Project Management Plan

for

Theia

Version 1.3

Prepared by
James Abitria, Isaiah Doan, Evan Glasscock
Michael Hull, Nick Lopez, Osaze Ogieriakhi

October 12, 2025

Table of Contents

1. Overview **1**
1.1. Project Purpose, Objectives, and Success Criteria 1
1.2. Project Deliverables 1
1.3. Assumptions, Dependencies, and Constraints 1
1.4. References 2
1.5. Definitions and Acronyms 2
2. Project Organization **3**
2.1. Process Model 3
2.2. Organizational Structure 3
2.3. Member Responsibilities 4
3. Managerial Process Plans **5**
3.1. Management objectives and priorities 5
3.2. Assumptions, dependencies, and constraints 5
3.3. Risk management 6
4. Technical Process Plans **7**
4.1. Methods, tools, and techniques 7
4.2. Software documentation 7
5. Appendix **9**
5.1 Appendix A: Risk Management Table 9

Revision History

Name	Date	Reason for Changes	Version
Initial draft	Sep 4 2025	initial draft	1.0
Draft 2	Oct 5 2025	updated organizational structure of team	1.1
Draft 3	Oct 11 2025	added documentation timelines and requirements process model	1.2
Final Draft	Oct 12 2025	added additional definitions, references, finalizing for completion	1.3

1. Overview

1.1. Project Purpose, Objectives, and Success Criteria

The goal of this project is to develop a mobile app that assists blind and visually impaired individuals with indoor navigation. The mobile app, named Theia, will be designed to guide users around multiple floors, towards rooms, and dynamically detect obstacles to prevent collisions. Caretakers will set up the initial configuration of the app for the user, such as predetermined routes that the user will regularly use.

The application must be intuitive, as blind/visually impaired individuals cannot view the phone screen, but it should still be possible for them to determine their location, and set a destination they wish to be guided towards without secondary assistance. The application will also account for the user's daily routine to predict where the user needs to go.

The application will work with some form of AR, also known as augmented reality, which will be able to relay information based on what it sees to the user. This allows our application to integrate with other systems using a similar form of AR [2].

The main success criteria is to allow someone who is visually impaired to use our application as a tool for artificial sight, it's also important that it can properly convey instructions to its user using sound, vibrations of the phone, and have proper instructions on the screen for anyone who is configuring the app for another person.

1.2. Project Deliverables

Deliverable	Delivery Date	Delivery Method	Comments
Preliminary Project Plan	9/14/2025	Electronic	First draft of determining requirements for the project
Final Project Plan/Presentation	10/12/2025	Electronic	Revisions with refined scope of requirements
Project Tech Demo	12/1/2025	Electronic	Presentation of a mock of the application, with the basic features

1.3. Assumptions, Dependencies, and Constraints

This document is the preliminary plan for the project, representing the ideas and expectations present in the initial stages of its development. This project plan focuses on the requirements gathering aspect of the project, as it provides the best plan of action moving forward. The goal is

to emphasize the importance of finalizing good requirements to ensure the success of the Theia project before moving on to the development phase.

The primary constraint surrounding the Theia project is the team's lack of connections to representative users, or individuals/groups that have created similar software. None of the team members experiences life-altering visual impairment or blindness, nor does the team know individuals that live with visual impairment. As such, eliciting requirements primarily comes from speculation performed by the team to imagine attempting to navigate indoors while blind, performing online research, and experimenting with utilizing blindfolds and noting observations. Because this type of software is not widely utilized, planning features and determining the complexity of said features will be sourced primarily from online research.

1.4. References

[1] S.K. Pal. "Agile Development Moels - Software Engineering." GeeksForGeeks. https://www.geeksforgeeks.org/software-engineering/software-engineering-agile-development-models/ (accessed Oct 1, 2025).

[2] McGraw Hill Education. "Augmented Reality App for Education." McGraw Hill Education. https://www.mheducation.com/prek-12/program/microsites/mcgraw-hill-ar.html (accessed Oct 1, 2025).

[3] Microsoft. "C# Guide." Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/csharp/ (accessed Oct 1, 2025)..

[4] Unity Technologies. "Unity Engine: 2D and 3D Development Platform." Unity. https://unity.com/products/unity-engine (accessed Oct 1, 2025).

[5] P. Loucopolous and V. Karakostas, *System Requirements Engineering*, London, England: McGraw Hill Publishing, 1995.

1.5. Definitions and Acronyms

Agile — A project management approach which allows flexibility, iteration, and collaboration. Project work is done in *sprints* in which the project is iterated, feedback is acquired, and a plan for the next development period is formed.

AS-IS/TO-BE Scenarios — A term for addressing a current problem, by describing a specific, "AS-IS" scenario where the problem causes issues for the user, and speculating how the project will remedy the given problem with a "TO-BE" scenario describing a feature that aides the user in a way that solves the original problem.

AR (augmented reality) — augmented reality relies on set symbols in the environment to center the user so the computer/program knows where they are as accurately as possible.

C# — An object oriented programming language, developed by Microsoft. It is a versatile language used in a variety of different applications.

Discord — An online messaging platform, similar to GroupMe or WhatsApp. Users can create

group chats on the application to communicate with users about topics, or create servers with multiple text channels to better organize discussions.

Documentation — A record of the plans/implementations related to a given software. Documentation can refer to the requirements gathering stage like a WRS document, or it can refer to README files that describe how a system was implemented, and additionally how to utilize the system.

Git — A version control software commonly used by developers to collaborate on a singular project. The software allows for multiple contributors to create copies of the current project, make/propose changes, and store changes on an external database.

GitHub — An online Git repository hosting platform that has widespread use throughout the software development industry. It also has integrated project planning tools including issue tracking, Kanban boards, and more.

IDE — Integrated Development Environment. A collection of tools used for programming and developing an application.

Kanban — A visual workflow management interface where project tasks are arranged on a board into categories, typically based on the task's status.

Requirement — A criterion the software must meet in order to address a specific real-world problem that the software is being created to solve.

Theia — The name of the mobile application being designed for the project, intended to navigate visually impaired individuals indoors, provide descriptive directions, detect incoming obstacles, and options for emergency services.

Unity — A real-time 2D and 3D rendering engine, often used for creating games and simulations. It supports C# as its primary programming language, and allows for easy setup of visual applications with its usage of GameObjects, Components, and Scripts.

WRS — A concept of the combination of the World, Requirements, and Specification with regards to requirements gathering for software. The World represents the environment the project is going to be deployed in. The requirements are the criteria the software should meet. The specifications are how the features of the software will be built to meet the requirements.

2. Project Organization

2.1. Process Model

To gather the requirements for Theia, our team has decided to follow the process of requirements elicitation, specification, and validation process model as described in Chapter 2 of Loucopolous and Karakostas' *System Requirements Engineering* [5]. Our team will gather needs that Theia must fulfill through elicitation techniques, such as performing experiments to put ourselves in user's shoes, researching about similar products online, and reaching out to representative users if possible. The goal of elicitation will be to gather as many possible needs for the app as possible, to have plenty of information for requirements specification.

Once the key needs of the users are identified, the requirements can be specified and broken down into the main features for Theia, where each feature covers a single requirement, or in some cases will cover multiple related requirements. These requirements are developed and recorded in a WRS document, which delineates everything about the domain, possible issues, and transforms them into functional requirements for the project. Another form of requirements specification is the development of AS-IS/TO-BE scenarios, which provide a high-level description of a current issue that users face, and the scenario in which the app solves that issue. The scenarios can also guide further analysis into who it caters to, how it would be implemented, and what else should be factored into that scenario.

After specifying the requirements that Theia must meet, the last stage of requirements gathering will be validating the requirements. Beyond initial reviews performed by the team members to agree on the direction that the project will go, the team will also consult with stakeholders. Our main stakeholder is our project mentor, scheduling checkup meetings to ensure we are on the right track for the project, as well as presenting our key AS-IS/TO-BE scenarios to ensure the main aspects of Theia are being met. If possible, we will also review the requirements we create with representative users, and gather opinions on whether the requirements make sense, and if the planned features will help users.

2.2. Organizational Structure

Due to the smaller team size, all team members work as developers, documentation authors, and documentation editors. In Section 2.3, the specific tasks that the members were assigned to complete will be specifically laid out. Therefore, this section is a general overview of any additional roles that specific members will perform, beyond creating/reviewing requirements documents or creating implementations.

James Abitria serves as the primary communication liaison. While any member can request to schedule team meetings or coordinate client-developer meetings, James gathers information regarding the availability of the developers and clients, and communicates any necessary meeting dates to the clients.

James also serves as the team leader. As team leader, James reviews all the required materials for the project, and assigns specific tasks to team members, including the requirements documents described in Section 2.3. James sends notices to team members about upcoming tasks, and occasional progress check-ups.

Evan Glasscock serves as the meeting scribe. Evan ensures that the discussions from each meeting are compiled in a single document, noting the day, the main topics discussed, and any decisions that were made over the course of the day's meeting.

2.3. Member Responsibilities

This section focuses on the specific requirements documents that each member was assigned to complete. The major requirements documents for the Theia project include this Preliminary Plan, a **WRS document**, meeting records, and a presentation discussing the **AS-IS TO-BE scenarios** of the Theia project.

Each document has one or two members assigned to author the documentation, with each having an initial completion date of October 6th. Over the following week, different team members were assigned to review/edit the document to ensure requirements were met, and for grammatical proofreading, completed on October 12th. This system ensures that team members gain a greater understanding of the project by both writing and reviewing multiple different documents that delineate the plan for the Theia project.

Project Responsibility	**Documentation Author(s)**	**Documentation Reviewers**
Preliminary Plan Revisions	James Abitria	Isaiah Doan, Evan Glasscock, Michael Hull, Osaze Ogieriakhi, Nicholas Lopez
WRS Document	Michael Hull, Osaze Ogieriakhi	James Abitria, Evan Glasscock
AS-IS TO-BE Presentation	Isaiah Doan, Nicholas Lopez	James Abitria, Michael Hull, Osaze Ogieriakhi
Meeting Records	Evan Glasscock	Isaiah Doan, Nicholas Lopez

3. Managerial Process Plans

3.1. Management objectives and priorities

The management approach for this project is built around clear communication, structured task tracking, and accountability among all team members. To ensure progress is steady and transparent, the project will be divided into well-defined tasks with achievable deliverables.

Primary communication and coordination will occur through Discord, which will serve as the central channel for team discussions, announcements, and real-time collaboration. Project tasks, progress tracking, and issue reporting will be managed through GitHub Issues, as the project repository is hosted on GitHub. Each task or feature will be logged as an issue, with assignments and deadlines made visible to all team members to promote accountability and transparency.

Accountability will be reinforced by regular check-ins during team meetings, where members will provide progress updates, discuss blockers, and adjust priorities as needed. The team recognizes that success depends on timely task completion and active participation from all members. Deliverables will be reviewed incrementally, with an emphasis on adapting the scope and timeline based on lessons learned during early stages of development.

Effective collaboration with sponsors, mentors, and potential users will be prioritized. Design and development decisions will be informed by user needs, particularly accessibility requirements critical for visually impaired stakeholders. Communication with external stakeholders will be coordinated through a designated liaison, ensuring information flows efficiently without loss of context.

3.2. Assumptions, dependencies, and constraints

The management of this project is based on several key assumptions, dependencies, and constraints that guide how the team operates and delivers results:

Assumptions

- All team members will have consistent access to Discord and GitHub to ensure smooth communication and task management.
- Team members will maintain active participation and complete assigned tasks by their deadlines, with transparency about progress and blockers.
- The project will continue to evolve as user needs and accessibility requirements become clearer during research and prototyping.
- Team members will have access to all the necessary requirements documents and the ability to view/edit them at any time.

Dependencies

- The project depends on GitHub as the central repository and issue-tracking system for version control and task management.
- The team will depend on timely feedback from instructors, mentors, and (if available) visually impaired users to validate usability and accessibility.

Constraints

- The team is unfamiliar with creating projects related to **AR**, and will need to learn the limitations and possible features for **AR** applications through research.
- The team does not have first-hand experience as visually impaired individuals, nor does the team work as caretakers for visually impaired individuals, therefore the team will need to do independent research, and make educated guesses as to the user's experiences being blind.
- Communication with external stakeholders (sponsors, mentors, or potential users) may be limited by availability, and will therefore be routed through a designated liaison to avoid delays or miscommunication.

3.3. Risk management

The project faces several potential risks that may impact its success. One concern is the misuse of the GitHub repository, which could occur through incorrect Git usage or overwriting important code. Another significant risk is the possibility of failing to meet deliverable deadlines, given the tight schedule and varying workloads among team members. The team's limited experience with advanced app features, such as obstacle detection and real-time indoor navigation, may also pose technical challenges. Effective communication is another potential risk; if members do not consistently check Discord or GitHub Issues, misalignment and delays may arise. There is also the managerial risk of inconsistent commitment from team members, which could slow progress and reduce quality. Finally, accessibility design itself presents a challenge, as ensuring usability for blind and visually impaired users requires specialized approaches and careful adherence to accessibility standards. A table of risks is available in Appendix A.

4. Technical Process Plans

4.1. Methods, tools, and techniques

Our team will be utilizing the **Agile** process model to help develop this project [1]. At the end of every sprint, we will write a sprint recap explaining what changes were added and what objectives need to be addressed in the next sprint. This is to ensure we keep on track with project goals within a timely manner, and have written history to refer back to in case objectives are not met within the time requirement.

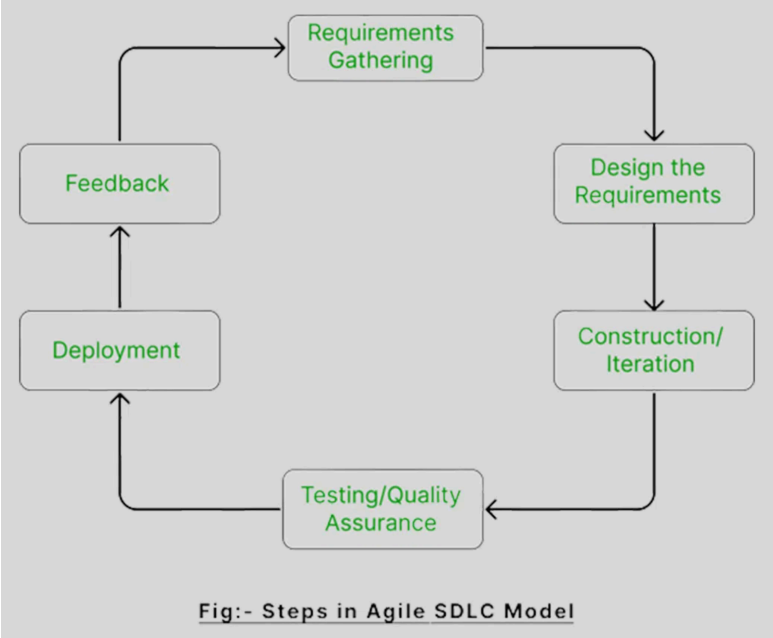

Fig:- Steps in Agile SDLC Model

This model will also allow us to split into teams and develop more than one part at a time. This way during our testing and quality assurance phase we can merge the two new parts together and make sure there are no conflictions. The team can then come back together and implement the new additions into the main branch where we will all give feedback and determine our next sprint.

C# will be our programming language of choice for this project, as we will be using the Unity game engine for creating the app [3][4]. Development will occur using the standard Unity editor and the Visual Studio IDE. Our development methodology for this project will follow agile methodology as outlined previously. For communication our team will be using Discord to communicate. We will also be using GitHub for source control and keeping track of our project deliverables, utilizing the Kanban board to keep track of issues and deliverables within the project. Documentation of project requirements, reports, and team coordination plans will be done in Google Docs for easy collaboration. See the Software Documentation section for information on build-specific documentation.

Since Unity is cross-platform, the app should be able to work on both Android and iOS mobile devices, provided these devices have a camera, modern processing features, and OS software

updates at least at the standard for mid-range phones released in the last 5 years. The application will be fully self-contained, with all location determination and routing happening on-device. The plotting or programming of routes will also be able to be accomplished on-device by a human route planner capable of seeing and placing location marking tags such as AprilTags.

The software will be tested at the unit level where possible with unit tests, as well as quality tested by team members for each release build. Code will be reviewed by peers before being implemented into the application. Details of the testing approaches will be provided in a separate Test Plan.

4.2. Software documentation

As described in Member Responsibilities, the documentation for the project is comprised of the following deliverables:

- Preliminary Project Plan
- WRS Document
- AS-IS TO-BE Scenarios Presentation
- Meeting Records

Each document was originally given a deadline of October 6th to make the changes necessary, and allow for roughly a week to review the documents before being distributed. However, as time passed, it became clear that the documents required more significant edits than initially anticipated, and as a result, the timelines for their completion were pushed back. In the table below, each of the major deliverables has their finalized timeline of their completion, along with a record of all contributors, reviewers, and distribution method(s).

Document	Template or Standard	Created By	Reviewed By	Target Date	Distribution
Preliminary Project Plan	Project Management Plan by Karl Wiegers	James Abitria, Isaiah Doan, Michael Hull, Evan Glasscock, Nicholas Lopez, Osaze Ogieriakhi	James Abitria, Isaiah Doan, Michael Hull, Evan Glasscock, Nicholas Lopez, Osaze Ogieriakhi	October 12th, 2025	GitHub
WRS Document	WRS Sample Document	Michael Hull, Osaze Ogieriakhi	James Abitria, Evan Glasscock	October 12th, 2025	GitHub
AS-IS TO-BE Scenarios Presentation	Theia AS-IS TO-BE Template	Isaiah Doan, Nicholas Lopez	Michael Hull, Osaze Ogieriakhi	October 8th, 2025	Zoom presentation, GitHub
Meeting Records	N/A	Evan Glasscock	Isaiah Doan, Nicholas Lopez	October 8th, 2025	GitHub

5. Appendix

5.1 Appendix A: Risk Management Table

No.	Risk	Type	Likelihood	Description
1	Misuse of GitHub repository	Technical	Unlikely – High potential impact	Incorrect Git usage, accidental deletion, or lack of proper branching may result in loss of code or overwriting important contributions.
2	Failure to meet deliverable deadlines	Managerial	Likely – High potential impact	Team may miss milestones due to scope underestimation, workload imbalance, or time conflicts.
3	Limited team skill set for advanced features	Technical	Likely – Medium potential impact	Some advanced app features (e.g., obstacle detection, real-time indoor navigation) may be too complex given time and expertise constraints.
4	Lack of consistent communication	Managerial	Possible – High potential impact	If team members do not consistently check Discord or GitHub Issues, miscommunication and delays may occur.