Table 1: Revision History

Date	**Developer(s)**	**Change**
March 30 2023	Jared Bentvelsen	Revision 1

Reflection Report on Software Eng 4G06

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

This document serves to reflect upon the entire development process of the Olympian application. It shall highlight the accomplishments, issues, and lessons learned throughout the eight months of development.

1 Project Overview

The purpose of the Olympian application is to provide a single platform for creating, distributing, consuming, and tracking exercise content within the fitness community, with a focus on ease of use and pleasant user experience.

In the current market, fitness content is extremely popular on general purpose social media platforms such as YouTube, Instagram, and TikTok. Other formats of exercise planning and tracking often involve excel sheets and notes applications. However, content on these platforms is not optimized for fitness content that one wishes to learn and perform themselves. Existing social media exercise content is posted as very short videos, which are often unable to be paused, or replayed halfway through. Spreadsheets of information are difficult to view in exercise settings, and are cumbersome to track progress with.

Olympian seeks to resolve this by focusing the application on creating informative content that can be easily viewed and shared among friends and followers. This is done through a simple UI and intuitive organization. This enables users to create and view programs containing specific workouts (and subsequently specific exercises) to share with their friends and track their own progress towards their specified goals, all within one dedicated application.

2 Key Accomplishments

- Many UI and UX features were implemented successfully, creating a customer experience very close to what was envisioned.
- Data organization (e.g. schema creation) and API implementation with specific tools such as GraphQL and Prisma (see System Design) was successfully done, which allowed front end functionality to be seamlessly hooked up to the back end.
- Most social features such as saving programs created by others, and viewing live follower activity were successfully implemented.
- Documents were finished on time, with relatively equal contribution from all group members.
- Mocking of specific features using Figma was done relatively consistently, which made implementation of UI/UX far easier.
- The group was able to settle into a standard PR process of descriptively tagged GitHub Issues to Figma mockups (if applicable), to implementation, and finally review. This greatly accelerated development and reduced friction between team members.
- CI/CD: GitHub Actions were used to spellcheck documents, and Husky pre-commit hooks performed linting and formatting before code was pushed to the repository. This kept the code base nice and clean.

3 Key Problem Areas

- Several documents were done very close to the deadline, leaving little room for a thorough review process. Some documents lacked detail as a result.
- Time management was not consistent. Often productive work took place in sporadic bursts instead of consistently throughout the year.
- Communication regarding implementation was lacking. There were a few instances of one group member performing work that had (unbeknownst to them), already been done by someone else on an yet to be merged branch. This was improved with more consistent use of GitHub issues, and the above mentioned standardized PR process.
- Code documentation was sparse. React code can be difficult to comment, but not having sufficient code documentation hindered development speed, as one developer had to figure out the work of the other, and oftentimes ask questions to clarify code.

- Issue prioritization was often not aligned with the requirements and goals of the project. Disproportionate amounts of time were spent of visual tweaks that did not have a significant impact on the final design, and could be easily changed later. This improved further into the development process, where a lack of time forced focus onto very important features.

4 What Would you Do Differently Next Time

Summarizing the above problem areas, next time we would focus on more consistent time management and development, with strict GitHub issue use from the start. GitHub issues could also be tagged with a priority (e.g. High, Medium, Low), and time-boxing could be implemented on low priority features to keep development focused and on track, and prevent time wasting on unimportant features. Consistent use of GitHub issues and following proper branch protocols would allow for smoother parallel development.

We also realized too late into development just how large our project scope was. It seems that our eyes were bigger than our keyboards, and we didn't think about how much work each of the features we tagged onto the SRS entailed. Next time, we should spend more time on scope definition and think critically about the core features and how much time we have to develop them.