Table 1: Revision History

Date	Developer(s)	Change
September 23, 2022	Arkin Modi	Create project scheduling section
September 23, 2022	Arkin Modi	Create workflow plan
September 25, 2022	Timothy Choy	Create proof of concept demo plan

Development Plan
Software Engineering

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

[Put your introductory blurb here. —SS]

1   Team Meeting Plan

2   Team Communication Plan

3   Team Member Roles

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 our 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 our 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 strat-
egy 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 and GitHub Projects 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.

Labels, as described in Table 2, will used to organize issues. The GitHub Project board will be used to visualize the progress of each issue and track overall progress.

Table 2: 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 issues 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 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

- It is our team's first time working on a PWA so there may be unfamiliarities

To prove that we will be able to overcome these risks and have a successful project, we 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

- Specific programming language

- Specific linter tool (if appropriate)

- Specific unit testing framework

- Investigation of code coverage measuring tools

- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done

- Specific performance measuring tools (like Valgrind), if appropriate

- Libraries you will likely be using?

- Tools you will likely be using?

7 Coding Standard

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 our milestones.

Decomposition of large tasks into smaller tasks for documentation deliverables will 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.