 21

UML Diagram Updates

This week we made a minor change to the Component Diagram (Figure 8.9) to reflect that we are no longer using Arduino and using Raspberry Pi instead. We also updated all of the explanations in the report for all the updated diagrams to keep this document up to date as best as possible.

A.10 Week Report 17 (02/14/2025)

What We Did This Past Week
This past week, we focused on refining our use case diagrams and tables to ensure consistent naming across all documentation. We also made updates to the Sequence Diagram and Activity Diagrams to improve clarity and accuracy. In preparation for our Preliminary Demo, we continued working on the slides presentation. Additionally, we worked on the SolidWorks prototyping and 3D printed initial hardware models to support our design process.

What We Will Do Next Week
Next week, we will finalize our presentation slides for the Preliminary Demo and refine our 3D-printed hardware design to improve its functionality. Additionally, we will work on coding the prototype to ensure it properly integrates with the app software interface. We will also continue updating our documentation to reflect project changes.

List of Action Items:

- Complete Preliminary Demo slides
- Refine printed model design
- Continue working on code
- Update Deployment Diagram based on professor's feedback

Issues and Risks:
No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

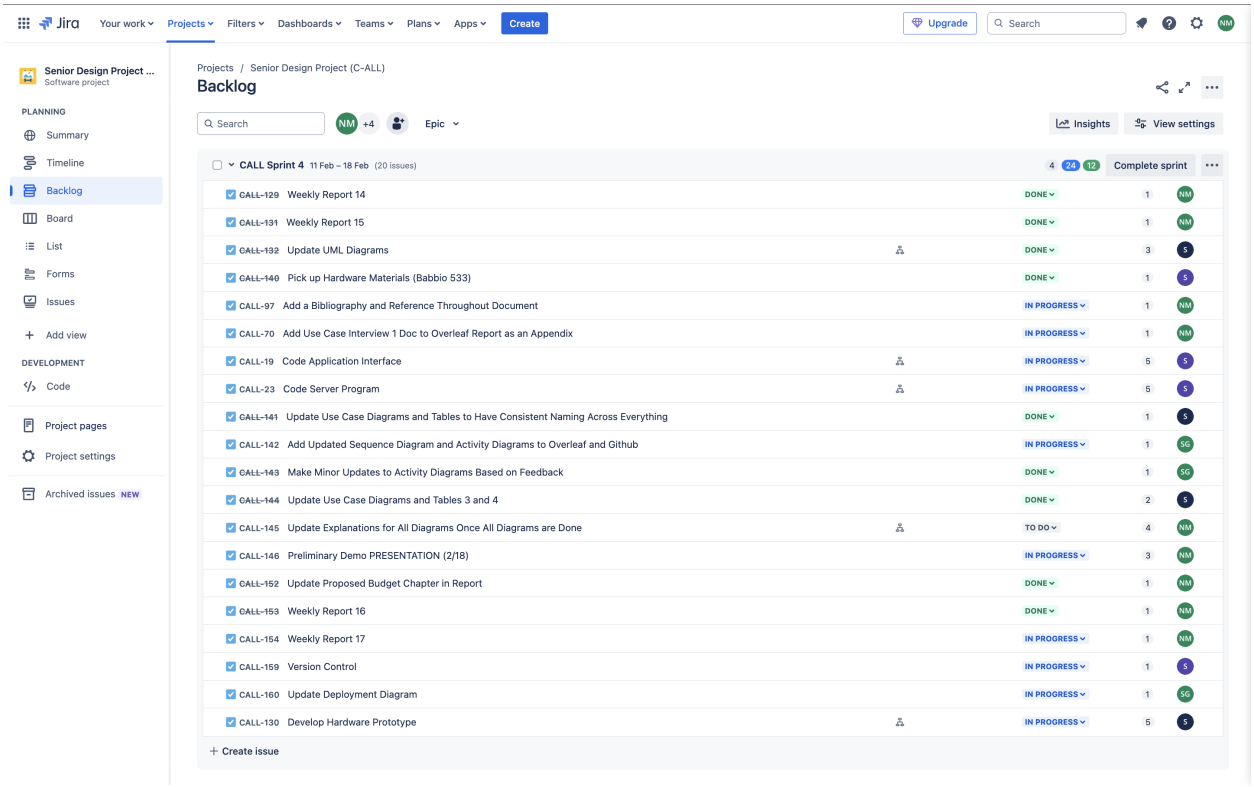

Figure A.10: Jira Sprint Update February 14

UML Diagram Updates
The use case tables were updated 6.1 to have consistency in naming. The activity diagrams 8.3 were also updated to provide a more accurate representation.

A.11 Week Report 16 (02/07/2025)

What We Did This Past Week
This past week, we focused on refining our UML diagrams based on feedback from the professor and discussions during class meetings. We updated the logic of several UML diagrams, ensuring they more accurately reflect the system's design.

Key updates include:

- Adding a digital figure of our CRC cards to improve clarity and documentation
- Enhancing our class diagram to make it more detailed and aligned with the current codebase
- Ensuring name consistency across all use case diagrams and tables for better uniformity
- Updating the sequence diagram to improve accuracy and flow

On the hardware side, we acquired the ordered materials, reviewed our inventory, and updated our list of items that need to be returned or additionally purchased. For the code, we made significant progress in

directing the user towards the main path and integrating this logic directly into the hardware tool.

What We Will Do Next Week

Next week, we will build a hardware prototyping plan, ensuring we have a clear roadmap for development and testing. We will also make further modifications to the use case diagrams and activity diagrams to improve accuracy and consistency. Additionally, we will begin preparing for our Preliminary Demo, which is scheduled to be presented in two weeks, focusing on key functionalities and system integration.

List of Action Items:

- Finalize the hardware prototyping plan
- Develop presentation materials for the Preliminary Demo
- Modify use case and activity diagrams.

Issues and Risks: No issues or risks at the moment.

Sprint Screenshot Showing Issues (Jira)

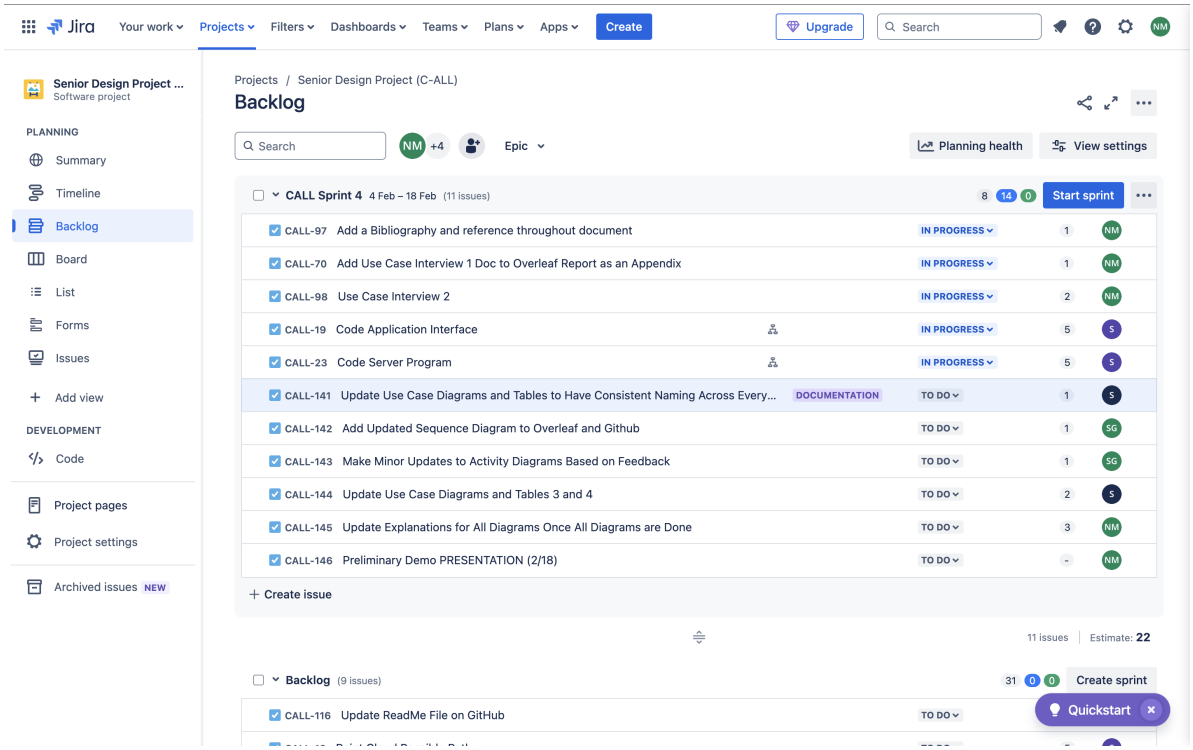

Figure A.11: Jira Sprint Update February 7

UML Diagram Updates

The use case diagram was updated 6.1 to combine the 4 separate use case diagrams we had earlier into a single, more cohesive diagram. The sequence diagram 8.8 was also updated to provide a more accurate representation. The first two activity diagrams 8.3 and 8.4 were updated to better follow the logic flow of the updated use cases.

A.12 Week Report 15 (01/31/2025)

What We Did This Past Week

This past week, we began updating the class, use case, and sequence diagrams, incorporating initial revisions based on professor feedback. However, further updates and refinements are still needed to fully align with the system design requirements. In terms of the codebase, we made significant progress in mapping the sidewalk instead of just the main road, improving navigation accuracy. Additionally, we decided to discontinue the use of Apple Maps for mapping due to incompatibility issues and have explored alternative solutions to ensure better functionality.

What We Will Do Next Week

Next week, we plan to acquire the ordered hardware materials and begin planning the hardware prototyping phase. As part of this process, we will assess whether any additional materials need to be ordered and develop a detailed plan for prototyping. Alongside hardware development, we will continue refining UML diagrams based on feedback and updating the code to align with recent improvements in system design.

List of Action Items:

- Continue updating UML diagrams
- Improve FailSafe use case to represent handling of all errors
- Conduct another use case interview to gather additional insights
- Plan and prepare for the next phase of hardware development
- Acquire and assess the ordered hardware materials for prototyping

Issues and Risks:

Issue: One of the primary challenges we are currently facing is merging pathfinding and obstacle avoidance effectively. Currently, we are using ARGeotracking to obtain the exact location of the user within the environment, allowing us to properly guide them along sidewalks and paths. At the same time, we are using ARWorldTracking to access LiDAR for obstacle detection. However, these two systems cannot run simultaneously, posing a significant limitation. Switching between ARGeotracking and ARWorldTracking is also not a viable option because doing so requires closing the active session, which results in all previously collected data being deleted. Additionally, LiDAR scanning speed is a limiting factor, as it requires a few seconds to map the environment, distinguish obstacles, and identify the floor. Constantly restarting the session would reset this process, causing delays in real-time navigation. Similarly, restarting geotracking would force the system to recalculate coordinates.

Risk: Navigation performance delays due to session restarts could significantly impact real-time assistance and user safety.

Mitigation: At this point, we are exploring potential solutions to integrate both functionalities without data loss or excessive latency. We hope to explore potential background processing methods that could preserve geotracking and LiDAR data without fully restarting sessions, and research alternative sensor fusion techniques that can optimize geolocation and obstacle avoidance without requiring session resets.

Sprint Screenshot Showing Issues (Jira)

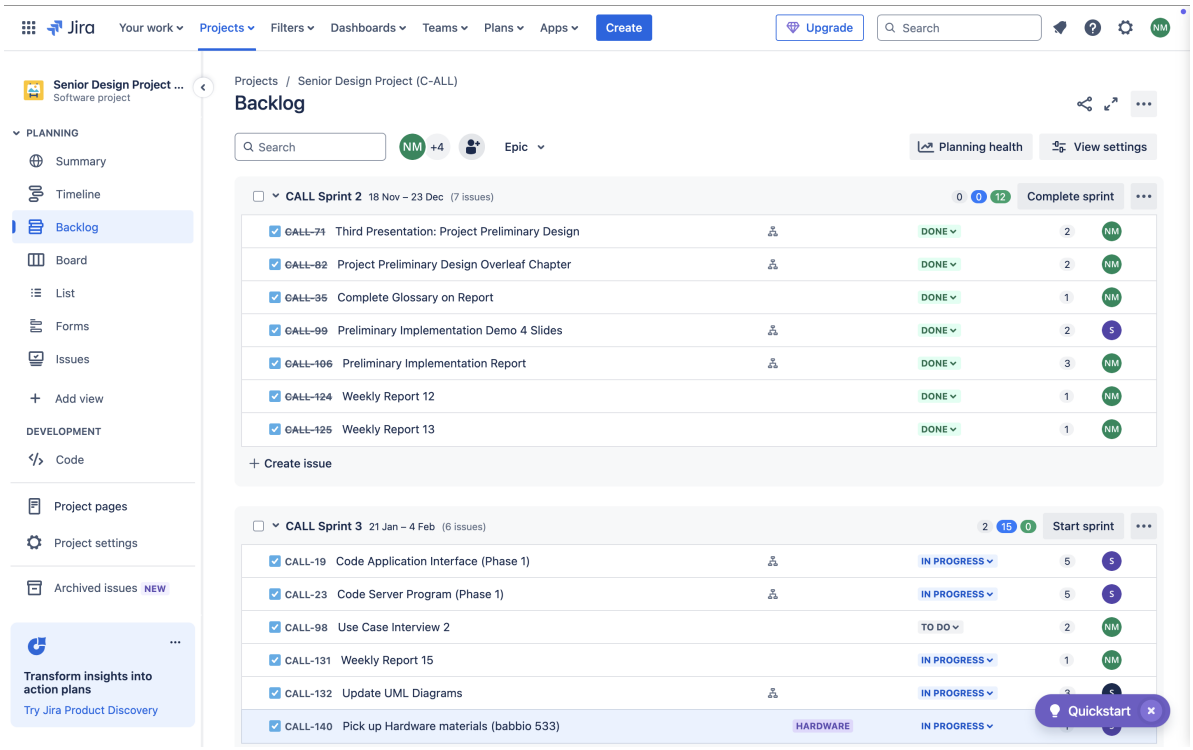

Figure A.12: Jira Sprint Update January 31

UML Diagram Updates

The class diagram 8.1 has been updated to contain functions and collaboration with other classes according to the most recent version of the software. The CRC cards 8.2 have also been made to show a high level description of the relationships between and purposes of classes.

Note: Class diagram is yet to include hardware logic.

A.13 Week Report 14 (01/24/2025)

What We Did This Past Week

This past week, we had our first class of the spring semester following winter break. As a team, we reviewed the progress made over the break and discussed our plan moving forward. We reorganized and created new issues in our Jira sprint to better manage and track our tasks. Additionally, we began updating our current UML diagrams, making improvements based on feedback from the professor.

What We Will Do Next Week

Next week, we plan to continue refining and improving the UML diagrams to ensure they better depict the system's code and functionality.

List of Action Items:

- Update UML diagrams to align with feedback and better represent the system architecture and codebase
- Update the project report to reflect recent progress, including improvements to UML diagrams and system design
- Review and optimize existing code for navigation and obstacle detection
- Begin integrating new features identified during the planning phase
- Conduct more use case interviews with potential clients
- Plan and prepare for the next phase of hardware development, such as acquiring materials and refining prototype design

Issues and Risks:

Issue: Ensuring that the updated UML diagrams accurately reflect the evolving codebase and system design.

Risk: Potential delays if diagram updates take longer than expected.

Mitigation: Prioritize the most critical diagrams first to ensure timely completion of essential updates. Regularly consult with team members responsible for coding and seek professor feedback to confirm accuracy.

Sprint Screenshot Showing Issues (Jira)

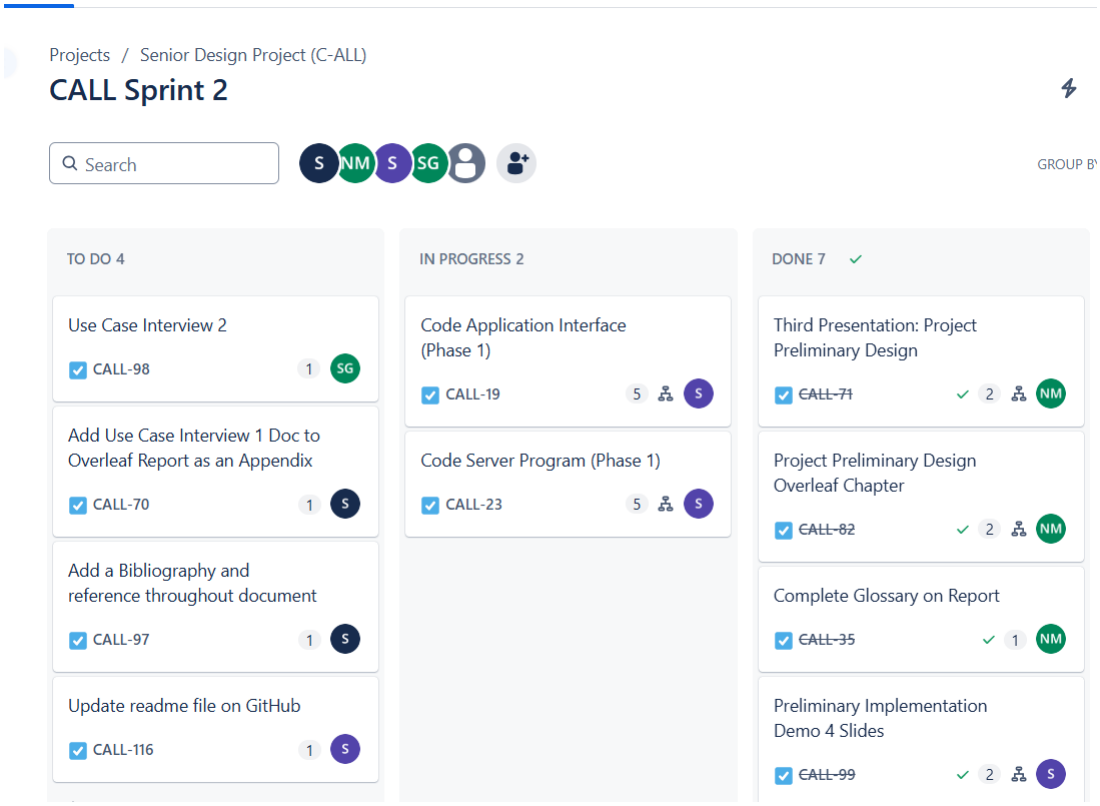

Figure A.13: Jira Sprint Update January 24

UML Diagram Updates

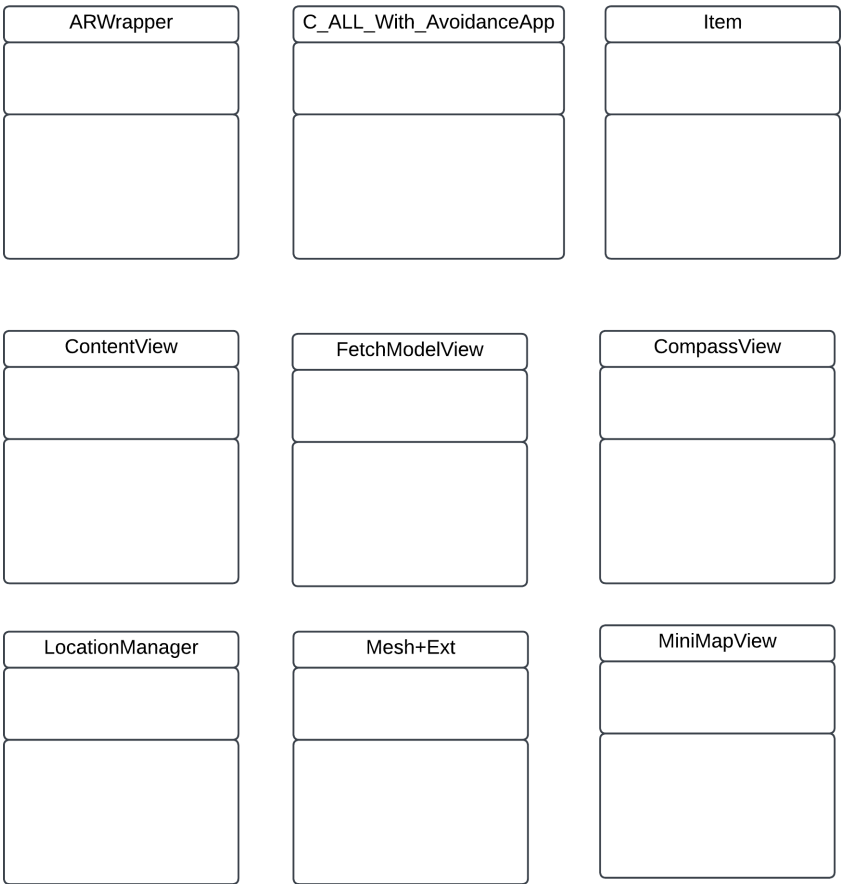

Figure A.14: Avoidance Class Diagram Update

This is an early version of a re-designed class diagram which incorporates many classes involved in the object avoidance calculations within the C-ALL mobile application. This will be combined with more classes which handle pathfinding as well as the Arduino control code. See Chapter 8 for more information on the system architecture.

A.14 Week Report 13 (12/13/2024)

What We Did This Past Week

This past week, we successfully completed the final in-person presentation and demo of the semester, focusing on the Preliminary Implementation of our project. Additionally, we finalized the corresponding chapter in the report, ensuring all details were accurately documented. Based on our professor's feedback from the previous demo, we updated a class diagram to address the suggested improvements. Moreover, we dedicated time to further developing the codebase, implementing key functionalities and refining existing features to align with our project goals.

What We Will Do Next Week

Next week, we plan to outline our goals and identify key areas of focus for the project over winter break. Since we will not be meeting in class, we aim to set individual milestones to ensure continued progress

during this period. Additionally, we will continue working on the codebase, addressing any remaining issues and optimizing the system's performance. Furthermore, we will incorporate additional updates into the report, including refinements to diagrams, code documentation, and a summary of our progress to date. We also hope to focus more on the hardware and developing the app after break.

List of Action Items:

- Develop a plan for winter break, including individual tasks and goals for the project
- Continue refining and optimizing the codebase to address remaining issues
- Update and improve diagrams, ensuring alignment with project requirements and feedback
- Enhance code documentation and add relevant sections to the project report

A.15 Week Report 12 (12/06/2024)

What We Did This Past Week
This past week, we finalized the User Interface Design chapter, incorporating detailed user personas and an initial prototype that aligns with the identified personas. The System Architecture chapter was updated with key UML diagrams. Additionally, we integrated the GitHub repository into the documentation to showcase development progress and version control. We developed a deployment diagram to represent system distribution and the interaction across components. We continued refining requirements and use cases based on feedback from the preliminary demo and documentation review. Furthermore, we continued to develop additional user personas and refined the user interface to better match their specific needs.

What We Will Do Next Week
Next week, we plan to complete our presentation for the final demo of the semester, the Preliminary Implementation Demo, ensuring it highlights our progress and achievements effectively. Additionally, we aim to continue refining the corresponding chapter in the report, integrating updated insights and documentation. Finally, we hope to conduct initial usability testing with visually impaired users to gather valuable feedback on the system's functionality and interface design.

List of Action Items:

- Complete final presentation demo of the semester
- Complete Preliminary Implementation
- Enhance the UI prototype based on persona-specific needs
- Coordinate with team members to schedule usability testing and collect feedback

A.16 Week Report 11 (11/22/2024)

What We Did This Past Week
This past week, we finalized the User Interface Design and System Architecture chapters of our project report. For the User Interface Design, we developed detailed user personas to better understand the needs

of our target audience and created a preliminary paper prototype to visualize and refine the system's UI interactions. In the System Architecture chapter, we developed key user stories and created several UML diagrams to represent the system's structure and behavior. Using the 4+1 View Model, we described the design of our system through key activity diagrams for each use case, as well as a class diagram, sequence diagram, component diagram, and deployment diagram. Additionally, we presented our third demo in class on November 19, 2024, showcasing the progress of our project.

What We Will Do Next Week
Next week, we hope to conduct another client interview to review our user personas and user stories, as well as gather feedback on our UI prototyping. This feedback will help us better align our system with client expectations and refine its usability. Additionally, we are considering creating a Google Form to distribute to visually impaired clients. This form will allow us to collect contact information and establish a centralized database to streamline future client interviews. Alongside these efforts, we hope to make further progress on the development of our LiDAR code.

List of Action Items:

- Conduct a client interview to review user personas, user stories, and UI prototyping
- Create and distribute a Google Form to visually impaired clients for collecting contact information
- Refine and further develop the LiDAR code for improved system functionality
- Align feedback from client interviews with use cases and project goals

Any Issues and Risks (and what's being done about them):

Issue: The LiDAR sensor is only available on the Pro models of the iPhone, and only one team member owns this model. This limitation creates challenges in testing, as it requires frequent coordination with that member and scheduling additional meetings to access the necessary device.

Risk: The dependency on a single team member's device could delay testing and hinder progress, particularly if there are scheduling conflicts or if the device becomes unavailable.

Mitigation Plan: To address this issue, we plan to consult with the professor to discuss potential solutions. This includes exploring the possibility of acquiring a compatible iPhone for dedicated testing purposes. Having a shared device would streamline the testing process, reduce dependency on a single team member, and improve efficiency in meeting project milestones.

Highlight any work you've done on the 4+1 views, or any updates you've done on the documentation, in general:
This week, we made significant progress on the 4+1 View Model for our system architecture. Updates included refining the class diagram in the Logical View, clarifying module interactions in the Development View through an updated component diagram, and finalizing activity diagrams for key use cases in the Process View. The Physical View was enhanced with a detailed deployment diagram showing hardware configurations, while the Scenarios View was updated with more detailed user stories based on team discussions. Additionally, we updated the User Interface Design chapter, incorporating user personas and a preliminary paper prototype. These updates ensure our documentation is comprehensive and clearly communicates the system's functionality and user interactions.

A.17 Week Report 10 (11/15/2024)

What We Did This Past Week
This past week, our team focused on finalizing the core elements needed for our upcoming in-class presentation. We focused on refining our system architecture and solidifying the remaining components of our user interface design. Additionally, we worked on completing the final set of UML diagrams to better represent our system's structure, processes, and user interactions. These diagrams are essential for clarifying project functionalities and flow to our stakeholders.

What We Will Do Next Week
Next week, we will concentrate on finishing our third presentation demo, scheduled for November 19, 2024, where we'll showcase the system architecture and UI Design. This includes polishing the presentation content to effectively communicate our project's unique value and demonstrating key diagrams. We also plan to conduct another use case interview to gather more details from potential users and clarify some questions we have regarding the everyday lives/routines of the visually impaired (and how our solution can fit into that), ensuring our project aligns with their needs.

List of Action Items:

- Finalize and complete the third presentation for the in-class demo on November 19, 2024.
- Complete the remaining UML diagrams for system documentation.
- Conduct a final review of the UI design to ensure alignment with user requirements.
- Conduct another use case interview to gather further details and refine project scope based on user needs.

A.18 Week Report 9 (11/08/2024)

What We Did This Past Week
This past week, we started working on the Project Preliminary Design, focusing on the development of our User Interface Design and System Architecture. We created some UML diagrams, including an Activity Diagram, but we still need to complete the remaining diagrams. Additionally, we successfully set up our Mac environment, allowing us to begin coding our software.

What We Will Do Next Week
Next week, we hope to complete the rest of the UML Diagrams for the System Architecture and begin preparing for our third presentation demo on November 19, 2024.

List of Action Items:

- Complete the development of all UML Diagrams
- Continue refining UI Design Development and create paper prototypes
- Advance work on System Architecture
- Prepare the presentation for the in-class demo on 11/19/2024

A.19 Week Report 8 (11/01/2024)

What We Did This Past Week
This week, we focused on developing several UML diagrams and troubleshooting to get our Xcode environment up and running. We successfully acquired a Mac environment, allowing us to begin coding.

What We Will Do Next Week
Next week, our team will focus on the Project Preliminary Design, which emphasizes user interface design and system architecture. This phase will be instrumental in preparing us for our third demo and in-class presentation.

List of Action Items:

- Begin working on Project Preliminary Design
- Work on Activity Diagrams, Package Diagrams, Class Diagrams, and Deployment Diagrams

A.20 Week Report 7 (10/24/2024)

What We Did This Past Week
This past week, we successfully presented our second in-class demo of the Project Specification on October 24, 2024. This demo provided an opportunity to showcase our progress and gather feedback from peers and instructors. Simultaneously, we have been working toward completing our Requirements and Use Cases, a crucial step in finalizing our Project Concept Development and Specification. These elements will ensure a clear understanding of the system's functionality and user interactions, which are vital for the project's development moving forward. Additionally, we have also set up a Mac environment in the lab, configuring all necessary tools and applications required for the project. This setup ensures that our development environment is fully operational, allowing us to move forward efficiently with the implementation phase.

What We Will Do Next Week
Next week, our focus will be on further developing our user stories and creating the corresponding UML diagrams to better define the system's architecture and user interactions. These elements will help solidify our understanding of our project's functionality from both a technical and user perspective. Additionally, we aim to strengthen our connection with the visually impaired community — our primary user base. We plan to gather more insights and feedback by engaging with potential clients, allowing us to refine our user stories and ensure the system meets their needs effectively. This feedback will be invaluable as we continue shaping the direction of our project.

List of Action Items:

- Complete requirements
- Complete use cases and diagrams
- Develop user stories

A.21 Week Report 6 (10/17/2024)

What We Did This Past Week
This past week, our team worked on the Requirements chapter and added a Glossary to our report document. We outlined key stakeholders, focusing on user requirements and system requirements. We also had a second use case interview with our client to further develop user needs and refine the requirements.

What We Will Do Next Week
Next week, we will continue working on the Project Concept Development and Specification, where we will complete our second in-class presentation on October 24th and the Requirements and Use Case chapters. We will finish writing our main system requirements and then move on to non-functional and domain requirements.

List of Action Items:

- Complete Project Concept Development and Specification
- Complete second in-class demo presentation
- Work on use case diagrams and use case descriptions

A.22 Week Report 5 (10/10/2024)

What We Did This Past Week
This past week, our team made significant progress on our project. We successfully submitted our proposed budget, outlining the necessary costs for hardware components, software tools, and other essential resources. Additionally, we organized our project management tools, opting to use Jira to track issues instead of GitHub due to its detailed project management functions, such as enhanced issue tracking and team coordination features. We also gave our first in-class presentation, where we introduced our project idea, the problem we aim to solve, and our development plan, including key milestones and team roles. The feedback we received during the presentation was valuable and will help guide our next steps as we move into the requirements phase.

What We Will Do Next Week
Next week, our primary focus will be on several key tasks essential to advancing the project. We plan to add a glossary to the report to ensure consistency and clarity in terminology. Additionally, we will complete the detailed project requirements chapter, specifically outlining key stakeholders, focusing on user requirements, system (constraints) requirements, non-functional (quality) requirements, and domain (business) requirements. Another important task for next week is to schedule and conduct a second use case interview with our clients to gather more insights into user needs and refine the requirements further.

List of Action Items:

- Add a detailed Glossary to report
- Complete detailed Requirements (Project Concept Development and Specification)
- Determine further User Requirements

Any Issues and Risks (and what's being done about them):

- One potential issue we are currently facing is ensuring that we accurately capture the needs of our visually impaired users during the requirements gathering phase. Misinterpreting their needs could lead to design flaws that affect the system's usability. To mitigate this, we are conducting multiple use case interviews and working closely with our clients to validate the gathered information and adjust as necessary.
- Additionally, managing budget constraints could become a challenge if unforeseen expenses arise. To address this, we are closely monitoring our spending and ensuring that all hardware purchases remain within the allocated budget.

A.23 Week Report 4 (10/03/2024)

What We Did This Past Week
This week, we finalized the Development Plan for our project, creating a thorough and detailed overall strategy. Additionally, we prepared a presentation for our first demo, which we presented in class on October 3, 2024. The presentation covered our overall project mission, identified key constraints and drivers, and showcased some use cases along with an activity diagram.

What We Will Do Next Week
Next week, our team will focus on several key tasks to advance our project. We will complete the Requirements phase by identifying key stakeholders and gathering user requirements to ensure we fully understand the needs of our target audience. Additionally, we will create an itemized list of hardware and materials needed for budget approval, laying the groundwork for our project's resource allocation. To further refine our understanding of user needs, we plan to conduct additional use case interviews. We will also begin developing the first iterations of essential diagrams, including use case diagrams, activity diagrams, and an architecture diagram, to visually represent our project's structure and workflows.

List of Action Items:

- Complete Requirements, identifying key stakeholders and user requirements
- Create an itemized list of hardware and materials needed for budget approval
- Conduct further use case interviews to ensure user requirements and needs
- Start developing the first iterations of diagrams, such as use case diagrams, activity diagrams, and architecture diagram

Any Issues and Risks (and what's being done about them):

- **Risk: Conflicting User Needs** - Different users within the visually impaired community may have varying and potentially conflicting needs, making it challenging to prioritize features.
  - **Mitigation:** We will analyze feedback from user interviews to identify common themes and prioritize the most critical needs. A user advocacy role within our team will ensure that the focus remains on delivering value to the end users.

- **Risk: Technical Challenges** - Integration of various hardware components and software systems may present unexpected technical challenges that could delay progress.
  - **Mitigation:** We will conduct thorough research and testing of hardware and software compatibility early in the development phase. Regular brainstorming sessions will encourage team collaboration to troubleshoot issues as they arise.
- **Risk: Budget Constraints** - A limited budget and approval time may restrict our ability to procure necessary hardware and materials in time, impacting the project's development.
  - **Mitigation:** We will prioritize essential items on our itemized budget list and seek to minimize costs through creative solutions, such as utilizing available resources on campus.

A.24 Week Report 3 (09/27/2024)

What We Did This Past Week
This past week, we began working on our project's development plan and first presentation slides that covers our overall project mission, key drivers, key constraints, and our overall development plan. In preparation for our first presentation, we began creating slides that outline these components, ensuring clarity in our approach and goals. To refine the project's scope and determine the best course of action, we consulted with Professor Lu Xiao, drawing insights from a previous LiDAR-based project. This communication has helped us better understand the challenges and opportunities ahead, enabling us to define a clear direction for the project. We also interviewed a visually impaired student in order to gain more information and understanding of the needs and use cases of the visually impaired community. This interview provided valuable insights into their experiences and specific requirements, which helped us identify key features and functionalities to incorporate into our project. By gaining a deeper understanding of their perspectives, we are better equipped to design solutions that truly meet the needs of the visually impaired community.

What We Will Do Next Week
Next week, we plan to finalize both the Development Plan and the presentation, ensuring that all elements align with the project's mission and scope. This will provide a solid foundation as we move forward with the project.

List of Action Items:

- Complete Development Plan
- Complete first presentation slides
- Further enhance Github issues and Kanban board with more thorough details

Any Issues and Risks (and what's being done about them):

- Risk: As we incorporate insights from our interview with the visually impaired student, there is a risk of expanding our project objectives beyond what was initially agreed upon. This could lead to additional features being added that may not be feasible within our timeline or resources.

  - Mitigation: Schedule regular check-ins to review project progress and discuss any new insights or feedback received. This will allow the team to evaluate whether these insights align with the original project goals or if adjustments are needed.

- Risk: While the feedback from the visually impaired student is invaluable, there may be conflicting needs among different users within the visually impaired community. Prioritizing which features to implement based on this feedback could be difficult.

  - Mitigation: Conduct interviews with a diverse range of visually impaired individuals to gather a broader spectrum of feedback. This includes users with varying degrees of impairment, different age groups, and diverse experiences with technology.
  - Organize and categorize feedback into themes, such as accessibility, usability, and functionality. This helps identify which features are universally desired and which are more specific to individual preferences.

A.25 Week Report 2 (09/20/2024)

What We Did This Past Week
This week, we added a Team Declaration chapter to our project document, finalizing our team name, listing all team members, and outlining the project proposal. Additionally, we completed a detailed mission statement, defined the key drivers behind our project, and outlined the key constraints. We also set up a GitHub repository and a Kanban board to assign specific tasks to each team member and effectively track the project's status and progress. Finally, we interviewed a potential client which gave us further insight into the issue that we are attempting to solve and a clearer direction on our product.

What We Will Do Next Week
Next week, we will begin working on our development plan, which includes finalizing roles and responsibilities for each team member, such as Development Lead, Testers, Documentation Lead, and more. We will also create a detailed software development method, outlining the software tools we will use (languages, operating systems, code conventions), hardware requirements, backup plan, review process, build plan, and modification request process. Additionally, we will finalize our workspaces, establish communication and meeting plans, create project timelines, and define our testing policy to ensure smooth project progression. There is also a Demo 1 presentation in class on October 3rd that we will begin brainstorming and preparing for. We will prepare a 20-minute presentation to cover our overall project mission, key drivers, key constraints, and our overall development plan.

List of Action Items:

- Finalize Roles and Responsibilities
- Develop Software Development Method
- Finalize Workspaces and Communication Plans
- Set Timelines
- Define Testing Policy
- Prepare for Demo 1 Presentation (October 3rd)

Any Issues and Risks (and what's being done about them):

- Risk: One of the key issues we encountered was the scope of our project being too large, particularly with the development of a glove component (that would replace the traditional cane for visually impaired users). This aspect of the project presents significant challenges due to the lack of access to specialized hardware and our team's limited experience with hardware development. After consulting with the professor, we recognized that trying to complete this part of the project within our current resources and time frame would be difficult. As a result, we are considering to narrow the scope and build on a similar project from a previous year to ensure feasibility as a backup.
  - Mitigation: This adjustment mitigates the risk of overextending our resources and allows us to focus on areas where we can make meaningful progress, while still delivering a functional and impactful solution for our users. We will need to remain mindful of the revised project scope and timeline to avoid similar issues in the future.

A.26 Week Report 1 (09/12/2024)

What We Did This Past Week
This past week, we completed the finalization of our team members, ensuring that everyone is enrolled in the same class sections. We also solidified our senior design project idea and set up the necessary tools for collaboration. This included creating and sharing the Overleaf document and GitHub repository with all team members. Additionally, we finalized our communication channels to ensure efficient coordination throughout the project.

What We Will Do Next Week
Next week, we will focus on adding a Team Declaration chapter to our project document. This will include defining our team name, listing all team members, and providing a few paragraphs outlining our project proposal. We will also begin working towards our first project milestone, which will focus on 3 key components: the mission statement, key drivers, and key constraints.

List of Action Items:

- Create team name
- Outline detailed project proposal
- Define mission statement, key drivers, and key constraints
- Finalize GitHub Repository; setting up specific tasks for each team member and a Kanban board

Any Issues and Risks (and what's being done about them):

- Risk: Skillset Alignment and Utilization - It may take time to fully understand each team member's diverse technical and business skills, leading to delays in task assignment or inefficient use of individual strengths.
  - Mitigation: Discuss everyone's skills early on to identify each member's strengths and areas of expertise. Assign tasks based on those strengths, ensuring that both technical and business skills are leveraged effectively. Regular team meetings can help adjust task prioritization based on team capabilities and project needs.

- Risk: Project Scope - The project could become too ambitious, with certain features (e.g., Deep SLAM or real-time app integration) extending beyond the available time or resources).
  - Mitigation: Prioritize key deliverables and break the project into manageable phases. Focus on a minimal viable product (MVP) first, then expand functionality.

Appendix B

Improving Senior Design

– Neeti Mistry

Senior Design is a crucial capstone experience that bridges the gap between academic learning and real-world software engineering practices. While it provides valuable hands-on experience, there are some areas where improvements could enhance the learning experience for future students. Below, we discuss aspects that could be improved at both the Software Engineering course level and the Institute level.

Improvements at the Software Engineering Course Level

1. More Structured Guidance on Project Scope: Many teams struggle with defining a project scope that is both challenging and achievable within the given time frame. Providing structured workshops or templates for scope definition early in the semester (or even the previous year) would help teams set realistic goals.

2. Early Introduction to Agile and Project Management Tools: While students are encouraged to manage their projects using Agile methodologies, there is often a learning curve in implementing them effectively. More guidance on industry-standard tools such as Jira or GitHub, or even introducing new tools, could improve project tracking and efficiency.

3. Better Integration with Industry Best Practices: Introducing formal code review sessions, DevOps practices, or test-driven development (TDD) could give students a more industry-like experience. Encouraging teams to maintain better documentation and testing practices could also be beneficial.

Improvements at the Institute Level

1. More Interdisciplinary Collaboration: Senior Design projects often benefit from expertise beyond software engineering, such as business, UX design, or cybersecurity. Creating a better structure that allows cross-department collaboration would improve project quality and better reflect real-world development environments.

2. Earlier Exposure to Senior Design: Instead of introducing Senior Design logistics only at the start of senior year, students should gradually learn about it throughout their undergraduate studies. In freshman and sophomore years, brief overviews of the process could be integrated into introductory courses. In junior year, students should begin forming teams and selecting potential project topics. This would reduce the stress of figuring out logistics in the first few weeks of senior year and allow for more thoughtful project planning.

3. Stronger Industry Partnerships: Partnering with industry professionals for mentorship, feedback, and potential sponsorships would enrich the learning experience. Guest lectures, networking sessions, and hands-on workshops could bridge the gap between academia and industry expectations.

4. A Formal Knowledge-Sharing Platform: Many students struggle with technical roadblocks that past teams have already solved. A centralized knowledge base or archive of past projects, challenges, and solutions could help new teams avoid reinventing the wheel.

While Senior Design provides an excellent opportunity to apply skills in a team-based project, incorporating these improvements could make the experience even more valuable. More structured support, interdisciplinary collaboration, and real-world alignment would help students gain deeper insights and better prepare them for industry challenges.

Appendix C

Individual Learning Reflections

– Ahmad Shah, Neeti Mistry, Sara Gaber, Sohan Chatterjee

Each team member gained valuable skills and experience throughout the Senior Design project, spanning technical abilities, soft skills, and project management expertise. Below is a summary of the key takeaways for each participant.

Neeti Mistry

- Technical Skills: Learn the Swift programming language, gained stronger skills to develop UML diagrams, and refined debugging and testing techniques.
- Soft Skills: Strengthened public speaking and presentation skills, enhanced ability to collaborate effectively in a team, and improved adaptability when facing unexpected challenges.
- Project Management Skills: Learned Agile development principles, improved task delegation and sprint planning, and gained experience in documenting progress and tracking milestones.

Sara Gaber

- Technical Skills: Strengthened proficiency in Swift and Solidworks, contributed to UI/UX prototyping and front-end development, and assisted in integrating haptic feedback systems into the application.
- Soft Skills: Gained confidence in cross-functional communication, improved critical thinking during collaborative design reviews, and demonstrated strong time management while balancing competing priorities.
- Project Management Skills: Supported task tracking in Jira, and contributed to weekly team updates, maintaining transparency and accountability.

Sohan Chatterjee

- Technical Skills: Understand the structuring of the Swift programming language, Apple-specific software such as ARKit, and usage of Raspberry Pi OS. Ensure following of standard UML notation in all software designs.
- Soft Skills: Reinforce presentation skills, prioritize own communication and reliability as a team member.

- Project Management Skills: Familiarize self with Jira, enhance LaTeX documentation abilities, and practice consistent organization.

Ahmad Shah

- Technical Skills: Developed and integrated systems using Swift, ARKit, and Raspberry Pi OS, learning to efficiently process LiDAR data for real-time obstacle avoidance with haptic feedback. Improved my skills in modular hardware-software communication and the documentation of architectural diagrams.
- Soft Skills: Improved communication through public presentation and user interviews, and practiced adaptability under strict constraints (optimization, and hardware limitations).
- Project Management Skills: Applied Agile Methodologies using Jira for sprint planning and assignment delegations, and considered risk management and stakeholder feedback for iterative development.

Bibliography

[1] T. C. Lighthouse. (2025) How do blind and visually impaired people get around? [Online]. Available: https://chicagolighthouse.org/sandys-view/getting-around/

[2] (2025) Uml class diagram tutorial. [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

[3] A. Inc. (2025) Tracking geographic locations in ar. [Online]. Available: https://developer.apple.com/documentation/arkit/tracking-geographic-locations-in-ar

[4] ——. (2025) Understanding world tracking. [Online]. Available: https://developer.apple.com/documentation/arkit/understanding-world-tracking

[5] (2025) Swift programming language. [Online]. Available: https://developer.apple.com/swift/

[6] (2024) Cognitive assistance with lidar localization (c-all) github repository. [Online]. Available: https://github.com/AhmadShah-1/C_ALL

Glossary

4+1 View Model A software architecture model that organizes a system's design into five interrelated views: Logical, Development, Process, Physical, and Scenarios. It helps stakeholders understand and communicate different aspects of the architecture. 45

Accessibility The design and implementation of products, devices, services, or environments that ensure they can be used by all individuals, including those with disabilities. Our project prioritizes accessibility in both hardware and software components. 26

Activity Diagram A type of UML diagram that models the workflow or the sequence of activities in a system. It illustrates the flow of control between various activities and actions, often used to represent business processes or user interactions. 50

Agile A collaborative, iterative approach to software development and project management that emphasizes flexibility, incremental progress, and responsiveness to change. It focuses on delivering small, functional parts of a project frequently to enhance adaptability and user satisfaction. 47

ARKit Apple's augmented reality (AR) framework for iOS that enables developers to create immersive AR experiences by combining device motion tracking, camera scene capture, and advanced computer vision analysis. 64

Assistive Technology Devices or systems designed to support individuals with disabilities in performing tasks that might otherwise be difficult or impossible. In this project, assistive technology includes systems that help visually impaired users navigate their surroundings. 26

CI/CD (Continuous Integration and Continuous Deployment) Continuous Integration and Continuous Deployment; a development practice where code changes are automatically built, tested, and deployed, ensuring faster and more reliable software delivery. 92

Class Diagram A type of UML (Unified Modeling Language) diagram that represents the structure of a system by showing its classes, attributes, methods, and the relationships between objects. It is used in software engineering to model static aspects of a system. 48

Component Diagram A type of UML (Unified Modeling Language) diagram that models the physical components of a system, such as software modules or hardware elements, and their relationships. It is used to show how different parts of a system interact and depend on each other. 58

CoreBluetooth An Apple framework that provides support for communicating with Bluetooth Low Energy (BLE) devices. It enables iOS and macOS apps to discover, connect to, and interact with BLE peripherals. 70

CouldHave This defines the third highest priority requirement.The system could implement all of the tasks, requirements, or anything that is marked this way, but if resources are limited, it can be left out of the current and next version. Build in two versions from now. 27, 29

CRC Cards Class-responsibility-collaboration cards to describe the purpose of each class and direct relationship with other classes. 49

Deployment Architecture Diagram A diagram that illustrates the physical deployment of software components on hardware resources, showing how the system's components are distributed across servers, devices, and networks. 59

Depth Map A representation of the distance between the surfaces of objects in a scene and a specific viewpoint, typically captured by sensors like LiDAR or stereo cameras. 64

Development View A perspective in the 4+1 View Model that focuses on the system's software architecture from the perspective of developers. It shows the system's structure in terms of code organization, modules, and components, helping to manage the development process and ensure maintainability. 58

GitHub A platform for version control and collaboration, allowing developers to store, manage, and share code. It uses Git, a distributed version control system, to track changes and manage different versions of a project. GitHub is widely used in software development for managing open-source and private repositories, facilitating collaboration among team members, and providing tools for bug tracking, feature requests, and project management. 103

Innovation Expo An event at the end of the year where the project will be showcased to demonstrate the system's progress and final results. It serves as an opportunity for senior students at Stevens to present the developed prototype, share the outcomes of the project, and gather additional feedback. 67

Iterative Testing A cyclical process of testing, feedback, and refinement used to gradually improve a product over time. 87

LiDAR (Light Detection and Ranging) A remote sensing technology that uses laser light to measure distances and create precise three-dimensional information about the surrounding environment. LiDAR sensors are a critical component of our navigation system. 26

Logical View A perspective in the 4+1 View Model that focuses on the functional requirements of the system, representing its key abstractions, components, and their relationships. It is used to understand and design the static structure of the system. 47

MustHave This defines the first highest priority requirement. All of the tasks, requirements, or anything that is marked this way are build in the current version. 26, 27, 28

Obstacle Detection The process of identifying and locating objects or barriers in the environment that may pose a risk to visually impaired users. Effective obstacle detection is vital for ensuring safe navigation. 26

Paper Prototype A low-fidelity, hand-drawn representation of a user interface used to quickly visualize and test design concepts. It allows teams to explore ideas, gather feedback, and iterate on designs early in the development process without investing in detailed digital tools. 43

Physical View A perspective in the 4+1 View Model that focuses on the system's hardware and physical deployment. It represents the distribution of components across different hardware nodes, including servers, devices, and networks, to ensure scalability, performance, and fault tolerance. 59

Process View A perspective in the 4+1 View Model that focuses on the dynamic aspects of the system, including concurrency, communication, and runtime behavior. It addresses how the system's processes interact and perform under various scenarios. 49

Raspberry Pi A small, affordable, single-board computer developed by the Raspberry Pi Foundation, that is widely used for embedded systems, robotics, and IoT applications due to its versatility, low power consumption, and extensive community support. 82

RealityKit An Apple framework used for building augmented reality (AR) experiences. RealityKit offers photorealistic rendering, physics simulation, spatial audio, and animation tools for immersive AR development. 70

Seeing AI A free mobile app developed by Microsoft that uses artificial intelligence to describe people, text, and objects to visually impaired users. 90

Sequence Diagram A type of UML (Unified Modeling Language) diagram that models the interaction between objects or components in a system over time. It illustrates the sequence of messages exchanged between objects, showing the order of operations in a dynamic scenario. 57

ShouldHave This defines the second highest priority requirement. The system should implement all of the tasks, requirements, or anything that is marked this way, but if resources are limited, it can be left out of the current version. Build in next version. 27, 28, 29

sighted guide A technique where a person with vision assists a visually impaired individual in navigating physical environments. 90

SolidWorks A computer-aided design (CAD) software used for creating 3D models, assemblies, and engineering designs. It is commonly used for prototyping, simulation, and product development. 63

Swift A powerful and intuitive programming language developed by Apple for iOS, macOS, watchOS, and tvOS development. Swift emphasizes safety, performance, and modern syntax. 69

TestFlight Apple's official beta testing platform that allows developers to distribute iOS apps to testers before they are released on the App Store. It enables secure app sharing, feedback collection, and performance tracking without requiring full public deployment. 92

Think-Aloud Protocol A research method where users verbalize their thoughts, feelings, and reasoning while interacting with a product or system. 88

Usability Testing A technique used to evaluate a product or system by testing it with real users. This process helps identify areas for improvement in user interface design, functionality, and overall user experience. 62

User-Centered Design A design philosophy that prioritizes the needs, preferences, and limitations of end users at every stage of the development process. 89

User-Friendly Design A design philosophy that focuses on creating systems that are intuitive and responsive to the needs, preferences, and feedback of end-users. This project applies user-centered design principles by involving visually impaired users throughout the development process. 67

User Stories Short, simple descriptions of a feature or functionality from the perspective of the end user, typically written in the format: "As a [user role], I want [goal] so that [reason]." They are used in agile development to capture requirements and guide design and implementation. 45

User Personas Fictional, research-based profiles representing key segments of a product's target audience. They capture users' demographics, goals, challenges, and behaviors to guide design and development decisions. 40

User Interface (UI) The means by which users interact with a system or device. A well-designed UI is crucial for ensuring that visually impaired users can easily understand and control the system. 26

Visually Impaired A term used to describe individuals who have partial or complete loss of vision, impacting their ability to navigate and interact with their environment. This project aims to enhance the mobility and independence of visually impaired individuals. 26

VoiceOver An Apple screen reader feature that reads aloud what appears on screen for users with visual impairments. 90

Index