Development Plan
SFWRENG 4G06

Team #7, Team FAAM, SweatSmart

Daniel Akselrod
Jonathan Avraham
Sophie Fillion
Sam McDonald

Table 1: Revision History

Revision Version	Date	Developer(s)	Change
0	Sep 25, 2023	Daniel, Sam, Sophie, Jonathan	First draft of Document
0.1	Nov 16, 2023	Daniel, Sam, Sophie, Jonathan	POC plan revision
1	Apr 3, 2024	Sophie	Final documentation changes from TA comments and change from ML to regular algorithm

With more individuals trying to maintain a healthy and active lifestyle, embarking on or maintaining their fitness journey can be daunting and time-consuming. Creating an evidence-based fitness application will revolutionize the way individuals engage with their physical health. SweatSmart aims to help individuals trying to start or maintain their fitness journey.

This document outlines the development plan for the entirety of the project, creating an outline that team members can refer to. Section 1 outlines the schedule and expectations for team meetings. Section 2 lists the various communication methods and their corresponding use for the team. Section 3 specifies team members' roles and their responsibilities. Section 4 depicts the workflow plan for using GitHub. Section 5 describes the Proof Of Concept that the team hopes to complete by the described deadline noted in section 8. Section 6 notes the tech-stack that will be used for the implementation of this project. Section 7 indicates the coding standards that will be practiced throughout this project. Section 8 outlines the overall project schedule/deliverable deadlines.

1 Team Meeting Plan

Team meetings are scheduled semiweekly on Mondays at 3:30 pm and Wednesdays at 1:30 pm, either in person or virtually. All team members are expected to attend said meetings, unless they have given notice to the rest of the team with a valid reason. If the team decides that a team meeting is not necessary, then a meeting can be canceled. However, all team members should keep the semiweekly meeting times free and expect to have the scheduled meeting. During team meetings, members are expected to share progress updates for their assigned tasks, raise any concerning issues, and contribute to team discussions and decisions.

Every team meeting (including lectures) will be recorded using GitHub Issues. An issue will be opened before each meeting, outlining the meeting's agenda and attendance. All team members are expected to

review the agenda beforehand and come to the meeting prepared with progress updates, concerns, and/or questions. At the end of the meeting, the issue will be closed.

The team's project manager will act as the meeting chair, responsible for preparing the agenda, facilitating discussion and keeping the conversation focused and on-task. They will also monitor the length of the meeting, keeping meetings under an hour. Any notes, comments or concerns will be documented as comments under the GitHub Issue, which can be referred to in the future.

2 Team Communication Plan

Daily communication for administrative tasks, questions, concerns, or reminders will be done via the iMessage group chat. All team members are responsible for checking and responding to these messages frequently to inform the team that they are up-to-date with new information.

GitHub project boards will be used to communicate asynchronously about technical work, such as task details, project status updates, or key project documents. GitHub issues will also be used to raise concerns. These issues should be linked to the appropriate assignee(s), label(s), and project(s) on GitHub. All team members are expected to check the project boards and issues daily to stay on top of project tasks and their progress.

As previously mentioned, the team will have semi-weekly meetings, occurring in person or virtually. In-person meetings will take place in Thode Library, unless stated otherwise, while virtual meetings will occur on Discord. These weekly meetings will be used for project check-ins, status updates and deliverable updates.

3 Team Member Roles

On top of the specified roles specified below, all team members are responsible for coding, testing, documentation, and creating/commenting on issues. Additional roles and responsibilities may be added along the course of the project to accommodate and manage all of the tasks.

Team Member	**Role**	**Responsibilities**
Sam	Team liaison	Emailing course instructors/TAs for any questions or concerns, replying to emails received from course instructors/TAs.
	Industry/research expert	Being an expert in the fitness industry, knowing needs/demand for different application features.
Jonny	Full-stack developer expert	Answering questions and providing support for team members regarding UI/UX design, front-end, and back-end development.
Daniel	GIT/CI/Tech-stack expert	Answering questions and providing support for team members regarding the tech-stack, Git, or CI workflows.
Sophie	Project manager	Leading meetings (creating agenda, open meeting, facilitating discussion, monitor conversation), overseeing issues, monitoring project progress through project boards, ensuring the team is staying on track with deadlines and deliverables, providing LaTeX support

4 Workflow Plan

4.1 GitHub Usage

The following plan should be followed for the workflow:

1. GitHub Actions will be used to manage continuous integration workflow scripts

2. Pull changes from main branch
3. Create new branch under main branch (see section 4.1.1 for specific branches)
4. Implement changes/tests to relevant code or documents.
5. Create appropriate issues on Git for project management (see 4.1.2)
6. Push branch changes when needed, upon completion, branch can be merged into feature branch if CI build passes
7. Once build passes and another teammate approves PR, branch can be merged, the remote branch is then deleted
8. Production is then updated to reflect main once the core app is implemented correctly every week

4.1.1 Branch structure

- main → Main dev branch, where PR's are pulled into
- prod → Production branch, updated to mirror main when changes are ready to be deployed
- config → Used for updating project structure, runtime configs and environment
- docs → Used for all project documentation
- feat/... → High level branch per feature
- user/... → A grouping of all team members, will contain 1 branch per issue in the possible format 'user/<first name>/<ticket id>/<issue name>'

4.1.2 Issue Types

- Feature → Used for adding new/extending existing functionality of app
- Fix → Used for fixing bugs
- Refactor → Used for refactoring
- Test → Used for adding unit tests

4.1.3 Pull Requests

- Naming Convention → TBD but will include name of feature
- User branches are merged into feature branches, user branch is then deleted
- 2 approval required before merge, depending on circumstance, CI workflow build must pass

4.1.4 Commits

- Possible Naming Convention → (<Issue Type>)-<description of commit>
- Squash commits where applicable

5 Proof of Concept Demonstration

5.1 Objective

The objective of our Proof of Concept (POC) is to ensure our plan to use an AI* algorithm for workout generation based on user input is feasible and to have an initial working demonstration of our app. The main risk lies in finding sufficient and suitable data to train our ML model, as well as being able to develop and train the model. Our POC demonstration will aim to show that these risks are addressed and are feasible in our project.

*Note: At this stage of the project, we were still thinking of using an ML model for our workout generation algorithm. However, after finding and working with our supervisors, we discovered that it was not the best option and we changed our course of action.

5.2 Scope

The POC demonstration will center on the following aspects:

- **Basic AI algorithm Validation:** Validate that the AI algorithm can predict the strength of an individual given variable parameters (e.g. age, gender, current exercise frequency).
- **Simple UI:** Create a simple user interface for the web application.
- **Simple database:** Create a simple database that stores user profiles and information.

5.3 Algorithm Testing

- Create a limited dataset of 20 or so sample users with only a few parameters (e.g. age, gender, current exercise frequency).
- Run the AI algorithms against this limited dataset to validate the basic functionality of the strength prediction of the user.
- Create a simple proof-of-concept algorithm without extensive optimization.
- Create a basic user profile for each team member.

5.4 Success Criteria

The POC will be considered a success if the following has been demonstrated:

- The AI algorithms can predict the working weight for an exercise for users with limited information about the user.
- A user can create and log into their profile.
- Created profiles are stored in the simple database; profile information can be recovered.

6 Technology

6.1 Programming Languages

- **Front-end:** Typescript
- **Back-end:** C#, SQL

6.2 Linters

- **Front-end:** ESLint, Prettier
- **Back-end:** .NET format

6.3 Unit Testing

- **Front-end:** Jest
- **Back-end:** XUnit

6.4 Test Coverage

- **Front-end:** Jest provides coverage insights with `jest -coverage`.
- **Back-end:** DotCovert integrated with XUnit to provide insights on test coverage.

6.5 Continuous Integration

- **Front-end:** Run ESLint check, Prettier check, and Jest tests.
- **Back-end:** Run .NET format check, and XUnit tests.

6.6 Tools and Libraries

- **Front-end:** React-Native, Expo, Node.JS, Jest, React-Testing Library
- **Back-end:** ASP.Net Core, ML.Net, MySQL, XUnit, Microsoft Azure

6.7 Development Tools

- Jet Brains rider for both front-end and back-end development
- Git for version control, GitHub for hosting remote repository
- OverLeaf for LaTeX documents for all documentation

7 Coding Standard

7.1 Naming Conventions

- All variable names should be meaningful and descriptive.
- Use camelCase for function and variable names (e.g. 'userProfile', 'userPreferences').
- Use SCREAMING_SNAKE_CASE for constants (e.g. 'MAX_SETS', 'MIN_WORKOUT_DURATION').
- Use PascalCase for backend class and functions names (e.g. 'WorkoutPlanner').
- Use PascalCase with 'I' prefix for Interfaces.

7.2 Comments and Documentation

- Include comments for sections of code that are not immediately interpretable.
- Comments should be clear and concise.
- Documentation needed for functions and methods. Documentation should include descriptions, parameters, return values, and usage examples (if applicable).

7.3 Aesthetics/Formatting

- Consistent indentation.
- A single blank line will be used to separate logical sections of code.
- Limit line lengths to 80 characters for best readability.

7.4 Error Handling

- Extensive error handling needed for critical sections of code.
- Use meaningful error messages.
- Keep track of/log errors for debugging.

7.5 General Coding Practices

- Create and implement reusable functions to avoid code duplication.
- Break complex logical operations into smaller, more manageable sections/functions.

7.6 Testing

- We will use Jest for front end testing and XUnit for back end testing.
- Ensure tests are maintained so they are up to date with new code and changes to existing code.

7.7 Version Control

- Commit messages should be descriptive and concise.
- Commit messages should include a reference to a relevant issue or task (if applicable).

8 Project Schedule

Deliverables with steps needed before the final due date will have an earlier date specified, otherwise assume the date in the bolded title is the due date. These dates are subject to change based on the progress of the project but should be used as a baseline guide.

8.1 Project Initiation (Due Sept 25)

- Define scope and objectives (Sept 18)
- Draft Problem Statement
- POC Plan
- Development Plan

8.2 Requirements and Planning (Sept 26 - October 4)

- Research and possibly collaborate with stakeholders to gather and document requirements for the workout app (Sept. 29)
- Define use cases and functional requirements (Oct 1)
- Define non-functional requirements (Oct 1)
- Choose which requirements to prioritize and create a plan for future developments (Oct 3)

8.3 Hazard Analysis (October 5 - 20)

- Identify risks to users and our team (Oct 18)
- Begin thinking about steps to address these risks

8.4 V&V Plan (October 18 - November 3)

- Decide on and identify which types of testing we will conduct (e.g. unit testing) (Oct 29)
- Create criteria for validating the app's functionality, performance, and safety (Nov 2)

8.5 Proof of Concept Demonstration (November 13 - 24)

- Conduct internal testing to ensure all potential issues are addressed (Nov 10)

8.6 Design Document (November 28 - January 17)

- Architecture and system design (January 10)
- User interface and User experience design (January 10)
- Technical specifications (January 14)

8.7 Revision 0 Demonstration (January 7 - (February 5 - February 16))

- Demonstration planning (January 15)
- User testing and feedback (January 22)
- Bug tracking and issue resolution (January 29)

8.8 Validation & Verification Report (Feb 4 - March 6)

- Summary of test methodologies (March 1)
- Results of tests presented, highlighting key areas of both success and failure (March 4)

8.9 Final Demonstration (March 1 - (March 18 - March 29))

- Conduct testing to detect errors and issues (March 8)
- Script drafted (March 14)
- Final demo plan and error fixes (March 16)
- Script finalized and practiced for team rehearsal (March 17)

8.10 EXPO Demonstration (April 9)

- Display materials prepped (April 1)
- Talking points finalized (March 31)
- Demonstration video created for review (April 1)

8.11 Final Documentation (Revision 1) (April 4)

- Problem statement
- Dev plan
- Proof of concept plan
- Requirements document
- Hazard Analysis
- Design documentation
- V&V plan
- V&V report
- User's guide
- Source code