Development Plan
Sayyara

Team 31
SFWRENG 4G06
Christopher Andrade
Alyssa Tunney
Kai Zhu
Ethan Vince-Budan
Collin Kan
Harsh Gupta

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 24, 2022	Alyssa Tunney	Updated meeting plan, communication plan
September 25, 2022	Kai Zhu	Added Workflow Plan
September 25, 2022	Christopher Andrade	Added Technology and Coding Standards joint section
September 25, 2022	Collin Kan	Added Proof of Concept Demonstration Plan
September 26, 2022	Harsh Gupta	Added and edited risks, technology and project scheduling
September 26, 2022	Alyssa Tunney	Modification of team member roles, misc. document cleanup
September 26, 2022	Collin Kan	Added POC Demo Plan
September 26, 2022	Kai Zhu	Edit pass for consistency

1 Team Meeting Plan

The team will meet hold weekly in-person meetings every Friday at 2:30PM. The agenda for each meeting will be determined throughout the week based on current and upcoming deliverable and development tasks. The meetings are estimated to take between 30 minutes to 1 hour, with option to extend if necessary. Where applicable, meeting minutes will be recorded by Alyssa Tunney.

2 Team Communication Plan

The primary method of internal team communication is through the Discord application. Discord will also be used for asynchronous communication with the team's supervisor, Nabeel Ibrahim. Online meetings with Nabeel will take place on Google Meet. Development issues such as bugs and features, as well as discussions regarding these topics will be tracked on the GitHub issues board for traceability and ease of maintenance.

3 Team Member Roles

In addition to the roles below, each team member will be responsible for contributing to the topics for meeting agendas during the team's weekly meetings. Each team member is also expected to contribute to the documentation and deliverable coursework.

Christopher Andrade

- Team liaison, responsible for team communications such as emailing, meeting scheduling, and meeting notes
- Testing lead

Alyssa Tunney

- Responsible for recording meeting minutes
- Documentation lead

Kai Zhu

- Back-end development lead
- Responsible for desktop application testing

Ethan Vince-Budan

- Database development lead
- Responsible for mobile application testing

Collin Kan

- System design lead
- Responsible for managing and organizing issues on GitHub

Harsh Gupta

- Meeting chair
- Front-end development lead

4 Workflow Plan

The team will follow a feature branch workflow:

- Issues are created for new features based on requirements using the git issue board.
- Bug fixes, documentation-specific tasks such as proofreading, sectional-editing are also categorized and tracked as issues.
- Issues are assigned to team members based on preference or expertise.
- For every new feature or bug fix, a new branch is created from a pull of the latest main branch so that no change is pushed directly to the main branch.
- The new feature or bug fix is implemented along with documentation for each module and function.
- Commits should be small (such as after implementing a self-contained function).
- Before a pull request, the branch should merge from the main branch to resolve merge conflict.
- The developer(s) of the branch are responsible for performing unit tests prior to merging.
- Pull requests are created upon the completion of the feature with description of the changes, and linked to the corresponding issues for clarity.
- Pull requests must pass automatic unit testing using GitHub actions through CI/CD.
- Pull requests are reviewed and approved by another team member, and the corresponding issues are marked as resolved.

5 Proof of Concept Demonstration Plan

5.1 Main Risks

5.1.1 Implementation

The primary difficulty of implementing this project is the scope of the project. The MVP version of the web app consists of three main components, each containing numerous different views and components. There is a concern that the scope of the project potentially exceeds the amount of time available to implement it. In addition, designing and implementing numerous architectural components such as file and database structures, back-end API, and front-end user interface add to the challenge of ensuring that all components work together seamlessly.

Since the team's experience focus primarily in engineering rather than UX design, creating an intuitive user interface may prove challenging, especially when a clean, clutter-free, and navigable interface is one of the primary requests from the client.

5.1.2 Testing

The application, as described by the client, contains multiple user types and interactions. Consequently, it will likely contain a large number possible states, such that creating and running unit test cases to cover every state of the system can be challenging. In addition, as mentioned previously, important but qualitative aspects of the system such as the intuitiveness of the user interface are not measurable using automated tests. Manual testing of these features may require outside users, which can be time consuming.

Furthermore, automating end-to-end tests over the numerous possible user interactions will be a monumental undertaking. Additional care must also be taken to managing and synchronizing the tests with feature delivery to avoiding the risks of potentially breaking the CI/CD system.

The scalability of the system is a minor risk. During the development phase, the client will only provide simulated data. Without a real active user base, testing the system's performance and tolerance may be difficult.

5.2 Risk Mitigation

The main indicator of risk mitigation in the application will be the CI/CD history and the number of consecutive successful deliveries to the main branch. We expect to use Docker to further reduce the risk of dependency on specific hardware and software environment for the development, testing, and deployment stages.

5.3 Demo Plan

As a proof of concept, we will identify and implement at least one most important feature for each business event in each of the three main components, and ensure that the components interact and communicate with each other. The primary focus will be on these core logic and communication, featuring a minimal skeletal UI for demo purposes. This prototype will provide finer insights into a realistic, achievable scope for the MVP. Upon successful implementation of the POC, additional features can be added and the user interface refined.

6 Technology and Coding Standards

The team will develope a full-stack application using a JavaScript front-end and Python back-end. NextJS, using ReactJS as its base, will serve as the main front-end framework to ensure web and mobile compatibility. The back-end will use Supabase to provide a Postgres database, authentication, and instant APIs. The StandardJS linter will be used to enforce front-end source code compliance with the JavaScript Standard Style. For the Python back-end, PyLint will be used to help follow the PEP8 coding standards. Unit testing will be conducted using Jest and Pytest frameworks for Javascript and Python respectively. These testing tools also have extensions to provide code coverage measurement with LCOV reports. Using GitLab CI/CD, the project will also use Continuous Integration with pipelines for testing and building before merge requests are accepted to a master branch. Lastly, Docker may also be used to setup the project for further building, testing, and deployment. Throughout the early stages of development, some minor changes in the technologies may occur if required.

7 Project Scheduling

The project schedule will be managed and tracked using GitHub issues, in addition to online and in-person meetings. The project management feature in GitHub will be used as the Kanban and brainstorming boards. Each requirement will be decomposed into planing and implementation steps. The first major milestone of the project is to create a functional web app setup such that CI/CD can be deployed as a starting point for developers to deliver their features. The other major milestones involve delivering specific features for the given requirements, with the POC completed by November 15, 2022 and revision 0 version delivered by February 6, 2023.