Development Plan
Software Eng 4G06

Team 2, Parnas' Pals
William Lee
Jared Bentvelsen
Bassel Rezkalla
Yuvraj Randhawa
Dimitri Tsampiras
Matthew McCracken

Table 1: Revision History

Date	**Developer(s)**	**Change**
24/09/22	Jared Bentvelsen, Yuvraj Randhawa, Bassel Rezkalla	Initial draft of Problem Statement
30/03/23	Jared Bentvelsen, Yuvraj Randhawa, Bassel Rezkalla	Edits to Properly align with feedback and issues

1 Team meeting plan

Team meetings will be conducted weekly on non-holiday Mondays at 4:30pm. Further meetings can be scheduled if needed. Meetings will be conducted in person at a McMaster bookable room or library unless specified otherwise.
All Team members are expected to attend the Monday meeting unless specified on timely notice.
Meetings will follow the meeting plan (agenda) created by Matthew McCracken. And adjustments to this plan may be recommended by other team members by giving Matthew Notice. It is Matthews responsibility to ensure high priority topics are covered during meetings.

2 Team communication plan

Common team communication and questions will be conducted on **Microsoft Teams**.
Any online meetings will be conducted on **Microsoft Teams**.

3 Workflow plan

GitHub will be used for all workflow and version control. GitHub Actions will also be used for spell checking Latex documents. Husky will be used with pre-commit hooks for linting and formatting code before it is committed and pushed to the repository

3.1 Task Scheduling

The **Github Kanban** project board will be used to plan and assign upcoming and ongoing tasks to team developers.
If a task is large or general, sub-tasks will be created and assigned accordingly.
Upcoming project deliverables must include deadlines.

3.2 Git Branch Usage

All features, documents and file changes require a named branch describing the change. Branches will follow a similar format to:

$$topic\_or\_type/section/branch\_description$$

3.3 Git Commits

Tags will be attached to commits as needed to further display the change description.
Commit Squashing may be used prior to creating pull requests to clean up unnecessary commits.

3.4 Git Branch Merging and Pull Requests

The main or centered branch will be protected such that new features and additions will require a pull request in order to be merged.
Pull Requests require a minimum of 2 approvals to be accepted.

3.5 Technial Issues

For all technical questions, concerns and issues, the **Github Issues** feature will be used to open an issue or concern accordingly.

4 Team Roles

Team Role Descriptions	
Meeting Organizer	William
Front End Expert	Dimitri
Database Experts	Jared, Bassel
UX/UI Expert	Matthew, Dimitri
Networking Protocols Expert	Bassel
Backend Expert	Jared, William
Lead Testers	Yuvi, Bassel
Team Liason	Jared
Backup-Team Liason	William
Git Experts	Yuvi, Matthew
Scrum Leader	William
Mobile Development Expert	Dimitri, Matthew
Latex Expert	Yuvi
Hardware Expert	Bassel

5 Coding Style and Standards

The **Google Coding Style** will be used as a standard for all code in the project. The style guide for this can be found here. A language specific linter will be enforced.

6 Testing

6.1 Unit Testing

Jest will be used for this app's unit testing. Jest offers mocking and code coverage insights which will be useful for unit testing.

6.2 Simulation / Integration Testing

TestCafe will be used for this app's integration testing. TestCafe is a cross-browser integration testing framework that will be used to verify proper integration between the app's various components (back-end, front-end, database, etc). TestCafe also offers end-to-end testing which includes automatic simulation of many common web-app components such as forms and buttons.

7 Proof of Concept Demonstration Plan

The risks associated with the project are:

1. Implementing a workflow for users to add workouts and for other users to discover those workouts. If this interaction does not work, then our application will not fulfill its goals.

2. Building a back-end server capable of interfacing with front-end clients and storing user information in a secure database. This back-end is essential for our application to meet the social networking aspect of our proposed project.

3. Achieving %70 code test coverage. This is a challenge because unit tests alone will not cover most code. View code will need integration / end-to-end tests for full coverage. This will involve our team working with TestCafe which may be time consuming to learn and write.

4. Developing a cohesive UI/UX for our application. All user interactions must be mapped out. We must use group meetings to discuss how each interaction should look and map out user navigation.

5. Using git branches for up to 6 people to do simultaneous development and avoiding pushes to main or unsafe merges. Our group has varying levels of Git experience and so we must work together to ensure smooth development.

During the Proof of Concept Demonstration we will prove that these risks have been overcome by showing:

1. A working demonstration of a test user creating a workout and another test user discovering the created workout. Viewers will be walked through the process of creating their own workout and can see how these workouts are then displayed to the general userbase.

2. A demo of a new user signing up for the application and manipulating their own data. Viewers can then see this new user and their custom data in the database to prove that the back-end and server are cohesive with the frontend.

3. End-to-end / integration tests written in TestCafe that simulate the user experience across the application. This will include testing for user sign-in, workout creation, and workout discovery.

4. Designs created in Figma that describe desired user interactions and portray a cohesive application-wide theme.

5. A conflict-free Git history with even contributions from all members of the team.