Development Plan
Software Engineering

Team #2, Campus Connections
Waseef Nayeem
Zihao Du
Matthew Miller
Firas Elayan
Abhiram Neelamraju
Michael Kim

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept 25	All	Revision 0
Nov 13	Zihao Du	Update POC demo plan
Jan 4	Zihao Du	Revision 1: Add winter term meeting schedule and Acronym table
Jan 4	Zihao Du	Revision 1: Update technology we use; Add reasons for using Firebase real-time database

This document outlines the development plan for CampusConnections, an app designed for better social connections on campus. This development plan includes plans for team meetings and team communication, team member roles and the workflow plan. Furthermore, it discusses the technologies and coding standard that the team will be using and the plan for the proof of concept demo.

Table 2: Symbols, Abbreviations, and Acronyms

symbol	**description**
TA	Teaching Assistant
AR	Augmented Reality
IDE	Integrated development environment
JHE	John Hodgins Engineering Building

1 Team Meeting Plan

The team will have weekly meetings at 3:30 PM every Monday (fall term) or 2:30 PM every Monday (winter term) virtually on our Microsoft Teams group. The purpose of these meetings is to align the team around current deliverables, exchange ideas and thoughts, and assign tasks to team members. In addition to weekly meetings, the team will have ad hoc meetings if necessary. Formal meetings with our supervisor, Dr. Irene Ye Yuan will also be held every Thursday from 3:30 PM - 4:30 PM. All the meetings listed above will be held by our meeting chair and have meeting minutes created as GitHub issues. Our meeting scribe will be responsible for creating meeting minutes which include the meeting date and time, agenda, team actions and meeting notes. All team members must join weekly team meetings and meetings with our supervisor in person. If a member is absent or can only join virtually, they must notify the rest of the team half an hour before the meeting starts. If a member is absent from a meeting, he must reach out to other members and read the meeting notes to understand what happened in that meeting. If most of the team agrees to postpone a meeting, they are responsible for informing the rest of the team and finding another time that works for everyone.

2 Team Communication Plan

The team will communicate primarily through Microsoft Teams, emails and GitHub issues. Microsoft Teams group chat will be mainly used for asking and answering questions, scheduling meetings and sharing resources related to the project. Teams meetings will be used for weekly meetings and supervisor meetings. The team will use emails for any formal requests sent to the professor, supervisors and TAs. Any emails sent must have all teammates CC'd on it. For code-related issues and task assigning, GitHub issues will be used. The team has set up a Kanban board to track all tasks assigned to everyone. Members are responsible to update these issues and report their progress in weekly meetings.

3 Team Member Roles

Every member will occupy multiple roles in order to build such a big project. As the project consists of mainly three parts of development, our team will have three different kinds of roles focusing on these technical challenges. Members assigned these roles are required to have a deeper understanding of that specific topic and potential technical challenges. These roles will be rotated to further enhance our understanding in each area of the project and all work done will be logged and discussed in team meetings. Additionally, if there are unforeseen circumstances or potential roadblocks hindering our performance, we may consider switching roles to allow for smoother team operation and gain new perspectives for the issue. These will be discussed in team meetings and justified accordingly.

Other than development roles, there will be other roles assigned to everyone in order to organize the team. The following table shows all team member roles.

Table 3: Specific Member Roles

Team Member	**Role(s)**	**Responsibilities**
Firas Elayan	Server Developer	Work on the server-based aspect of the design
	GitHub Expert	Review coming PRs and manage feature branches
Abhiram Neelamraju	Server Developer	Work on the server-based aspect of the design
	Lead Tester	Lead the testing process and generate test reports
Zihao Du	User Interaction Developer	Work on the interactions between users and the environment
	Meeting Scribe	Take meeting minutes and book meetings for the team
Michael Kim	User Interaction Developer	Work on the interactions between users and the environment
	Meeting Chair	Organize all meetings and create meeting agendas
Matthew Miller	User Communication Developers	Work on the interactions between users
	Issue tracker	Track all GitHub issues, manage the Kanban board and assign tasks
Waseef Nayeem	User Communication Developers	Work on the interactions between users
	Game Engine Expert	Deal with bugs in the development process and answer questions of other members

Though everyone has their own roles, they still need to have a basic understanding of all parts and join all team meetings, otherwise it will be quite risky for the project if something happens to one expert.

4 Workflow Plan

The following steps will be used for workflow:

1. Pull new changes from the main branch

2. Create a new branch from the main branch to develop on
    - **NOTE:** Never develop on the main branch
    - **Branch naming convention:** Branch Name should start with the following descriptive keywords: **chores/docs/feat/fix**, followed by brief descriptions linked with **-**
    - **Branch naming example:** *chores-update-readme*
3. Develop new code, as well as tests for that code if applicable
4. Commit new code with descriptive messages (every member should use a pre-commit hook)
    - **Commit message convention:** Commit message should start with the following descriptive keywords: **chores/docs/feat/fix**, followed related issue number, wrapped with **():**. After that, write a brief description of the commit
    - **Commit Message example:** *chores(#2): update readme*
5. Push those changes to the branch you created
6. Merge the branch you created into the main branch via a pull/merge request
    - **NOTE:** Make sure all the tests pass before merging

The following tags will be used:

- **rev0:** Use for revision 0 of the project (default tag if the documentation does not specify a version tag)
- **rev1:** Use for revision 1 of the project

The following labels will be used for issue tracking:

- **documentation:** Use when documentation needs to be updated
- **meeting:** Use when meeting minutes need to be added
- **feat-enhancement:** Use when a new feature or request needs to be added
- **fix-bug:** Use when there is a bug in the code that needs to be fixed
- **chores:** Use when it is not a feature or bug or documentation related issue
- **Review Tracker:** Use when there are peer reviews to track
- **lecture:** Use when there is a lecture
- **TA-feedback:** Use when the team gets feedback from TA's marking
- **Review-Team10:** Use for team 10 members when they reviewing our work

5 Proof of Concept Demonstration Plan

Our proof of concept will be mobile apps and Unity Emulator running on Android OS and consists of three demonstration standing for different functionalities of the application: AR feature, friends system and location tracking.

- **Friends System** We are aiming for the user to be able to make an account and test out the main navigational, informational and social aspects of our application. This will be integrated in the future, the panel is present but the product is not linked into a single application yet. Currently, the account is automatically generated on start-up. They will be able to add other users as friends and delete existing friends.
- **AR Feature** Users will then be able to use the application to come in proximity to a specific building of our choosing (e.g. JHE) and recognize the surroundings. They will see a display about some sample AR objects taking place at the location. These object should be able to recognize user actions like touching. Augmented Reality should be available on both indoor or outdoor areas as far as the they are added to the database.
- **Location Tracking** We are also expecting the user can get the real-time location information and display that along with a campus map. This will be essential for core functionalities like location sharing and building recognition(a back up plan when AR recognition does not work).

One main risk in our project is our ability to successfully implement Augmented Reality when looking at buildings around campus etc. to increase the engagement of the user. Since it is not clear how feasible of a goal this is, we want to be able to produce the same functional result using just the location information of the user. We will do so as mentioned earlier and this proof of concept will allow us to move forward knowing we can depend on the location model if we are not able to seamlessly integrate AR into our app.
Another risk is the user communication system in Unity. Though there exists a friend service in Unity, it is not clear how it can fulfil all our requirements. This proof of concept will show us if that is a feasible solution or we need to move on with our own server and database.

6 Technology

- Programming languages - C#
- Linter tool - Jetbrains Rider inbuilt Linter for C#
- Testing framework - Unity Test Framework
- Code coverage tools- Test runner - code coverage package in Unity
- Performance measuring tools - Unity Internal Profiler

- Libraries - Unity in C#, Registration page library (e.g. Strapi), Firebase Library, Map Library Mapbox
- Tools - Unity, Vuforia, Firebase realtime database, Firebase Authentication, ASP.NET

Justification:
Currently we plan on using Unity completely for our Front-end. Unity has several plugins and libraries we can use to create an elegant user interface such as Mapbox for location sharing etc. Unity along with Vuforia will also be the main tool we use for developing the AR aspects of our app such as the AR sign/navigator in the building. It uses C# to write scripts to implement logic between components. We will be using the Rider IDE to write the C# code which also has an inbuilt Linter. Unity also comes with several helpful tools that we will be making use of to write better code. Specifically, we will be using the Unity Test Framework ("Play Mode Test") and Code Coverage package to thoroughly test our code and track coverage respectively. We will then be using the Unity Internal Profiler to measure performance of our application. For our Back-End we plan on creating a server in C# with ASP.NET. It will handle real-time location sharing and chatting. The user data will be stored using Firebase real-time database (a non-relational database) since it can store large amount of data and synchronized in almost real-time to all client.

7 Coding Standard

As mentioned earlier we will be using the Rider IDE from JetBrains for our C# code in unity. The Rider IDE has its own C# code style that we will be complying with and following. The reference is as follows:
https://www.jetbrains.com/help/rider/Settings_Code_Style_CSHARP.html

8 Project Scheduling

Scheduling for the project is centered around important dates in its timeline. The calender feature on Microsoft Teams will be used to keep track of deadlines as well as our team meetings and meetings with our supervisor, Dr. Yuan. The team will receive reminders from the calender tool which will help the team ask their work rate appropriately to ensure we're not falling behind schedule. The deadlines for the project deliverables will be used as a measure of the progress done, with the plan being to begin work on the next deliverable before the upcoming deliverable's deadline.