Development Plan
Software Eng

Team #11, Mac-AR
Student 1 Matthew Collard
Student 2 Sam Gorman
Student 3 Ethan Kannampuzha
Student 4 Kieran Gara

Table 1: Revision History

Date	**Developer(s)**	**Change**
2023/09/15	All	Initial addition of all sections
2023/09/23	Ethan	Updated team member roles chart (need to fix formatting), touched up workflow plan, technology, and project scheduling
2023/09/25	All	Final commit for Development plan deliverable
2023/11/14	All	Finalize PoC Plan
2023/11/14	Ethan	Prof instances updated to Professor. Acronym table added. Grammar issues fixed. Fixed inconsistent use of master vs main in workflow plan, as well as added information about use of git issues and tags.

This document outlines the rules and processes that the team will follow during the development process. All information about how the team will communicate and coordinate, along with what each team member is responsible for overseeing, can be found here.

1 Symbols, Abbreviations and Acronyms

symbol	description
UI	User Interface
NFR	Non-functional Requirement
PoC	Proof of Concept
VS	Visual Studio

2 Team Meeting Plan

Our plan is to have a meeting for every capstone lecture that is not for Software Engineering or is not running. When we are behind on a deliverable, we will schedule a meeting during our time between classes to meet and go over what we will need to accomplish before the due date.

3 Team Communication Plan

For most issues, we will communicate over Discord with each other. Matthew is in charge of communicating with the professors/supervisor. In addition, we will also have a meeting with Irene Yuan every week during Irene Yuan's Office hours.

4 Team Member Roles

The following responsibilities are required of each team member:

- Each team member will contribute to full-stack development.
- All team members will be present at team meetings, however, if a team member cannot attend for some reason they must let the team know at least an hour beforehand and give a reason.
- If a team member would like to change roles, they must discuss with the group so that a suitable swapping of roles can be done with another team member

Team member	Role(s)	Responsibilities
Matthew Collard	Team Liaison	Communicates with the Supervisor/Professor and any stakeholders
	Front-End Lead Developer	Make sure user interface is clear to the user, and functioning properly. Makes sure we are on track to complete UI goals on time
Sam Gorman	GitHub Administrator	Merge and maintain GitHub branches. Responsible for merging dev into main branch
	Non-Functional Requirement Lead	Makes sure every NFR is implemented in the design
Kieran Gara	Final Revision Editor	Last editor of project documents. Makes sure team adheres to writing guidelines
	Back-End Lead Developer	Leads development process and makes sure team is on-track to complete implementation goals on time
Ethan Kannampuzha	Functional Requirement Lead	Makes sure every functional requirement is implemented in the design
	Meeting Minute Writer	Keeps track and writes down meeting minutes

5 Workflow Plan

- For Git, we will have a master branch called main that will always have a working code base and up-to-date documentation. We will have a second branch called dev, this is the development branch, and all our new changes get merged into this branch. It is meant to be unstable and most of our issues will pop up in this branch. Every time dev has a stable build, we will push the code into main. We will have branches off of dev that we are going to use as our feature branches, every new line of code gets written in

these branches, and when the feature is done it will get merged into dev. This double buffer system ensures we always have working code easily accessible in main. Sam will be responsible for merging dev into main. Continuing on, we will also be using Git issues for issue tracking. Team members will create and assign issues depending on the work that needs to be done. Additionally, tags will be used to differentiate the subject matter that the issue falls under.

- Document changes are the only changes that can be merged directly from branch to main. All source code changes go through the development branch first.
- Changes into the dev and main branches will be done through formal pull requests. Reviewers will not be strictly necessary for these PRs, as this is mainly being used as a way to track merges and make sure branches are up to date before merging in. Reviews from other team members are recommended when working on the same file/directory.
- GitHub issues will be created for each team meeting, lecture period, document revision, and feature change. Additionally, issues for meetings will be used to track who attended and a general overview of the agenda. GitHub project boards will be used to keep track of deliverable deadlines and GitHub issues for document and dev work will be broken up by subtasks for each deliverable. These project boards and issues will be the main project management tools used for the project.

6 Proof of Concept Demonstration Plan

The main risks of the project are:

- Implementing AR elements into the game
- Networking between multiple devices

To demonstrate that these risks can be overcome, the following will be shown in the PoC Demo:

- Show that a digital object can be overlayed on top of a real-life surface
- Demo an example of at least two users being able to send and receive chat messages
- Demo an example of at least two users being able to connect to a live audio chat
- Demo at least two users being able to interact with the same AR element and have it be affected on both devices

If our PoC shows that our AR implementation is not working, then the entire scope of the project might be too large, and we would have to reduce the scope as AR implementation might not be as simple as we thought it would be. This would allow us to focus more time on these core components as opposed to adding more features. If the networking between multiple devices is not working, we would have to reconsider how the users can cooperate with each other to play the game. We could reduce our scope to only text messages as a final implementation instead of voice and text messages.

7 Technology

- Coding Language: C# (Unity) Unity has built-in AR modules we could use to implement our project more easily.
- Linter: Microsoft Code Analysis extension in VS. Unity has built-in Visual Studio support, so using a built-in linter for Visual Studio is very convenient for our group
- Test Framework: NUnit. Nunit has great support for the .net(C#) unit testing and can automate unit testing for us.
- Code Coverage measuring tool: dotCover. DotCover integrates with VS and is built for C#, so it is perfect for our usages, and it has specific support just for Unity projects.
- Specific plans for Continuous Integration (CI): Will be doing extensive unit testing as well as code linting. Hard to have full CI/CD in a Unity game engine
- Libraries: To be expanded based on project needs
  - C# standard library
- Tool/Framework: Unity

8 Coding Standard

- Variable name syntax: camelCase
- Function name syntax: PascalCase
- Class name syntax: PascalCase
- Brackets: Same line (Ex. If () {)
- Public/ Private: Use private when possible
- Indent: Tab

9 Project Scheduling

The team will use a Google Calendar to schedule goals for milestone progress and deliverable deadlines. Additionally, GitHub project boards will be used to keep track of ongoing deliverables, and GitHub issues will be used to assign sub-tasks for these deliverables. for As a general rule, the next deliverable will be started a minimum of one week before the previous deliverable is due for documentation. Code development will be ongoing throughout with deadlines set for various components.