System Design for Software Eng 4G06

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

March 30, 2023

1 Revision History

Date	Version	Notes
January 13 2023	1.0	Initial Draft
March 30 2023	2.0	Revision 1

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
Software Eng 4G06	Explanation of program name

Contents

List of Tables

List of Figures

3 Introduction

This document serves to illustrate and explain design decisions (in the context of alternatives) and the thought processes and considerations of the team that made said decisions.

4 Purpose

The purpose of this design documentation is to justify design decisions and prove that our final design meets all requirements as specified in the Software Requirements Specification (SRS).

The purpose of this design documentation is to make clear current design choices, and to show why these designs were chosen over possible alternatives. This document also must prove that the current designs fulfill the requirements enumerated in the Software Requirements Specification (SRS) document.

5 Scope

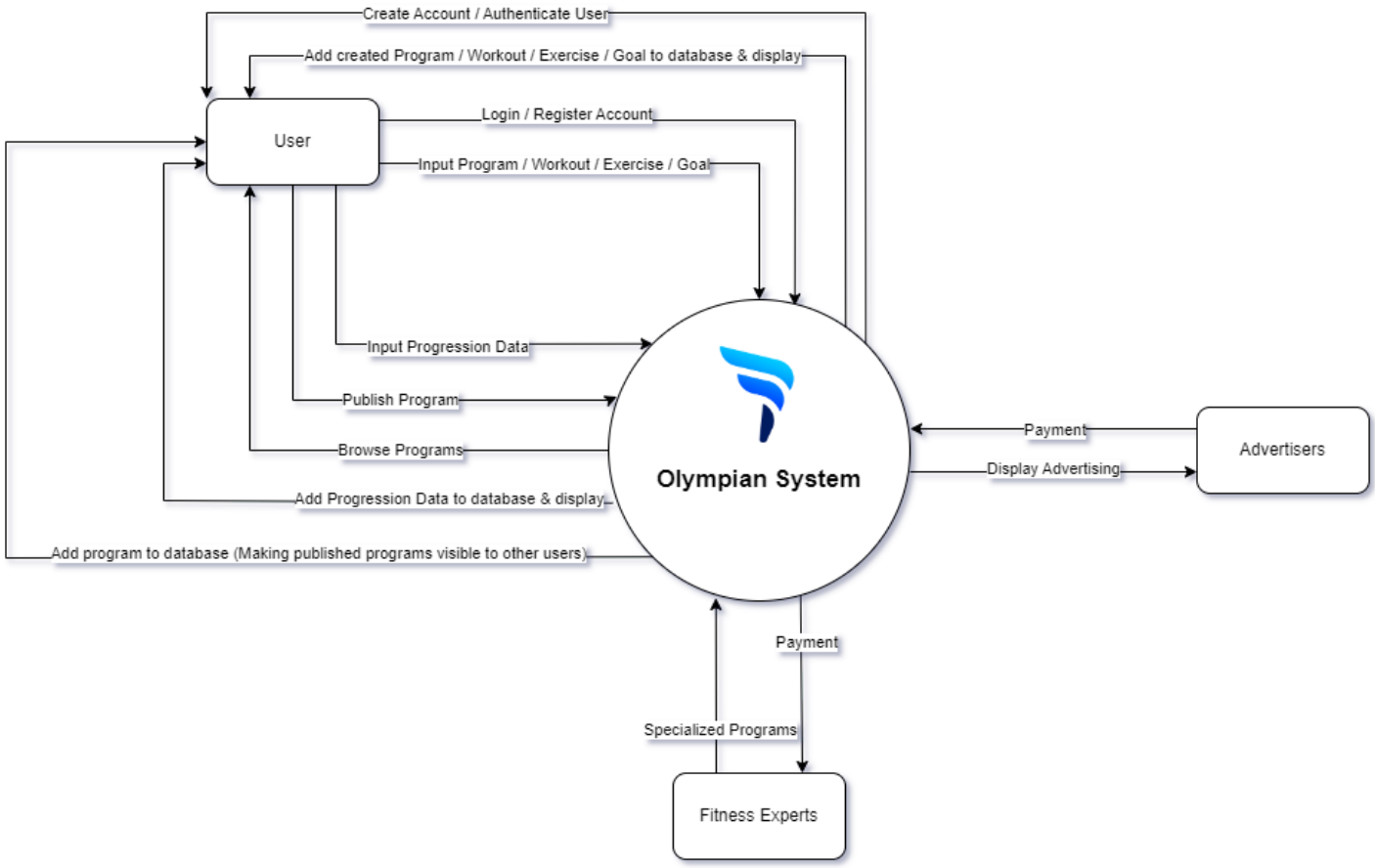

Figure 1: System Context Diagram

6 Project Overview

6.1 Normal Behaviour

Normal behaviour for the application can be defined as the state of the available program when all FRs and NFRs are fulfilled. If an FR or NFR is not fulfilled, this is abnormal behaviour, and indicative of the occurrence of an undesired event.

6.2 Undesired Event Handling

An 'undesired event' can comprise anything from incorrect user input to database failure to connectivity issues.

Invalid User Input: Whenever a user provides incorrect information to a form (e.g. an invalid email address during sign-up), an informative error message is displayed beneath

the incorrectly filled field, and a visual queue is given in the form of highlighting the field in red to indicate error. To prevent annoying the user with error messages before they have actually made an error, forms will only display error messages after the first submit attempt. After an initial submit attempt, the error message will remain until the contents of the field are valid. This is to give the user information as to whether or not their entry is valid before having to submit again.

Connection/Database Failure: If the application is unable to connect to the internet, and subsequently the server and database, an informative error message is displayed, informing the user to verify their internet connection. Note that a stretch goal for the future is to have the Olympian application save certain information locally to enable the user to perform certain functions offline. Changed local information would be synced with the database once connection is re-established. However, this is a stretch goal, and current requirements and designs list the database as the only information store, no data (other than an authentication token) is stored locally. If the database/server itself is failing, an appropriate message is displayed to the user. The message will indicate that the failure is not the fault of the user, but rather an internal failure outside of their control.

Error messages are kept consistent and concise in format, to make errors easy for the user to recognize and act upon.

6.3 Component Diagram

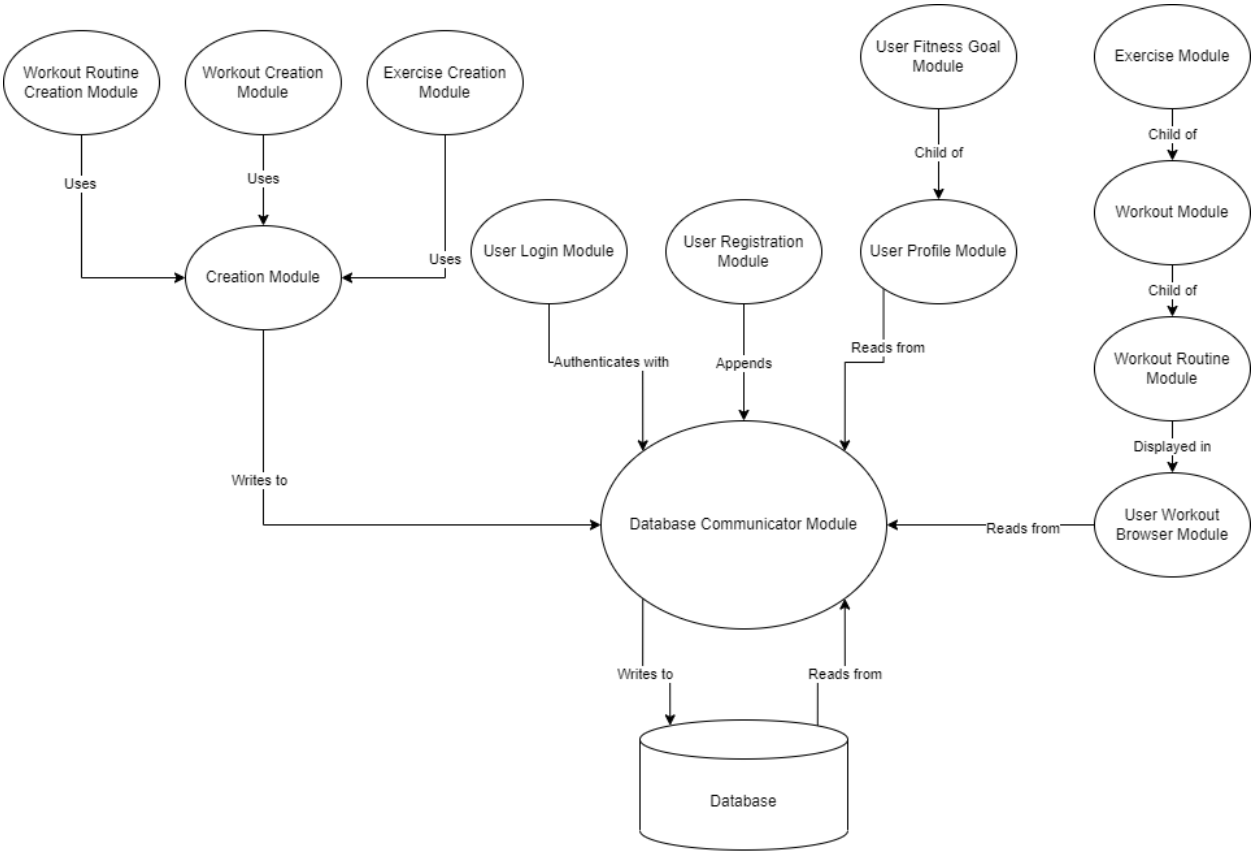

Figure 2: Diagram of Module Interactions

6.4 Connection Between Requirements and Design

Requirement	Design
NFR1 - The product shall appear minimal and straightforward	The application will contain an interface that only presents information that is necessary as to prevent cluttering and promote minimalism. Brief descriptions are used when necessary, with large utilization of symbols and icons.
NFR2 - The product shall use fonts of readable size to the target user group	Fonts of size 12 or larger will be used across the application.
NFR4 - The product shall be able to be used by untrained fitness enthusiasts and amateurs alike, who receive no training before using it	UI will be simplistic and consistent throughout. Specifically, a small set of fonts, colours, and interactive components (e.g. buttons) will be used to minimize the cognitive load for the users to learn and interact with the application.
NFR5: The product shall use symbols and pictures to provide users with an intuitive and efficient experience.	Icons are used in input fields and options to help the user instantly recognize information.
NFR6 - The product shall be usable by users with hearing loss or partial blindness	The application will rely on audio, visual, and haptic cues to indicate correct inputs or application events.
NFR14 - The application must inform users when maintenance is taking place and must warn them at least 1 day in advance	The application will utilize pop up screens to display important notifications to the user.
NFR16 - Passwords must be encrypted with SHA-256 when stored	The system does not store plaintext passwords. Passwords entered by the user (during login or account creation) are instantly hashed (using Argon2) before being sent to the database.
NFR13 - The application shall operate on iOS devices and Android devices.	Olympian is being developed in React Native, which is device agnostic.
NFR20 - The application will allow users to report offensive content and remove it from their feed	There will be a button on each post that gives user the option to report offensive content.

7 System Variables

N/A

8 User Interfaces

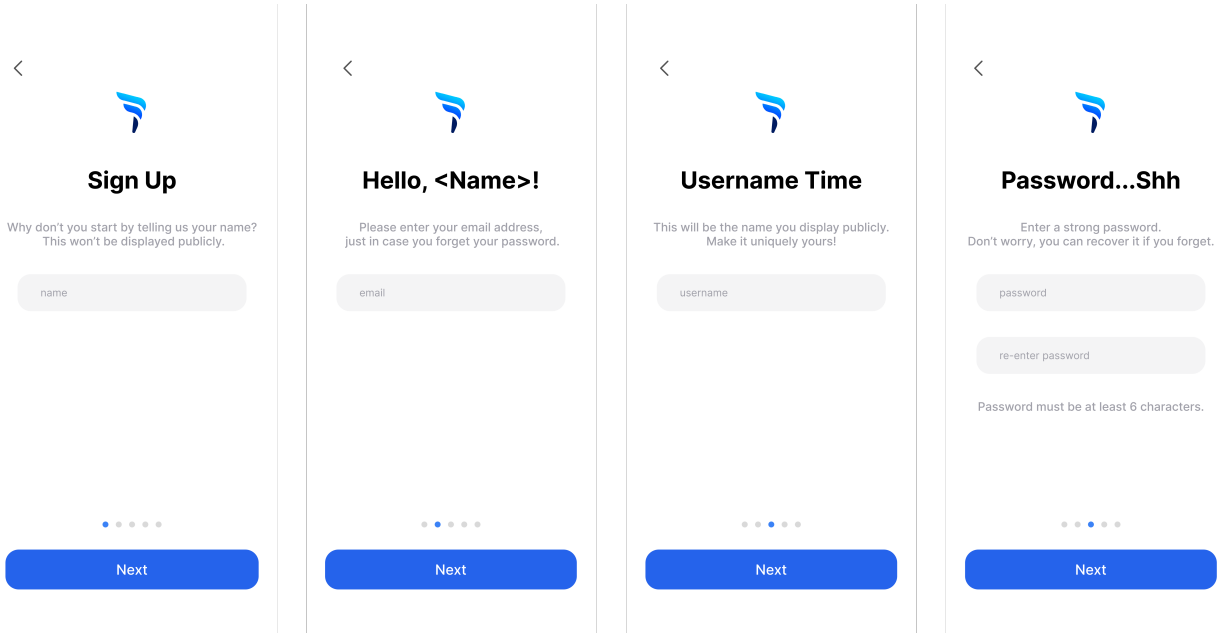

Figure 3: Multi page register form

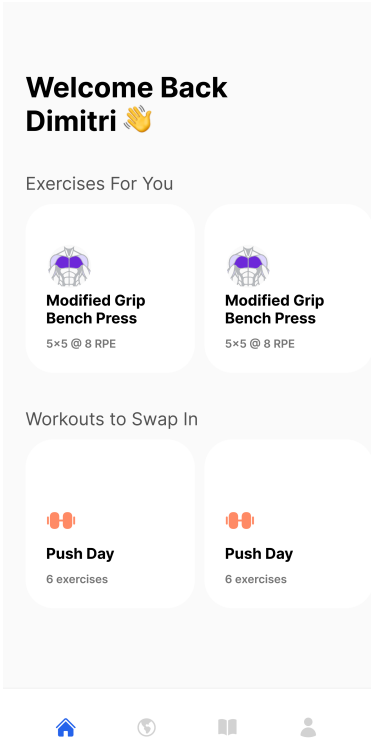

Figure 4: Home Screen (Landing page for logged in users)

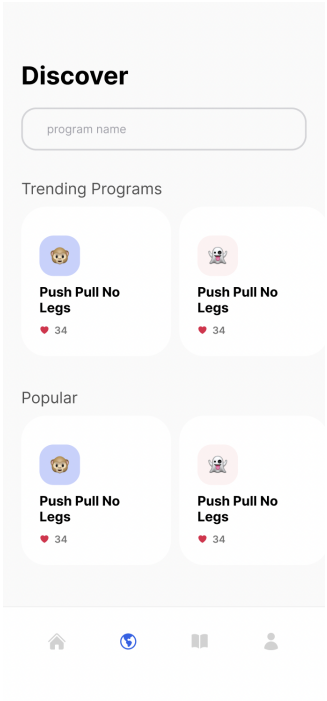

Figure 5: Discover Screen

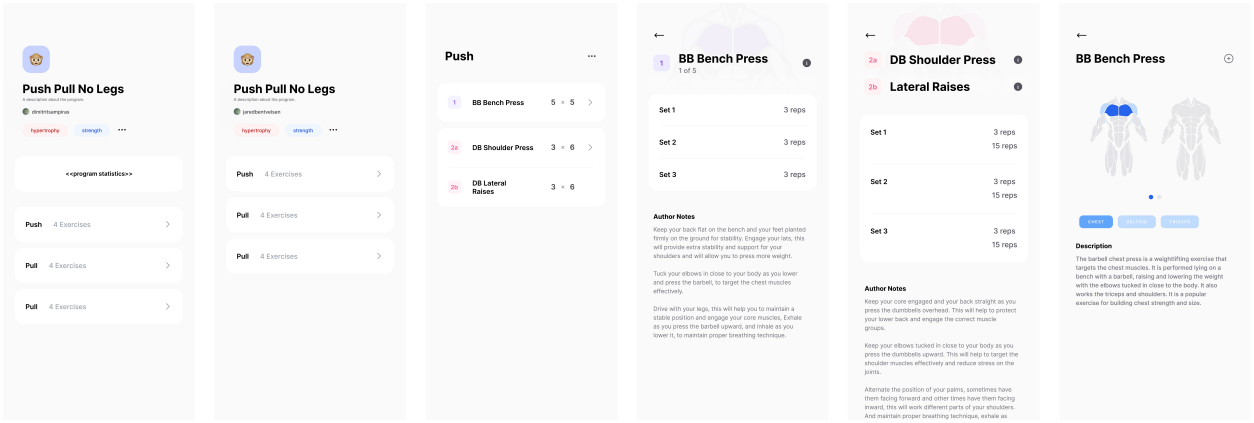

Figure 6: Static Program Stack: View when browsing programs from other authors or non active, personal programs.

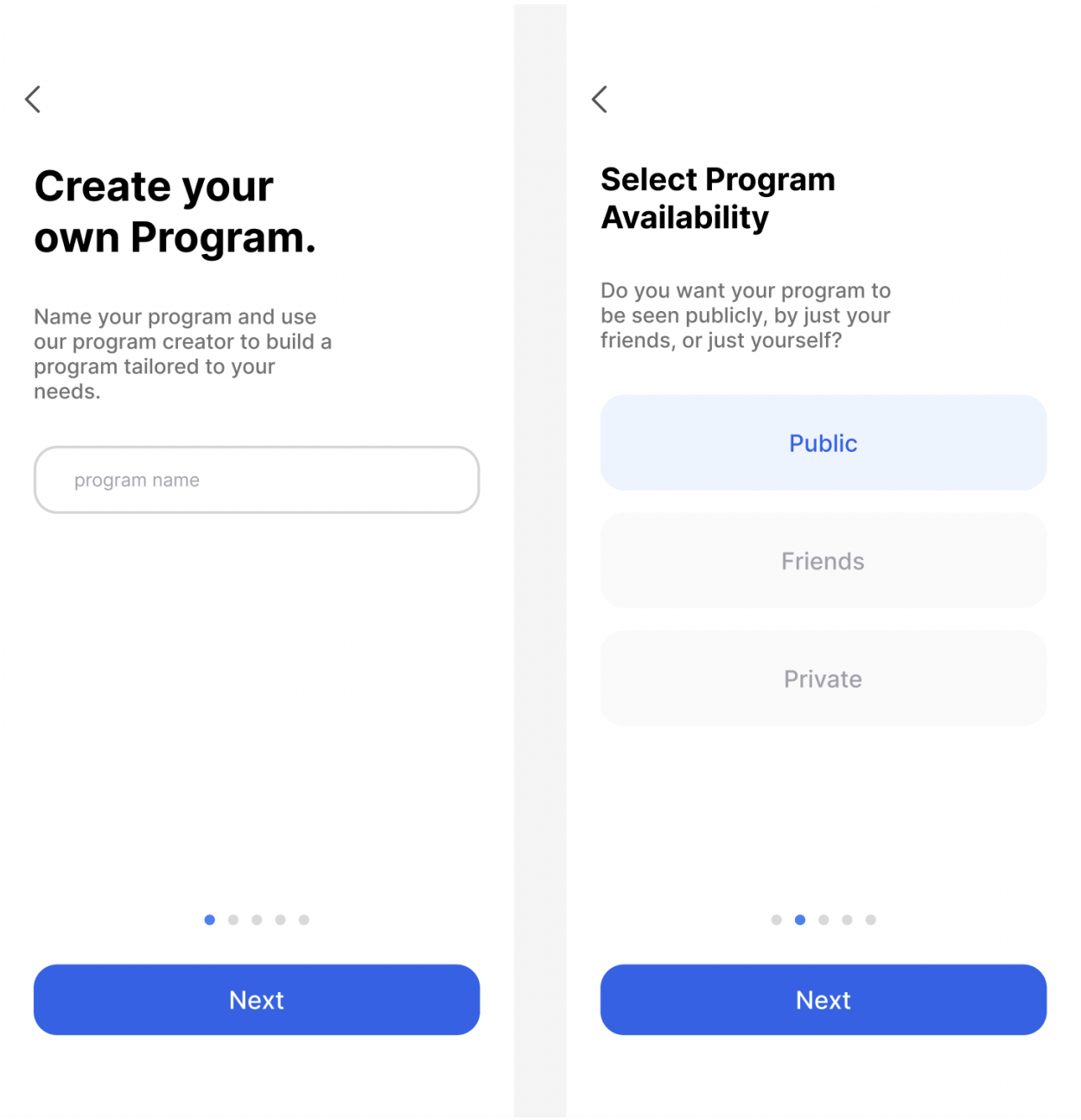

Figure 7: Multi page form for creating a program.

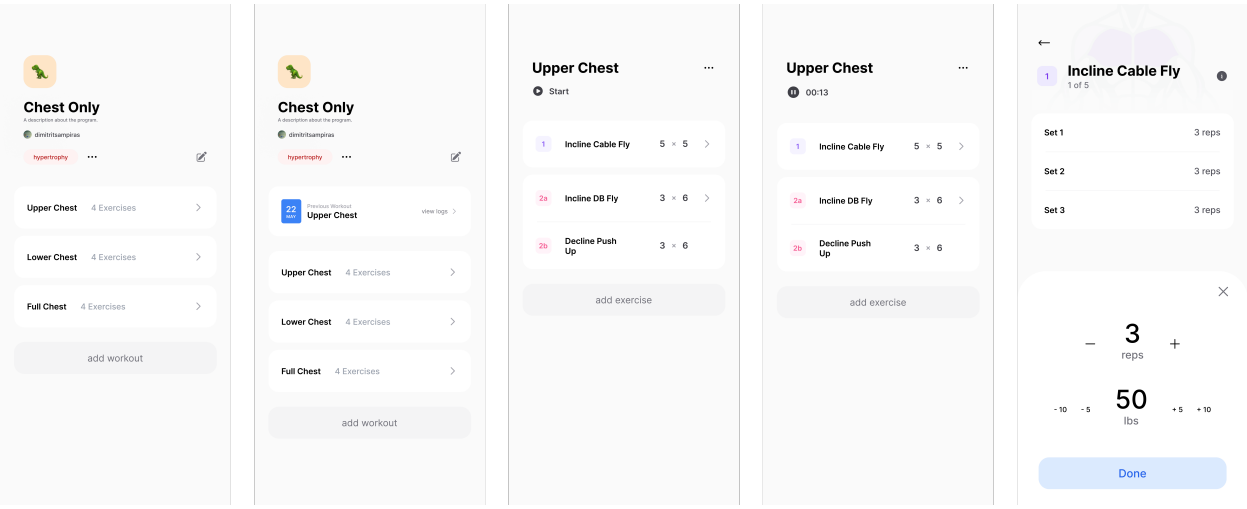

Figure 8: Multi page form for creating a program.

9 Design of Hardware

Olympian will run on mobile devices (e.g. iPhone, Samsung Galaxy) not designed or manufactured by the team.

10 Design of Electrical Components

N/A - See Section 9.

11 Design of Communication Protocols

Currently, all user information is stored in the remote database - no data is stored locally on the mobile device.

Communication between client and server will be done with HTTP requests. Specifically, the following technologies are used in client-server communication:

1. The query language GraphQL is used as a communication layer between the client and server. GraphQL aids in type checking and reduces the need for multiple endpoints (and thus requests made by the client) in the server. See https://graphql.org/faq/ for more information.

2. Prisma is used to abstract away SQL queries to the database, and define the database schema for all Olympian data. See https://www.prisma.io/docs/concepts/overview/what-is-prisma for more information.

12 Timeline

Below is a timeline of the major milestones and the developers responsible for completing them.

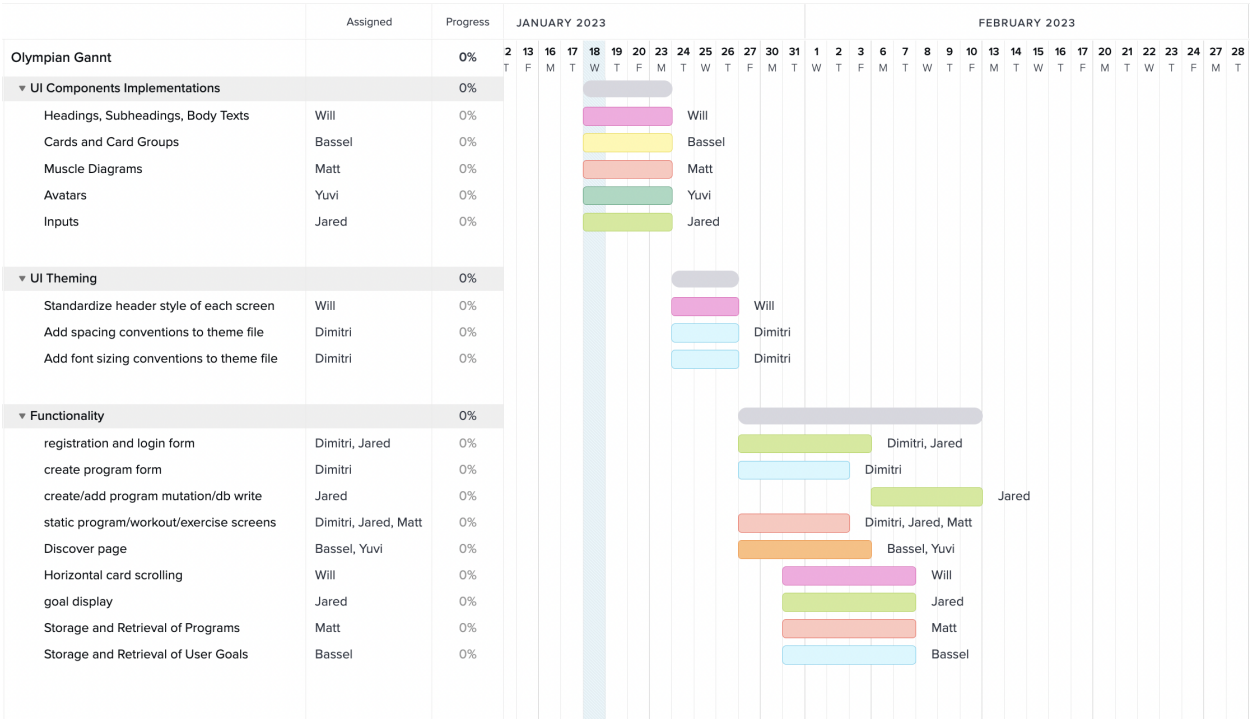

Figure 9: Multi page form for creating a program.

A Interface

See Figma.

B Mechanical Hardware

N/A

C Electrical Components

N/A

D Communication Protocols

The project employs Hyper Text Transfer Protocol (HTTP), enabling the client to communicate with the server and database through HTTP requests.

E Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   Limitations: require database to be populated (need users to create workouts and routines so other users can view and add) Given unlimited resources, having advertising or even starting users would help this problem of un-populated data.

   Another limitation would be that users cannot project their goals (extrapolation of their fitness goals). Given unlimited user data and statistics this could be predicted to help users estimate when they will reach their fitness goals

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

   Considering the software architecture, another alternative design for the flow of exercise, workout and routines that was considered was time-based. This means that each (workout, routine, exercise) would have a time and their sub-parts would schedule off a portion of the total duration.

For our actual implementation, we decided to keep these as separate modules because creation is its own process and does not directly interface with the user module. By keeping these as separate processes, it helps exercise the idea of 'separation of concerns'.

We decided not to use time-sectioning for the workouts / routines because it would cause limitations and be much more difficult to implement compared to having them as a sequence of events. If we wanted to collect the total duration of the workout / routine, a simple summation of parts would suffice.