CSC 431

UFIT

System Architecture Specification (SAS)

Team 02

Maansi Patel	Software Developer
Natalia Jimenez	Prototyper
Diep Vu	Scrum Master
Isabel Ogilvie	Software Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	03/28/2022	Maansi Patel, Natalia Jimenez, Isabel Ogilvie, Diep Vu	First Draft
2.0	04/25/2022	Maansi Patel, Natalia Jimenez, Isabel Ogilvie, Diep Vu	Second Draft
3.0	05/02/2022	Maansi Patel, Natalia Jimenez, Isabel Ogilvie, Diep Vu	Final Submission

Table of Contents

Table of Figures

1. System Analysis

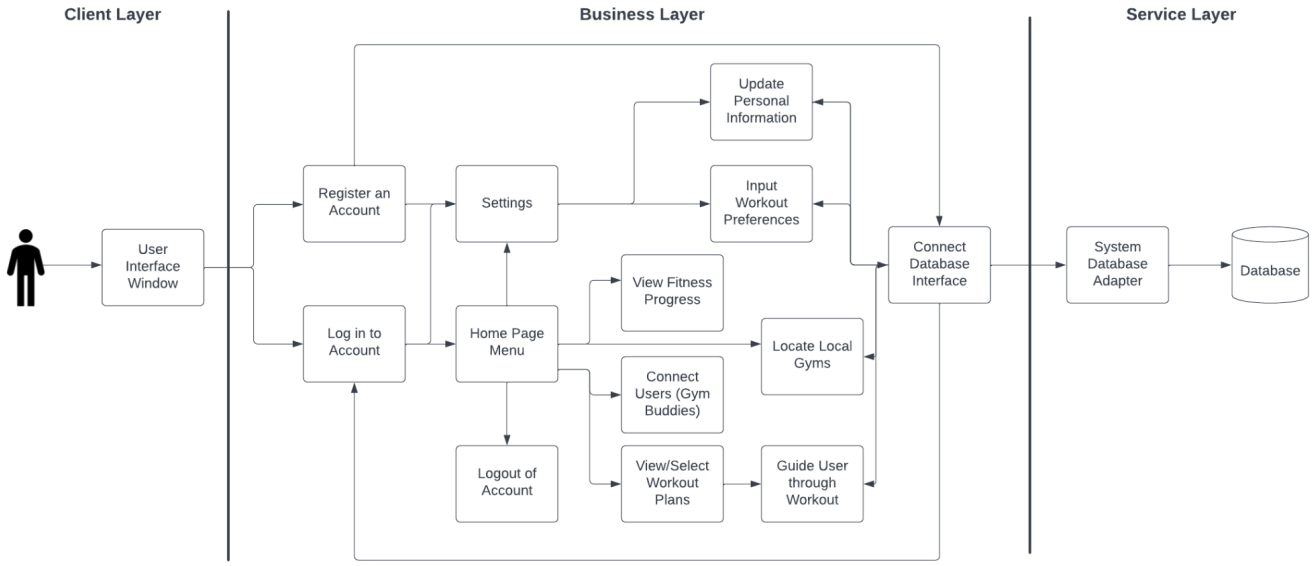
1.1 System Overview

Physical activity or exercising plays a crucial role in maintaining a healthy lifestyle and improving mental health. UFit is a fitness and social networking mobile application that will connect individuals who are looking for ways to stay active and meet their fitness goals. The composition of the system is focused around the following main functions:

- Creating customized workout plans (based on desired intensity, target muscles, existing injuries, etc.)
- Designing diet plans based on fitness goals (losing/gaining/maintaining weight/building muscles/increasing stamina, etc.)
- Connecting persons on the app to each other to help them find gym buddies to work out with based on fitness goals and location.
- Recommending suitable gym locations with the necessary equipment to achieve their goals nearby

The system will be composed of the user, the app, hosted through github, and the database which will store aloof the necessary system and user data. There will be interaction amongst all the components, and thus there will be a general flow from the main page through to the database component. The database will be hosted on the cloud, and within the database, each user's login, personal information, and workout profile/information will be stored and encrypted. The workout plans and instructions will be hosted offline, so our neural network will be stored on the device the app is hosted on for feed-forward purposes only. Lastly, because our app is being hosted on a remote server before delivering the app's capabilities to the users via the Internet, we will be using a SaaS (Software as a Service) Architecture.

1.2 System Diagram



- The above diagram represents a comprehensive overview of the UFit system architecture based on the functional requirements identified previously in the development process.
- Each node correlates to a functional requirement with the exception of *Main* and *Home Page Menu*. *Main* serves as the root of the application, and allows the user to access app functionality without a pre-existing account (i.e. registering an account). The *home page menu* serves as a landing page from which all other functionalities can be accessed.
- *View Fitness Progress, Connect Gym Buddies, View/Select Workout Plans* and so on are features only available to registered users, which is why they are only accessible through the *Log in to Account* node.

1.3 Actor Identification

- Gym-goer: It is a user (human) actor. The user will be able to add workouts to their list and modify their preferences based on their physical abilities. Users will be able to receive alerts on their planned workouts and connect with other users to do workouts together. The gym-goer is the primary actor because he is the one initiating the interaction with the system.

- Time: The system clock will help users keep track of their workout schedules by helping them stay organized through a shared calendar function with their Buddies.
- Database Administrator: It is an external system that serves as a secondary actor in a use case. It provides a workout database as well as tutorial videos. The content is provided by professional trainers.

1.4 Design Rationale

1.4.1 Architectural Style

The app will use 3-tier architecture in the form of:

1. **Presentation layer**: This is also called the UI layer. This is the top-most layer that the user is presented to interact with. The main function is to get input from users and send it to the Business Logic layer as well as to translate the tasks and results to something that the user can understand. The front-end of our application was developed with HTML, CSS, and JavaScript and is to be run on mobile applications.
2. **Business Logic Layer:** This layer processes information from the Presentation layer and can add, delete, or modify that data. The business logic layer can be deployed on the backend server and user remotely by the mobile application to reduce the load due to the limited resources available on mobile devices. The layer mainly focuses on the business front. The business logic layer includes workflows, business components, and entities beneath the hood.
3. **Data Access Layer:** User information received from the Business Logic Layer will be saved and encrypted in the database. This layer is created from the combination of data utilities, data access components, and service agents. This layer provides the business logic layer access to all data and uses Amazon Lightsail. The other factor for designing this layer is to select the correct data format and put in place a strong validation technique to protect from invalid data input.

1.4.2 Design Patterns

1. Factory (Creational): Factory Method Pattern allows the sub-classes to choose the type of objects to create without having to specify the exact class of the object. It promotes the loose-coupling by eliminating the need to bind application-specific classes into the code, which means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class.

2. Observer (Behavioral): Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically.

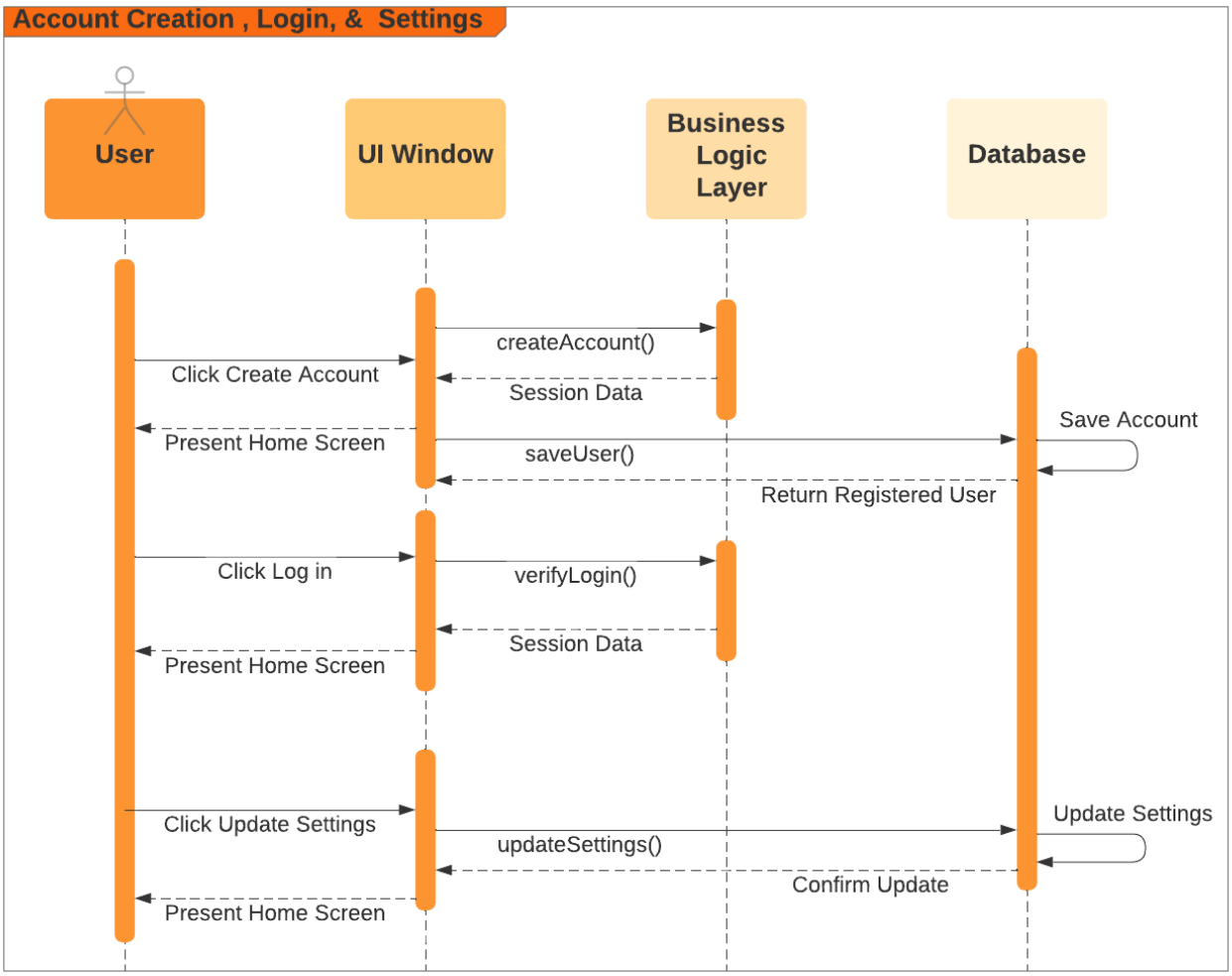
1.4.3 Framework

We will be using HTML, CSS, and Javascript for the frameworks. There is a lot of information on them and our team members are familiar with them. Amazon lightsail will be used to deploy the application. This will store user information and will be good enough for the early stages of the project. Our application will use Node.js as the framework, which will allow us to use HTML, CSS, and Javascript. Javascript will be used to write the backend and React Native Js will be used as the frontend framework.

2. Functional Design

2.1 Sequence Diagram

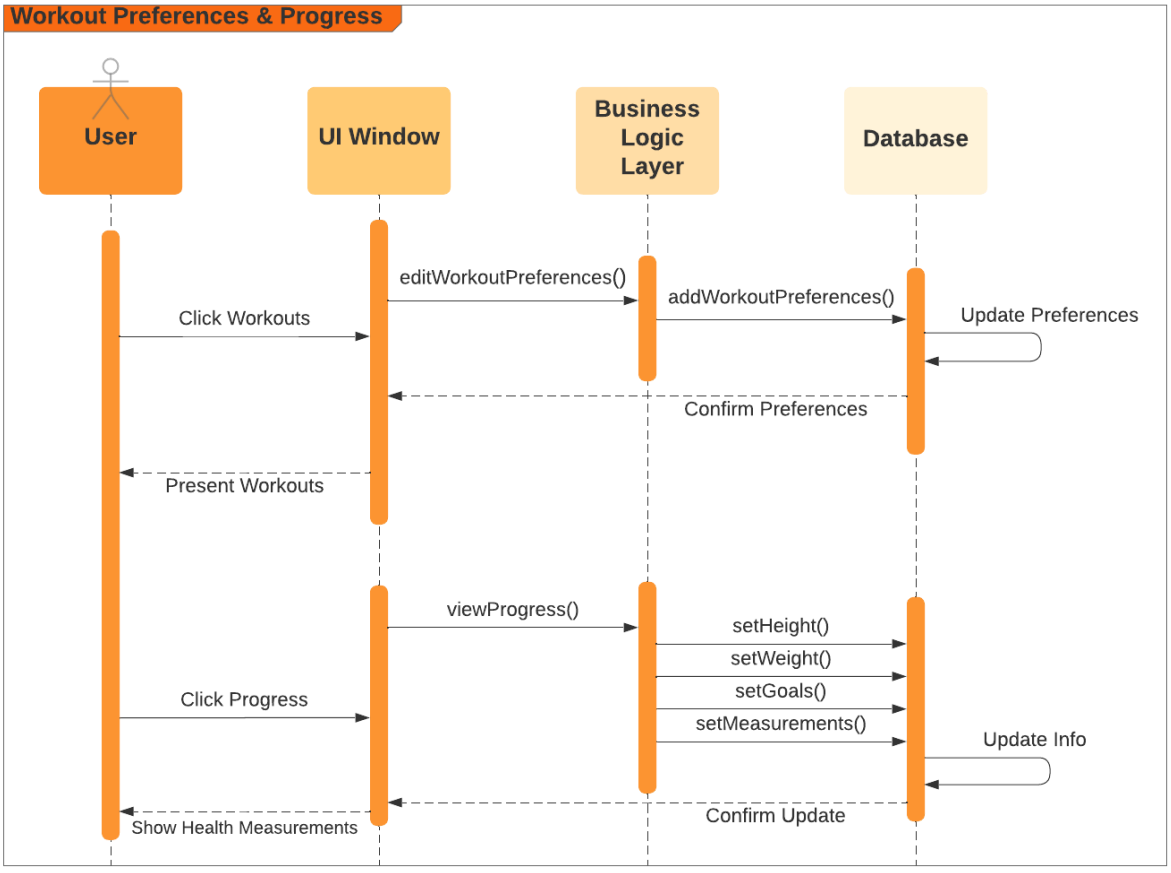
2.1.1 Account Creation, Login, and Settings



- When the user opens the application, they will be presented with the options to either create an account or login to an already registered account on the UI Window.
- If the user selects to create an account, the user will enter a desired username and password which will be registered and saved into the database. The database will then return the registered user and take the user to the home screen on the UI Window.

- If the user selects log in, the UI Window will verify the entered username and password with what the user registered with in the Business Layer and return session data. The user will then be presented with the home screen on the UI Window.
- The user can also update and change their account settings and information. Once they select update settings, the new information that the user enters will be sent to the database where it will be saved, and will signal an update confirmation and take the user back to the home screen.

2.1.2 Workout Preferences & Progress



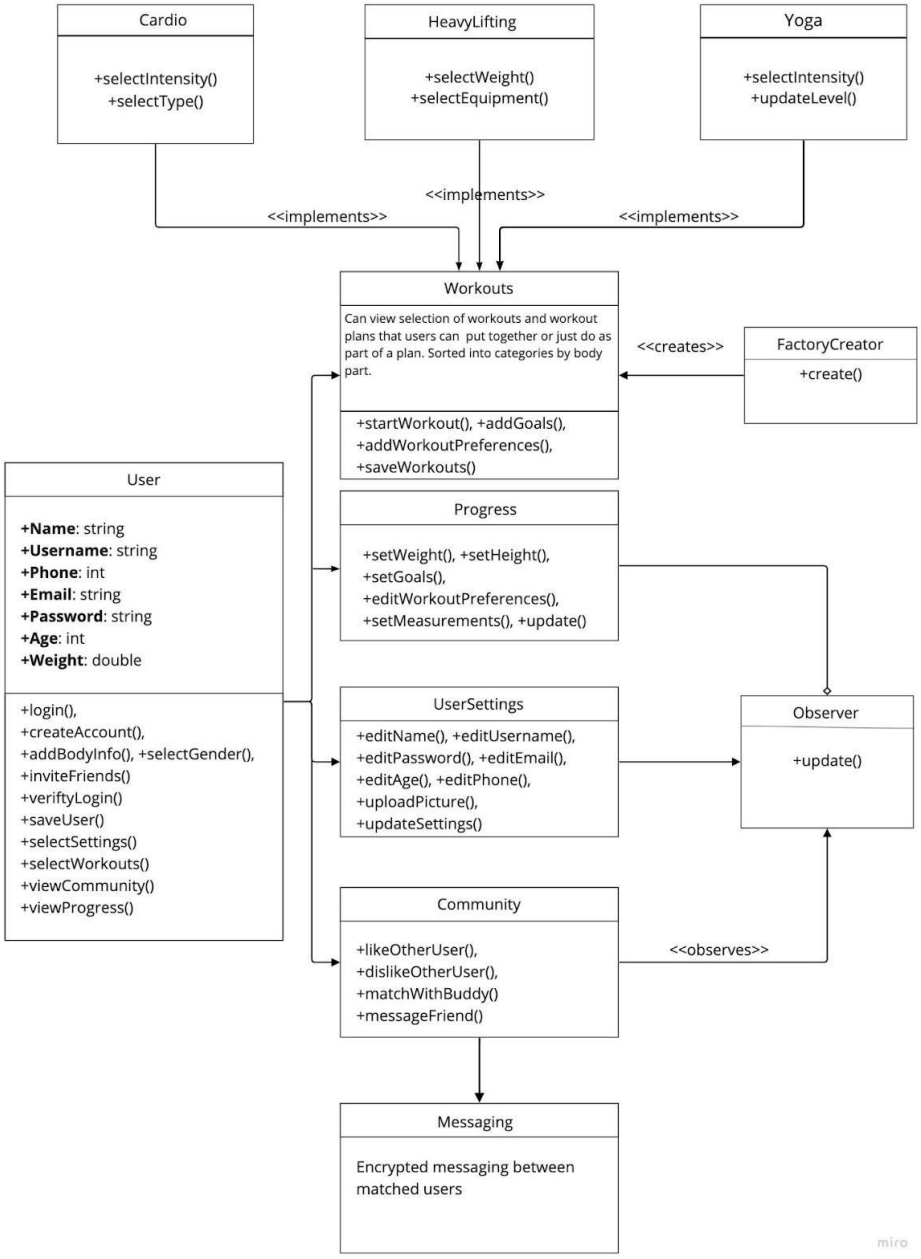
- Users can click the Workouts tab from the home screen where they can edit their workout preferences like intensity and type of workout. Once they add their new workout preferences, the Business Layer will send that information to the database where it will

be saved. Then a confirmation signal of the preferences will be sent back and the user will be presented with a list of customized workouts based on the input of preferences.

- The user can also view their workout progress from the home screen. Once they select view progress, they will be able to enter and update their current body measurements like height and weight, which the business layer will then send to the database where it will be updated. Then the database will send an update confirmation to the UI Window and the user will be presented with updated health measurements like BMI and calories burned.

3. Structural Design

3.1 UML Class Diagram


Cardio
+selectIntensity()
+selectType()
HeavyLifting
+selectWeight()
+selectEquipment()
Yoga
+selectIntensity()
+updateLevel()
<<implements>>
<<implements>>
<<implements>>
Workouts
Can view selection of workouts and workout plans that users can put together or just do as part of a plan. Sorted into categories by body part.
+startWorkout(), +addGoals(),
+addWorkoutPreferences(),
+saveWorkouts()
<<creates>>
FactoryCreator
+create()
User
+Name: string
+Username: string
+Phone: int
+Email: string
+Password: string
+Age: int
+Weight: double
+login(),
+createAccount(),
+addBodyInfo(), +selectGender(),
+inviteFriends()
+veriftyLogin()
+saveUser()
+selectSettings()
+selectWorkouts()
+viewCommunity()
+viewProgress()
Progress
+setWeight(), +setHeight(),
+setGoals(),
+editWorkoutPreferences(),
+setMeasurements(), +update()
UserSettings
+editName(), +editUsername(),
+editPassword(), +editEmail(),
+editAge(), +editPhone(),
+uploadPicture(),
+updateSettings()
Observer
+update()
Community
+likeOtherUser(),
+dislikeOtherUser(),
+matchWithBuddy()
+messageFriend()
<<observes>>
Messaging
Encrypted messaging between matched users
miro