Development Plan
Sayyara

Team 3, Tiny Coders
Arkin Modi
Joy Xiao
Leon So
Timothy Choy

Table 1: Revision History

Date	Developer(s)	Change
September 23, 2022	Timothy Choy	Add roles and responsibilities
September 23, 2022	Arkin Modi	Create project scheduling section
September 23, 2022	Arkin Modi	Create workflow plan
September 23, 2022	Arkin Modi	Create technology and code standard sections
September 25, 2022	Timothy Choy	Create team meeting plan
September 25, 2022	Timothy Choy	Create team communication plan
September 25, 2022	Timothy Choy	Create proof of concept demo plan
September 26, 2022	Timothy Choy	Add introductory blurb
March 8, 2023	Joy Xiao	Change component library
March 18, 2023	Arkin Modi	Update weekly meeting times
March 26, 2023	Arkin Modi	Add static analysis and code documentation tools

This document outlines the team's development plan moving forward. This document will include the plans for meeting, and communications between team members and the capstone project supervisor. Furthermore, this document also includes each members' roles and responsibilities. On the technical side, this document will state the technologies used, workflow plan, coding standards and what Tiny Coders will be demonstrating in the proof of concept demo.

1 Team Meeting Plan

The team will have weekly team meetings every Friday at 10:00am for the fall semester, and every Wednesday at 2:30pm for the winter semester. These meetings will be held on Discord. The purpose of these meetings are to realign the team to the current deliverables and to assign tasks to each team member. An agenda of every team meeting will be written as a GitHub issue, and tagged with the meeting number and meeting date. Each meeting will be archived and can be easily accessed by searching for issues with the "meeting notes" tag. Additional meetings with the supervisor will be held on an as-needed basis.

2 Team Communication Plan

The team will communicate primarily through Discord, Messenger, and GitHub Issues. Discord and Messenger will be used for quick communication between the team members, with the added benefit that the supervisor will also use Discord. Team meetings will also be hosted on Discord. For general communication, organization, and job delegation, GitHub Issues will be used. The team has set up GitHub Projects to further organize issues into categories such as "Todo", "In Progress", "Review/QA", and "Done".

3 Team Member Roles and Responsibilities

Every member will have the following responsibilities:

- Contribute to full stack development, not just components the member is leading
- Write tests for their code
- Contribute to relevant documentation
- Code reviews through approvals in pull requests
- Follow guidelines set for updating the main branch, stated in CONTRIBUTING.md
- Follow the development workflow
- Attend meetings, or giving adequate time to let the team know if they cannot attend
- Communicate with the supervisor of the project

Furthermore, each member has specific roles as shown in the table below. Though the team has assigned specific people to each role, they are subject to change and can be moved around from member to member.

Table 2: Specific Member Roles

Team Member	**Role(s)**	**Responsibilities**
Arkin Modi	Point of Contact	The liaison between the professor/TAs and the team
	Issue Tracker Manager	Manages issue tracker, keeps the issues organized
	Database, Deployment & CI/CD Lead	Manages the database component of the application, as well as the deployment and CI/CD process
Joy Xiao	Front End Lead	Manages the front end components of the application
	Note Taker	Takes meeting minutes for meetings, if necessary
Leon So	Server Lead	Manages the server components of the application
	Note Taker	Takes meeting minutes for meetings, if necessary
Timothy Choy	Meeting Chair	Organizes team meetings and creates meeting agendas
	Front End Lead	Manages the front end components of the application

4 Workflow Plan

4.1 Development Workflow

The team will be using Git as their version control system and the repository will be hosted publicly on GitHub. The most stable up-to-date version of the application will be on the “main” branch. The team will update the main branch using a feature branch workflow. The feature branch workflow entails the developer to:

1. Create a new feature branch
2. Push changes to the new branch
3. Open a pull request against the main branch
4. Link the pull request to an issue (if appropriate)

5. Add “needs review” label (if ready for review)
6. Assign assignees and reviewers
7. Pass any pull request pipelines in the CI/CD setup
8. Receive reviews from other team members
9. Address any comments or change requests
10. Merge or close the pull request

To prevent naming conflicts, the team will follow a branch naming convention of “<GitHub username>/<branch name>”. The merge strategy used for pull requests will be “squash and merge”. Using this merge strategy increases traceability on the main branch’s commit history by having every commit represented by a working/passing pull request and prevents adding broken commits to the main branch’s commit history.

All team members are listed as code owners of everything in the repository (defined in .github/CODEOWNERS). GitHub will read this and automatically assigned all team members to every pull request opened in the repository. Every pull request should receive, at minimum, one review before merging.

4.2 Project Management Workflow

The team will be using GitHub’s issue tracker, GitHub Projects, and GitHub Milestones to manage the project. The issue tracker will track work that needs to be done. For each task that isn’t trivial, an issue will be open to track the contribution and progress. Examples of trivial tasks include spelling corrections, dependency version updates, base repository template updates. Each issue will also be assigned to team member who will then assume responsibility for said issue. Issues may also be assigned to a GitHub Milestone. A milestone will represent a deliverable’s deadline and will be an additional way to track progress towards a deadline.

Labels, as described in Table 3, will be used to organize issues. The GitHub Project board will be used to visualize the progress of each issue and track overall progress.

Table 3: Issue Labels

Name	**Description**
documentation	These issues are related to documentation or reports for course deliverables.
Epic	These issues are tracking a collection of other issues. For example, an issue for a report has sub-issues for each section.
meeting needed	A team meeting is needed to discuss details related to the issue.
meeting notes	This issue contains notes from a completed meeting.
needs breakdown	This issue needs further breakdown into smaller issues/steps.
tech foundation	This issue is not directly contributing towards any course deliverables.

5 Proof of Concept Demonstration Plan

There are several risks that will determine the success or failure of the project. The main risks are:

- Communication between the modules of the PWA
- Responsiveness of the PWA to different devices and screen resolutions
- Peering problems between multiple editors (e.g., race conditions)
- Integration testing
- The team is unfamiliar with PWAs, as this will be the team’s first time working on one

To prove that the team will be able to overcome these risks and have a successful project, the team will be showing several aspects in the proof of concept demo:

- Develop two modules of the PWA to show communication between the modules in the application works

- Demo the application on several devices (e.g., an iPhone, an Android, and on a computer) to show that the application is responsive
- Demo an example of two devices attempting to schedule an appointment at the same time, and see the result

6 Technology

The project will be written in TypeScript using the Next.js framework. Next.js is a React framework that provides both a frontend and a backend interface. The team is comfortable with React, and with both the core Next.js team and the core React team recommending Next.js for "production-ready projects", the team sees this as a good choice. Tiny Coders will be using NPM as the package manager and the packages next-auth and next-pwa. NextAuth is a package built for Next.js to quickly enable authentication with many out of the box adaptors for providers and databases. Next-PWA is a package to quickly enable the creating of Progressive Web App (PWA)'s in Next.js. The team has decided to create a PWA since this was one of the key application attributes that the supervisor required.

For the user interface, the team has chosen to use PrimeReact as the component library alongside CSS. A team member has previous experience with this library and recommended using it.

The team will be using Prisma for ORM (Object-relational mapping) and MySQL as the database management system. The team has decided to use a SQL database due to the data having relations. MySQL was specifically chosen due to past experience within the team. In a NoSQL database, relations are harder to maintain and/or possibly inefficient to query. SQL databases also provide the assurances of ACID compliance, fewer consistency issues, and enforced field constraints. The use of Prisma is due to past experience within the team and a strong reputation within the web development world.

The Jest Testing Framework and React Testing Library will be used for unit tests, integration tests, and code coverage. Most of the team does not have much experience with frontend testing, therefore the team has decided to go with the libraries recommended by Next.js. The team has decided against using any end-to-end testing frameworks (e.g., Cypress, Playwright) due to lack of domain knowledge within the team and time constraints.

Git and GitHub will be used for version control and hosting a centralized online repository. This was chosen due to course requirements and past experience.

Documentation will be written in a mix of LaTeX and Markdown. Code documentation will be written inline with the source code as docstrings and will follow JSDoc annotations. The docstrings will be rendered into HTML documentation using TypeDoc.

With high performance not being a primary goal for this project, the team will not use be using any performance benchmarking tools. The team will be maintaining a Google Lighthouse score of greater than or equal to 90%.

All CI/CD pipelines will be created using GitHub Actions. This was chosen due to GitHub Actions having great integration with GitHub, a generous free and pro tier, past experience within the team, and ease of use.

The development environment for any developer is a highly personal decision and while the team will not be enforcing the use any specific local dev tooling, the team will be recommending to all developers to use Visual Studio Code for easy interoperability between developers. The team will also have a list of recommended extensions and setting for Visual Studio Code available in the ".vscode" folder. The team will also be using Docker for easy setup of external dependencies (e.g., TeX Live, MySQL).

7 Coding Standard

To ensure consistent code styling and code quality, the application code will be formatted using Prettier and ESLint. Prettier is an industry standard code formatter and widely agreed upon default configuration. ESLint is an industry standard static analysis tool for TypeScript.

For LaTeX docs, the team will be using latexindent.pl. latexindent.pl is a popular LaTeX code formatter. The primary use will be for managing whitespace to ensure consistent readability in source files. The

primary reason behind this choice was latexindent.pl's code formatter capabilities to automatically apply fixes.

8 Project Scheduling

The project schedule will primarily follow the course calendar with development of the application starting in October 2022. Progress of the project will be tracked using a GitHub Project board. The deliverables due date will act as milestones and will be tracked with GitHub Milestones.

Decomposition of large tasks into smaller tasks for documentation deliverables will be done by breaking down the reports by section. Sections will be determined by consulting provided templates, rubrics, and examples. Breaking large development tasks into smaller tasks will be done by first researching and scoping out how the task should be architected. Following this initial architecture, smaller tasks will be created.

Responsibility/ownership of tasks will be decided as a team during weekly meetings or offline communications. It is upon each member to communicate their availability and capable workload. Ideally, the work shall be equally distributed over the course of the project, but a shift in workloads on an individual deliverable basis is expected. If a member is unable to complete a task they are assigned, it is their responsibility to communicate this to the team and have the task reassigned. The expectation is each member shall contribute an average of 10 hours per week.