

# NextGym Design

JDE Team 2315

Akshin Vemana, Andy Kao, Tyler Jeng, Leo Lee, Grihith Manchanda, Shashank Lal

Client: Jonathan Solomon

Team GitHub: <https://github.com/grihithmanchanda/JIC-2315>

# Table of Contents

Introduction .....	3
Terminology .....	4
System Architecture.....	5
Data Storage.....	7
Component Design.....	9
UI Design .....	12
Appendix A: Firebase Login.....	19

# **Introduction**

## **Background**

NextGym is a mobile application created for and centered around gym-goers and managers who desire the ability to manage all their gym-related activities with a few, simple taps. The application allows users to manage and/or view the exercises, equipment, and gyms that are available to them. Utilizing a hosting service, NextGym allows 24/7 online access and management services in and out of the gym.

## **Document Summary**

The System Architecture section introduces the language and database of our application that will be available to mobile devices on iOS and Android. It describes the static and dynamic elements of our system's architecture and presents diagrams that further delve into the specifics of both elements.

The Data Storage section goes into the specifics of our application's database and its structure. As we are utilizing the Firebase NoSQL database, we will be presenting a diagram illustrating the interaction between the user interface and the data layer. Furthermore, we will be providing information on file use, data exchange, and our security measures.

The Component Design section details the components of our application, and like the System Architecture section, will present static and dynamic diagrams for the relationships and interactions between components outside and during runtime.

The UI design section highlights the user interface of our application and includes screenshots detailing this user interface. Specifically, we will be including the major screens of our application and will detail our rationale behind their layouts.

## Terminology

**Backend:** Portion of the application that handles logic and data storage.

**Class Diagram:** A type of diagram that describes the system's classes, methods, and attributes.

**Firebase:** A NoSQL database.

**Frontend:** Portion of the application that involves user interaction and the user interface.

**JavaScript:** High-level programming language that is used for developing various applications.

**NoSQL:** Non-tabular database systems that store data differently from relational tables.

**React Native:** Software framework used to develop applications for Android and iOS.

**System Sequence Diagram:** A type of diagram that showcases the interactions between objects in a particular use case.

**UML notation:** Unified Modeling Language notation is a standard method of diagrams that visualize system designs.

## System Architecture

This section will cover our choice of system architecture and present diagrams that illustrate the runtime behaviors and the major components of our application.

Our application will be one built for both iOS and Android devices.

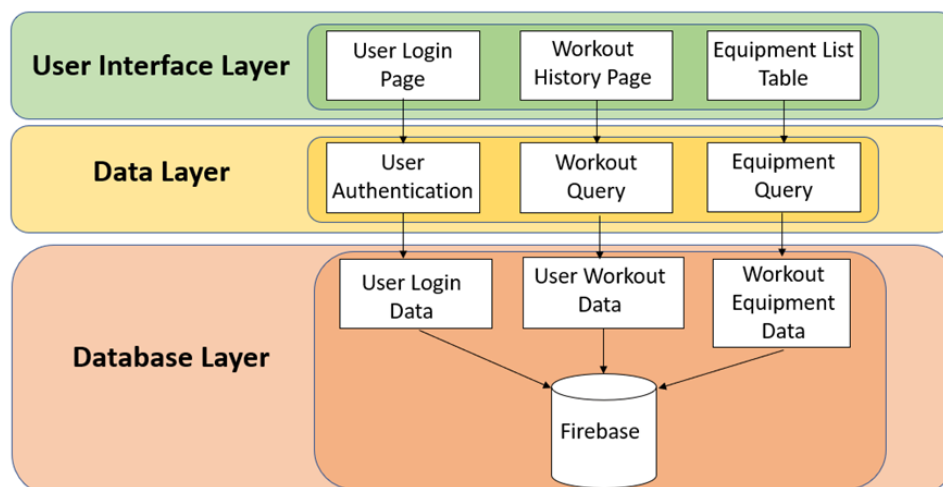
The front-end development and code will be done in React Native using JavaScript due to our group's familiarity with the language. This will allow us to focus more time on development instead of having to familiarize ourselves with a new language.

We will be utilizing Firebase as the database for our front-end and back-end. Several of our group members are familiar with NoSQL databases, including Firebase. With this database, we will have reduced development costs, easier scalability compared to other free databases, and inter-platform development opportunities.

### Static

We have chosen the layered architectural pattern as the basis for our application's system architecture. The application involves reading from a database and a subsequent processing of the data to display to our users. This means that we would have two layers for the system architecture. While the data layer interacts with the database and accesses information, the user interface layer sends requests for that information to the data layer and presents the data to the user.

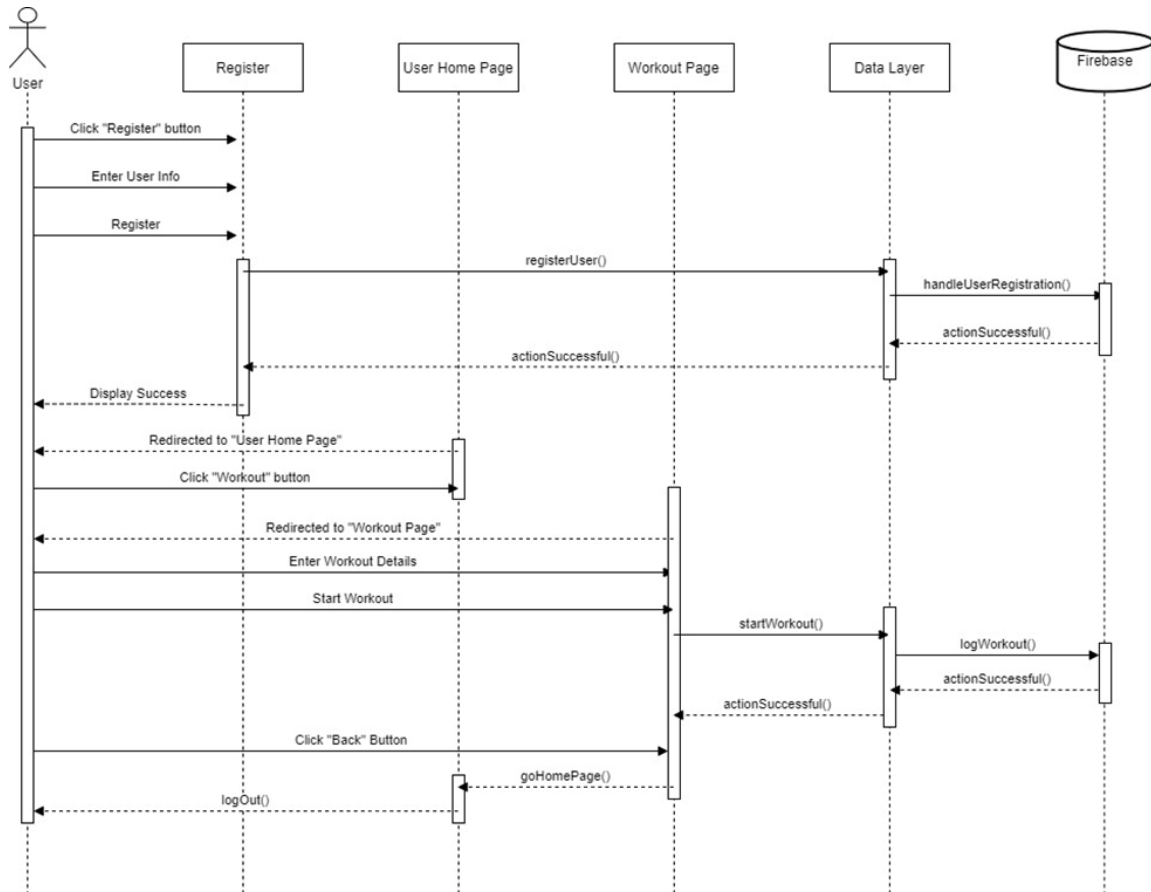
Figure 1 provides an overview of the various components within each layer. Using the user interface layer, users can log in to their accounts, view their previous workouts, and browse available workout equipment. This layer interacts with the data layer by allowing users to interface with the data and send any requests for additional data to the data layer. The data layer then interacts with the database layer by querying for these requests in our Firebase database, which houses the user login data, user workout data, and the workout equipment data.



**Figure 1.** NextGym's layered system architecture.

## Dynamic

An SSD, or System Sequence Diagram, further highlights our allayer model by providing insight into a particular task. Through UML notation and a high-level overview of the interaction between environments, the system sequence diagram (shown below) accurately showcases the registration process and workout sequence that users will follow.



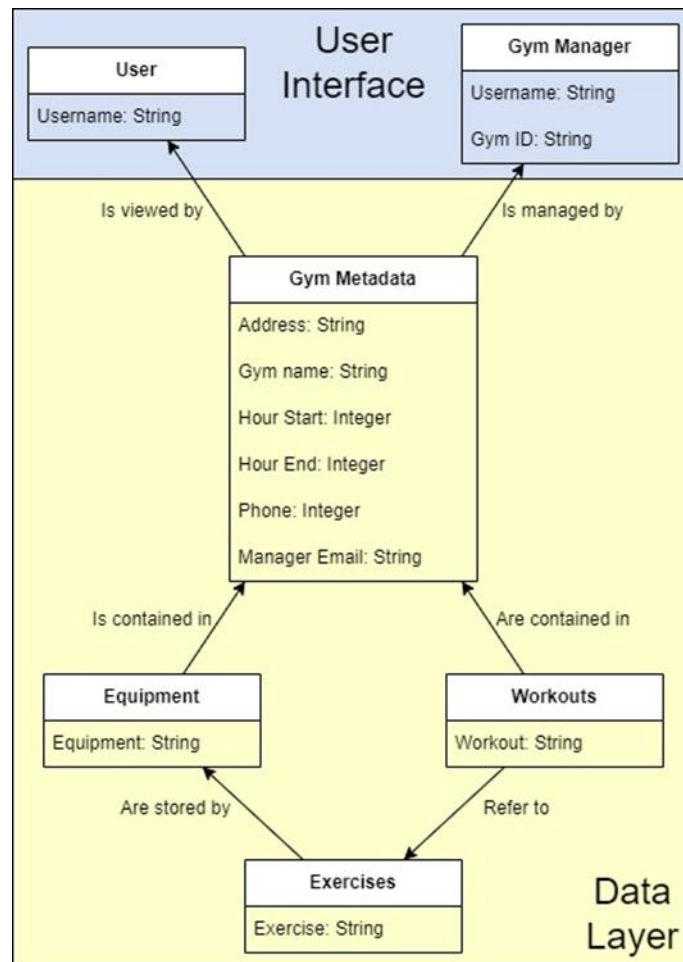
**Figure 2.** NextGym's system sequence diagram for the user registration and workout use case.

## Data Storage

In this section, we will cover our choice of database and data storage.

We will be utilizing Firebase as the database for our front-end and back-end, which is a NoSQL database. Several of our group members are familiar with NoSQL databases, including Firebase.

## Database Use



**Figure 1.** Database Diagram highlighting the hierarchy of and relationships between collections and data within our application.

Here is a list of all major entities:

1. **User:** Contains information relevant to basic gym-goers.
2. **Gym Manager:** Contains information relevant to administrative gym managers and gyms. Each gym account has a unique ID in addition to a simple username.
3. **Gym Metadata:** Contains various amounts of information pertinent to a gym.
4. **Equipment:** Contains a collection of equipment that can be viewed by users and edited by gym managers
5. **Workouts:** Contains a collection of workouts composed of exercises.
6. **Exercises:** Contains a collection of exercises that are linked to equipment.

## File Use

We do not create nor use any image files in our application.

## Data Exchange

Firebase seamlessly encrypts collections of data and allows users with proper authority (gym managers) to edit, add, or remove data pertaining to their gym. We use JavaScript files to interface between the user's input and Firebase. Firebase's JavaScript SDK allows us to directly make calls to the backend in parallel to the frontend code. These backend calls have been structured as asynchronous services, and these services work with Firestore Database and can perform CRUD operations on all data present within.

## Security

Firebase facilitates secure communication between front-end and back-end exchange of delicate data such as user information or credentials. The data is encrypted by Firebase, and we will not need to secure the data channel since we are deploying the app through mobile devices that run Android and iOS. We have two login channels: one for users and one for gym managers. Either of the types of users are unable to access the other type's home screen. It is important to note that even though we do not store passwords in our general database, we do store usernames as personal identifiers across the code. A future task would be to hash these usernames and store the encrypted hashes instead of their plaintext forms to further improve user security. We also realize that the application could collect a user's health data to improve their experience, and if/when the decision is made to do so, security and privacy of this data will be a paramount focus for the team.



## Component Design

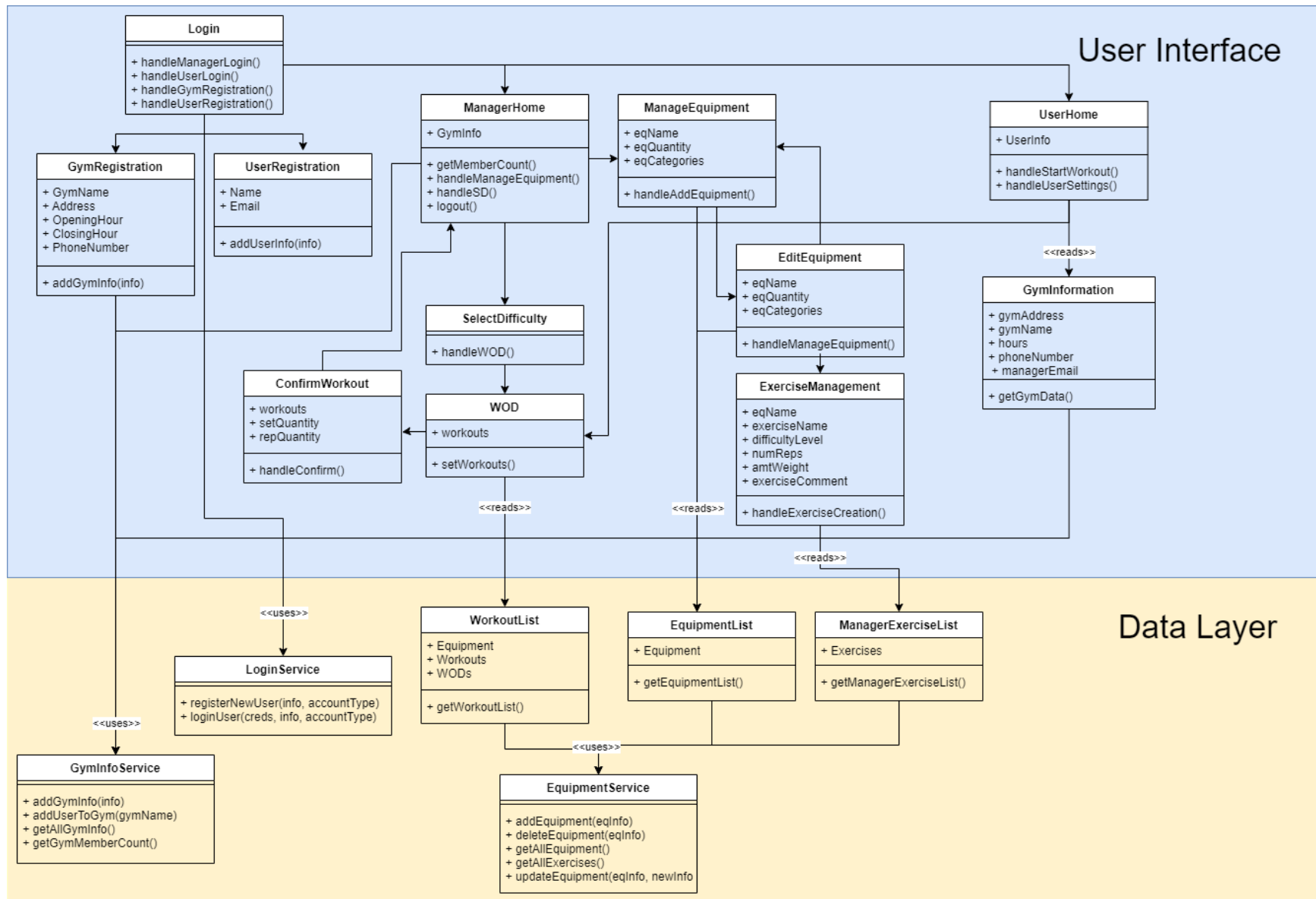
In this section, we will give an overview of the components we use in our application. We have both a static diagram, showing the relationships between different components, and a dynamic diagram, showing interactions that occur during runtime.

The pages present on the user interface sides of the static and dynamic diagrams match with the primary objectives of the application, and the services / components included in the data layer of the static diagram incorporate the components present in the data layer of the dynamic diagram.

### Static

The class diagram shown in Figure 1 on the next page shows our static relationships between objects. We have two layers shown here, with the third layer of the database not shown because it is not a formal class within our code. The blue represents all the user-facing content that makes up the user interface, helping both users and managers accomplish all the necessary tasks. The yellow represents the data layer, comprising of the components and services that load and interact with all the data stored in the database. We also show the type of relationship components have.

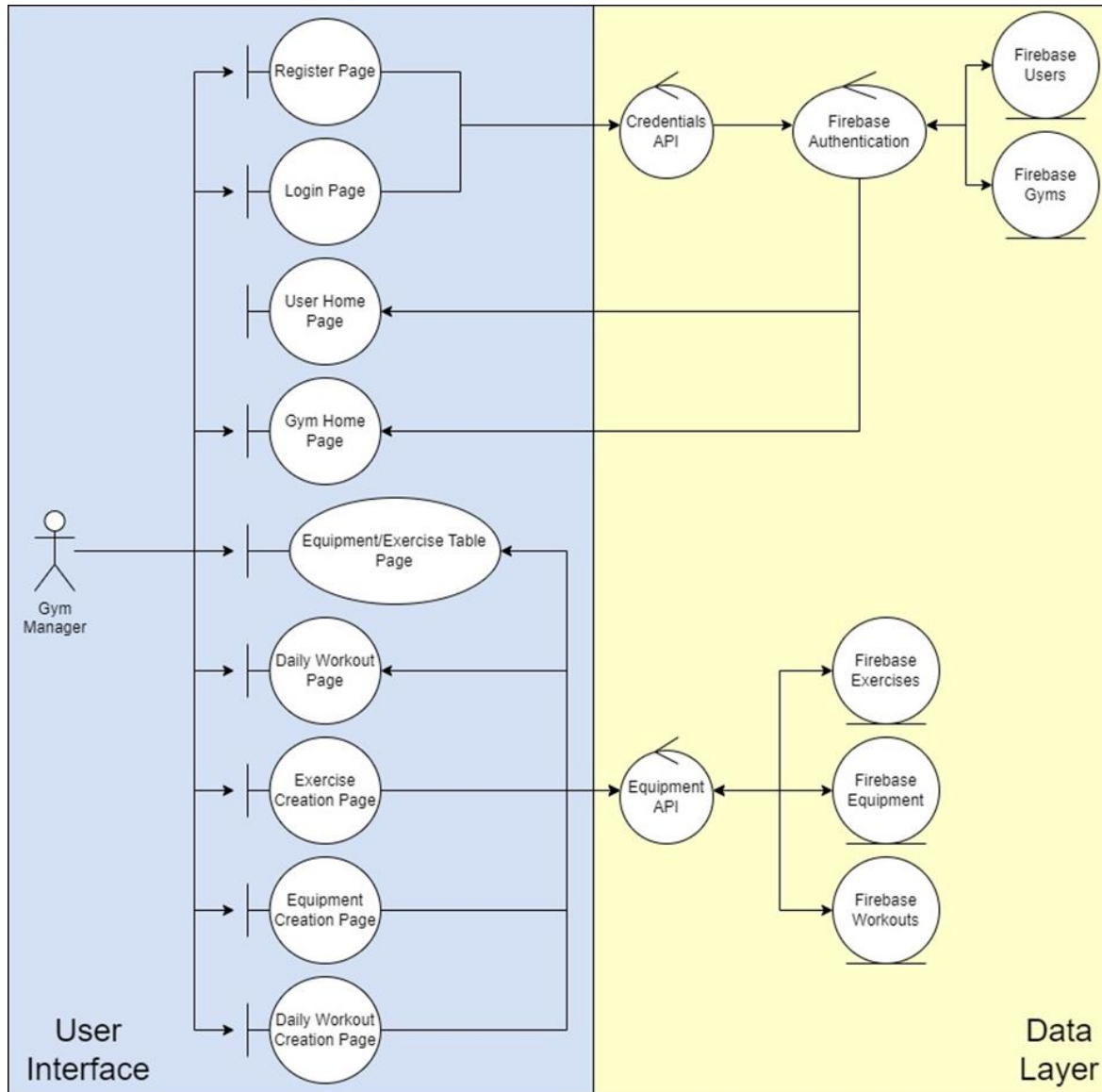
Unlabeled arrows represent screen transitions; <<reads>> labeled arrows represent a screen taking information from a component to display, and <<uses>> labeled arrows represent a service utilization necessary to complete a task.



**Figure 1.** A class diagram outlining the static relationships between all the different components of our app.

## Dynamic

We show a dynamic robustness diagram of our app in Figure 2. Again, we divide up into the user interface layer and the data layer. The user interface handles all the screens, which in turn send requests back to the necessary APIs to execute them.

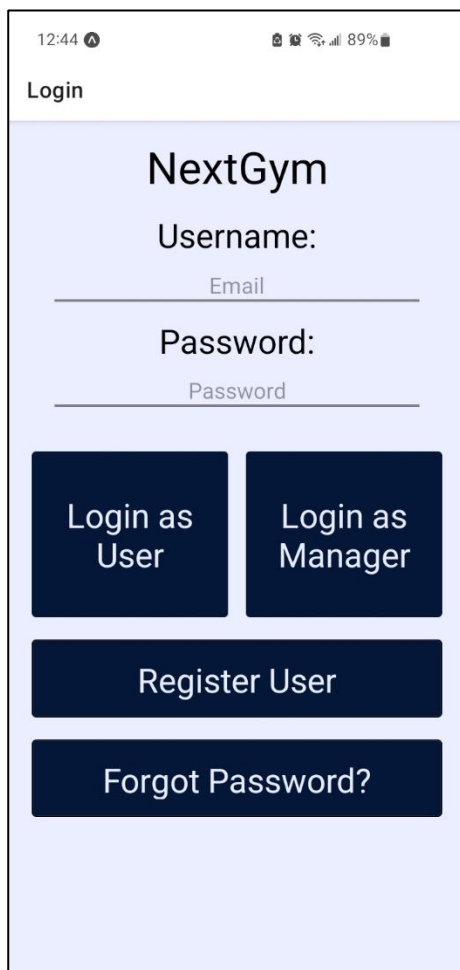


**Figure 2.** Robustness diagram outlining the interaction between gym managers and the various components of our application.

## UI Design

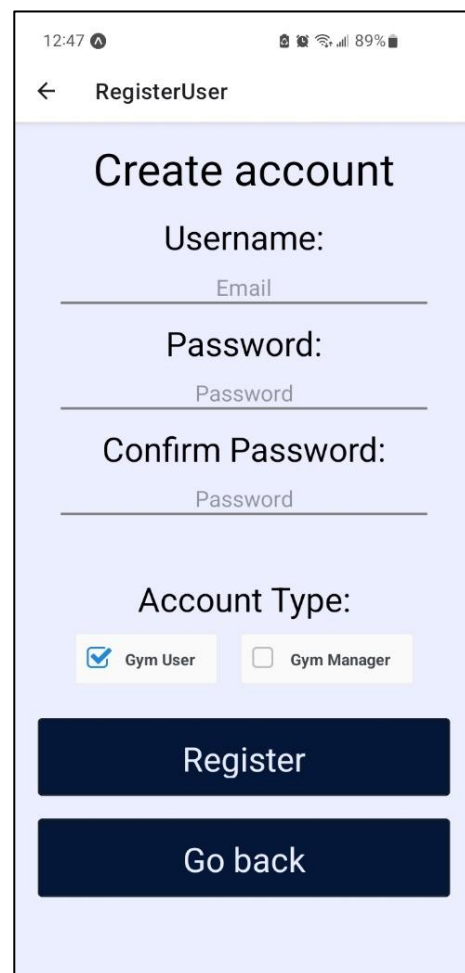
In this section, we will be presenting the User Interface of our application, which users and managers alike will utilize in order to manage all of their gym-related needs. We will also be showcasing the major screens of our application and our decisions behind their formats and layouts.

Upon opening the NextGym application, users will be brought to the screen shown in Figures 1.1 and 1.2, where they will login or register as a user or a manager of a gym. There is also an option to reset a password in case the user forgets it. This screen has a simple and efficient layout, with all buttons clearly marked. The checkboxes in the register screen easily allow the user to select the account type (user or manager) and the text boxes are all labeled for easy input.



The login page UI is displayed on a mobile device screen. At the top, the status bar shows the time 12:44, a battery icon, and 89% battery. The page title is "Login". Below the title, the app name "NextGym" is centered. The form includes a "Username:" label with an "Email" placeholder, a "Password:" label with a "Password" placeholder, and two buttons: "Login as User" and "Login as Manager". Below these are two more buttons: "Register User" and "Forgot Password?".

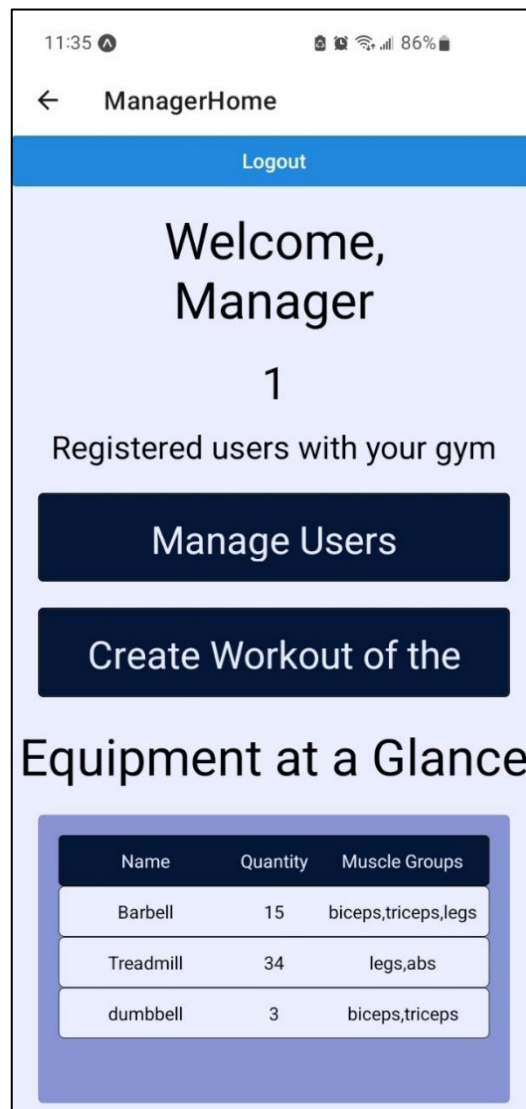
**Figure 1.1:** Login page UI



The register page UI is displayed on a mobile device screen. At the top, the status bar shows the time 12:47, a battery icon, and 89% battery. The page title is "RegisterUser" with a back arrow. Below the title, the form includes a "Create account" heading, a "Username:" label with an "Email" placeholder, a "Password:" label with a "Password" placeholder, and a "Confirm Password:" label with a "Password" placeholder. Below these are two checkboxes: "Gym User" (checked) and "Gym Manager" (unchecked). At the bottom are two buttons: "Register" and "Go back".

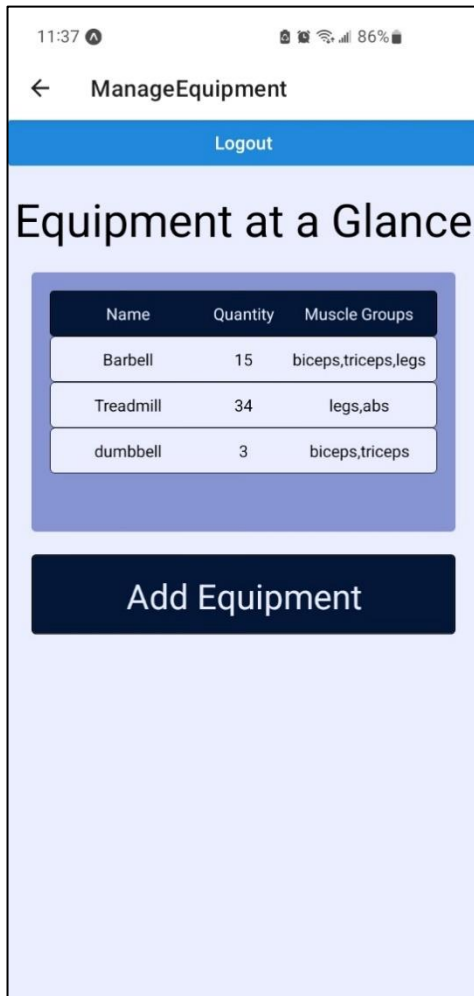
**Figure 1.2:** Register page UI

When logging in as a manager, the manager will be able to manage the users of their gym, create workouts of the day, manage equipment, and change gym settings. Each option is clearly laid out and labeled for the manager to select and leads to a separate page where the manager will conduct their business. Also, the logout button at the top is in the same location on every screen, both on the manager's side and the user's side. The consistency there allows for the app to be much easier to use by both users and managers. The manager home screen is shown below in Figure 2.1.

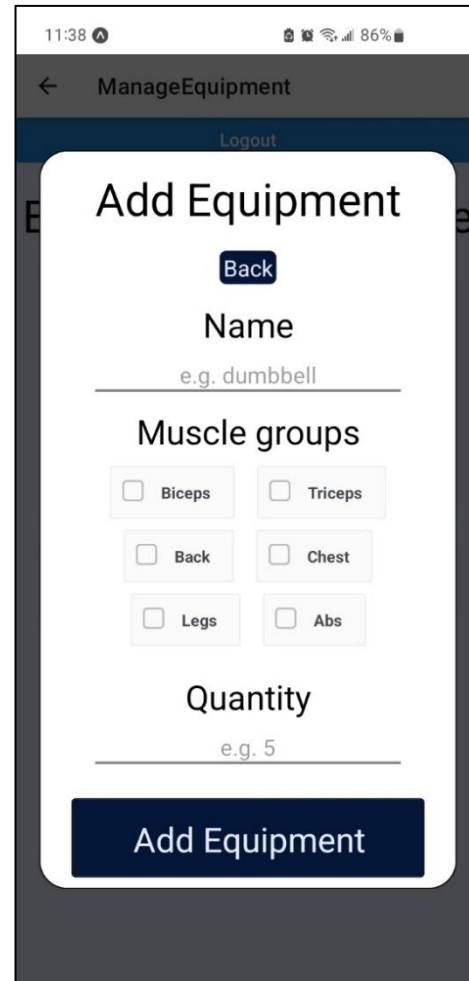


**Figure 2.1:** Manager home page UI

On the equipment management screen displayed in Figure 2.2, managers have access to a table of the existing equipment associated with their gym. The add equipment button produces a popup window that displays options and associated muscle groups for the new machine as shown in Figure 2.3.



**Figure 2.2:** Equipment management UI



**Figure 2.3:** Equipment management popup UI

Another screen that is accessible from the manager home page is the Workout of the Day (WoD) Creation screen, displayed below in Figures 2.4 and 2.5. Managers can select a difficulty level and a group of exercises that are associated with the gym’s equipment to create a WoD which is accessible to users.

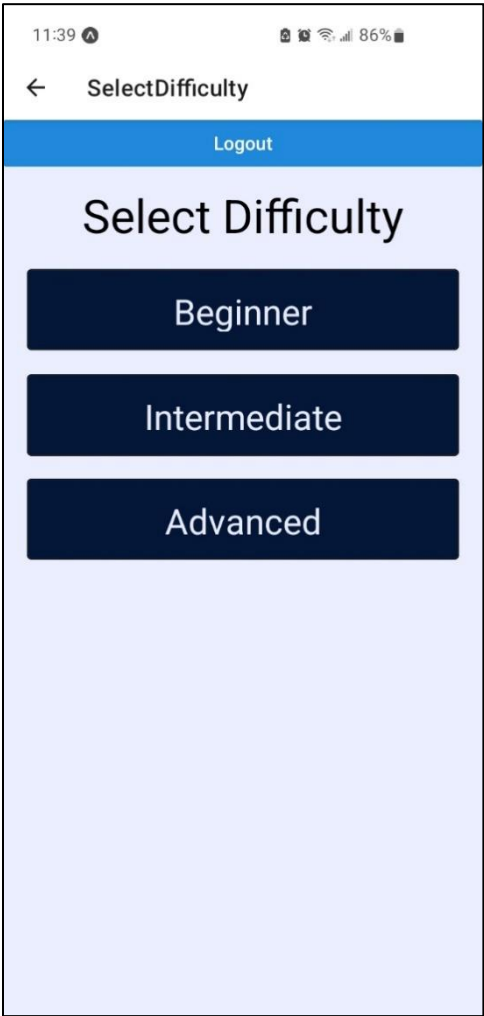


Figure 2.4: Workout of the Day Creation UI

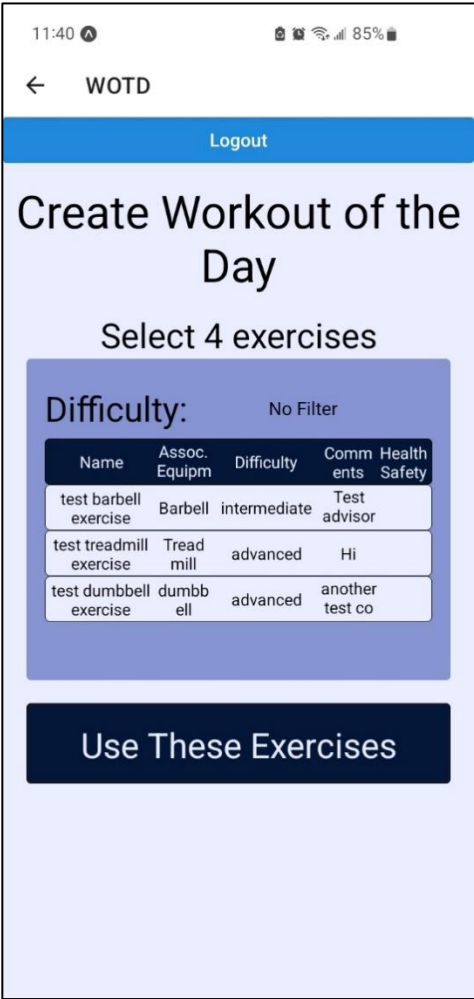
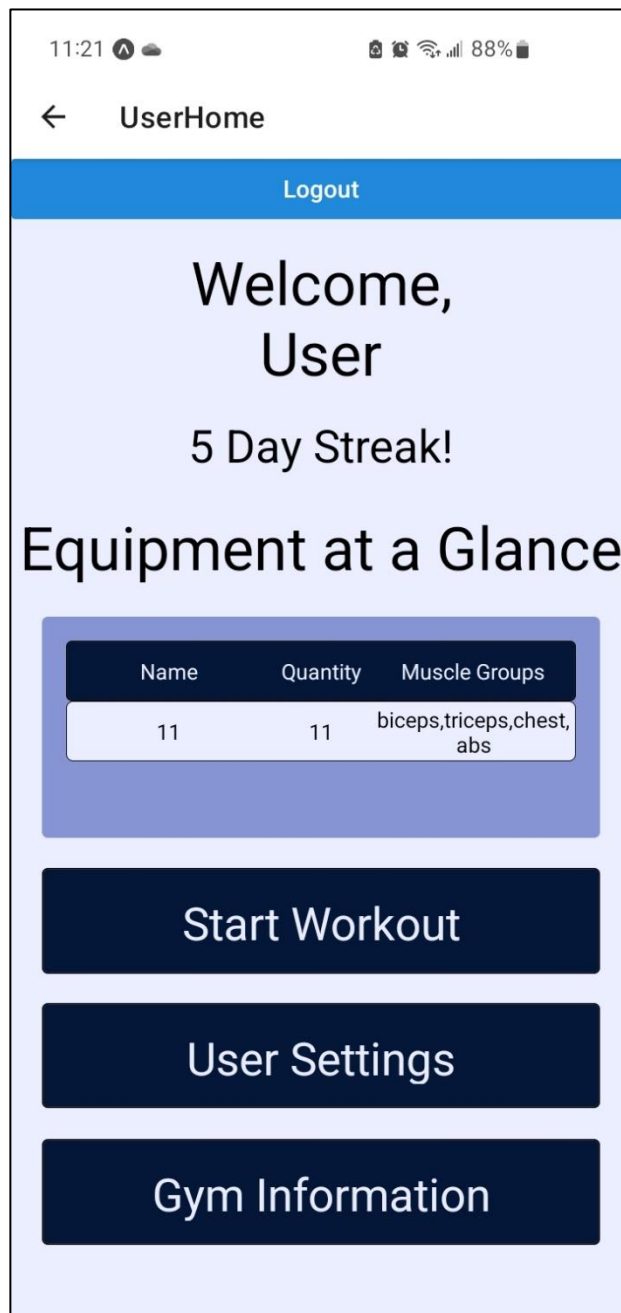


Figure 2.5: Workout of the day Selection UI

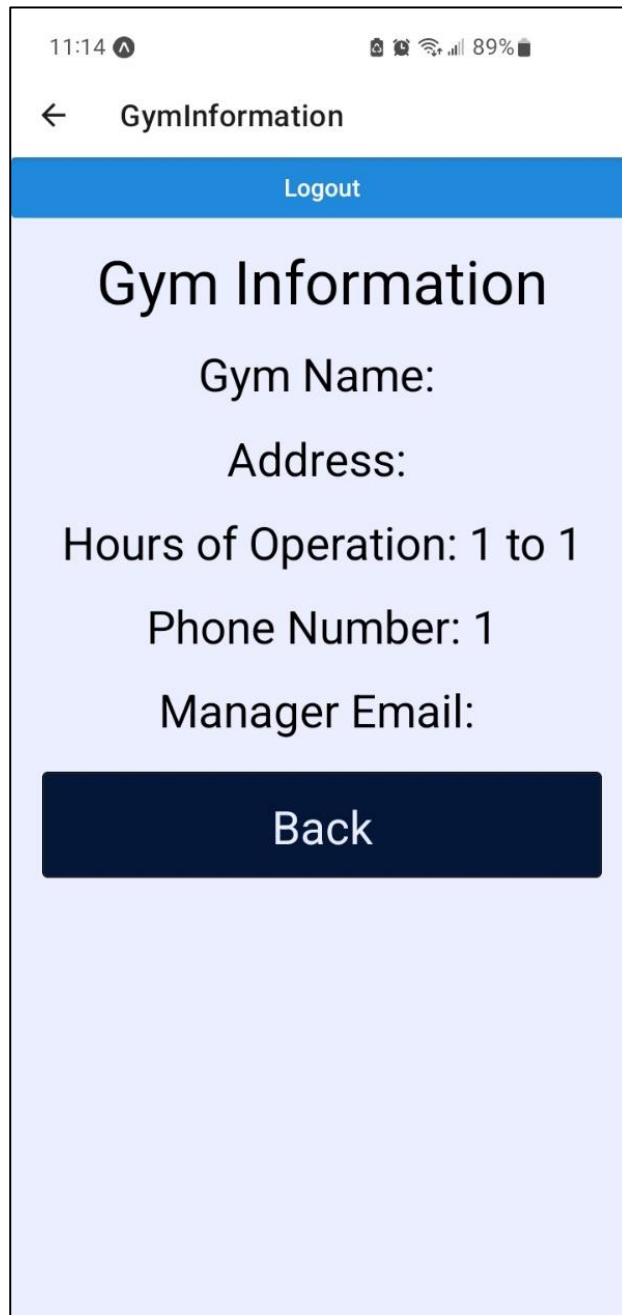
On the other hand, upon logging in as a user, individuals will be directed to the home screen, shown in Figure 3.1 below. They will be able to view the available equipment, select and start a desired workout, and change their settings on this page.



**Figure 3.1:** User home page UI

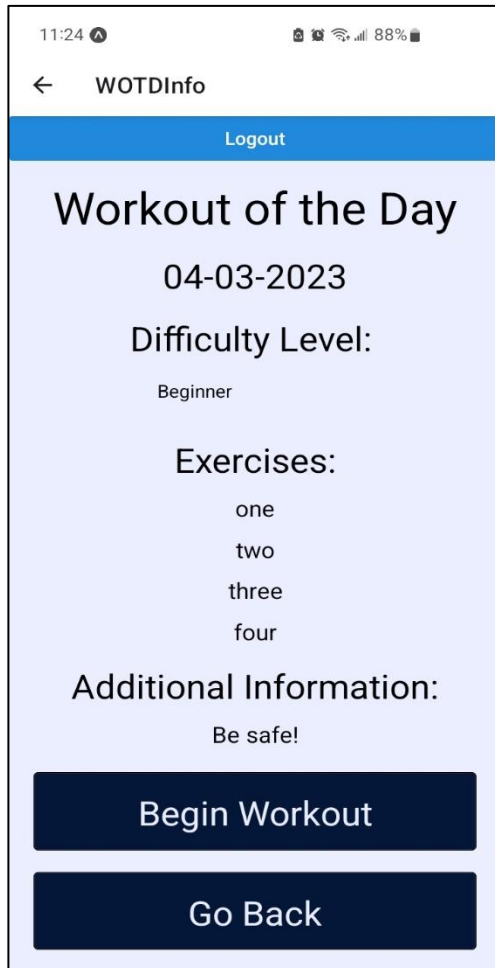


The “User Settings” button on the user home page navigates users to the gym information screen, which is displayed in Figure 3.2 below. Statistics such as the user’s associated gym’s name, address, hours, number, and email are shown.



**Figure 3.2:** Gym information UI

Another core feature accessible by the user is the WoD workout feature. By pressing the Select/Start Workout button in the center of the user home screen, users can view and select difficulty levels for the respective day's WoD. Upon starting the workout, a timer and counter will be activated for the user to track their progress. Both the start and progress screens are shown below in Figures 3.3 and 3.4, respectively.



**Figure 3.3:** Workout of the Day Start UI



**Figure 3.4:** Workout of the Day Progress UI

## Appendix A: Firebase Login

Google Firebase provides developers with its own login scheme. Firebase provides methods for signing in and creating a user given an email and a password. Data security is provided by firebase, helping us make sure that sensitive content, such as passwords, are protected. Rather than use these methods directly though, these are called within the login and logout functions we have for our app specifically. This is because we must also differentiate between user and manager accounts. The methods we use are described below.

### Methods

#### **getAuth()**

This method returns the authentication service reference to be used throughout the application.

#### **createUserWithEmailAndPassword(auth, email, password)**

Given the authentication service reference, an email string, and a password string, it creates a new user within firebase for this login. This method also returns a user credential object, which both contains information about the user and allows them to use the app.

#### **signInWithEmailAndPassword(auth, email, password)**

Given the authentication service reference and a valid email string and password string combination, it logs them in. It does so by returning a user credential object, which both contains information about the user and allows them to use the app. If the combination is invalid, it will not successfully log them in and will raise an alert.

#### **signOut(auth)**

Given the reference to the authentication service, it will log the current user out of the app and remove the credential. This does not return anything.