CSC 431

COR

System Architecture Specification (SAS)

Team 13

Julia Eisner	Scrum Master
Jeffrey Hudak	Developer
TC McCaffrey	Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	3/24/2022	TC McCaffrey, Julia Eisner, Jeffrey Hudak	First Draft
2.0	4/26/2022	TC McCaffrey, Julia Eisner, Jeffrey Hudak	Second Draft

Table of Contents

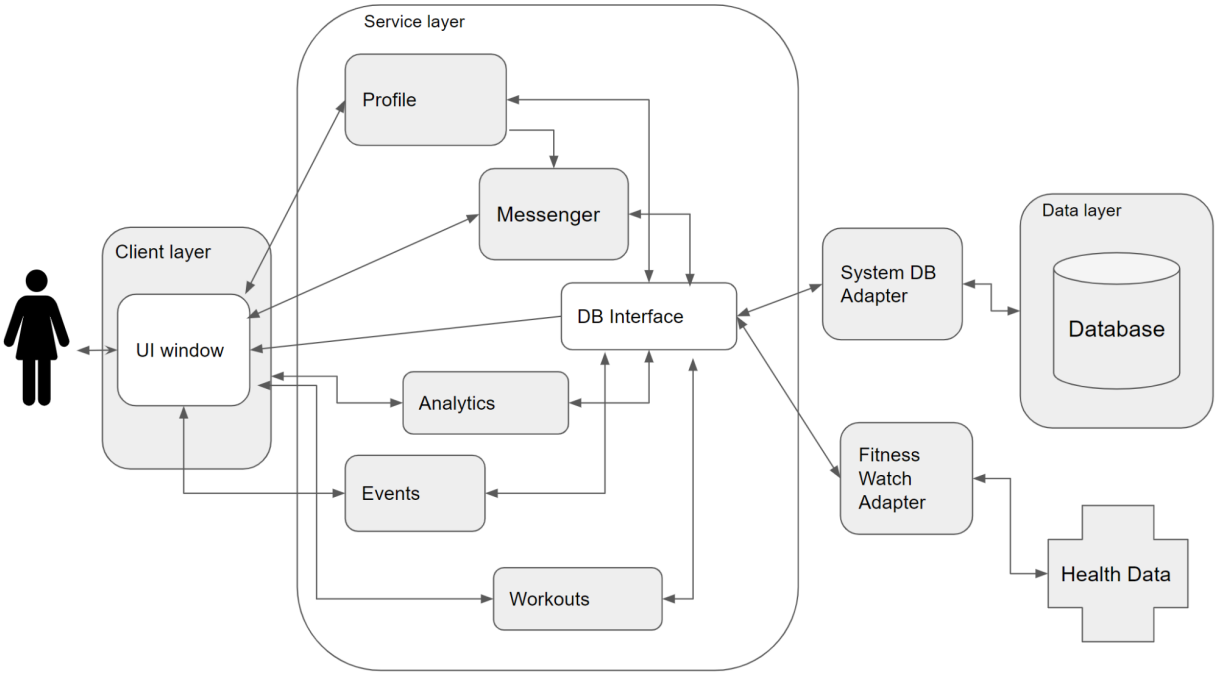
System Analysis

1.1 System Overview

This system architecture for COR will be following a 3 tiered architecture system including a client layer, service layer, and a database layer. Using this architecture will allow for a linear path of actions between the layers. The client layer will be the IOS user interface allowing the user to interact with the app on their mobile device. The service layer has the functionality and classes of the COR code. Included in the service layer programming will be the messaging feature, the events feature, the workouts finer, and the data analysis and visualization tools. The database layer will be the Firebase Database used to house the program data and provide authentication and security on the backend. Along with the Firebase Database, the user's fitness watch will be in the database layer, as it is providing user data to the service layer.

The client layer's user interface connects the user to the service layer. It displays the desired information to the user based upon the user's actions via graphics, widgets, and designed layouts. The user's interactions with COR will be almost entirely through the user interface, pressing buttons on the interface to initiate actions in the service layer. The only instance of an action the user will perform not in the client layer is connecting their fitness watch. The service layer consists of classes that carry out the functions of the app. When the user initiates an action via the user interface, the service layer will enter its functions, and if needed will pull or send data to the DB interface, which interacts with adapters that connect to the database. The adapters allow for the data to be in one uniform data type so that the service layer can function seamlessly with the different data files. The DB interface ensures that user interface requests can be interpreted by the database, and translate executed data to be compatible with the service and client layers. The database layer has the Firebase Database that allows for storage of the data that is sent by the service layer through the adapter. The Firebase Database and the Fitness Watch can both send data back to the service layer through the adapters. Firebase is also used in this layer to verify user login credentials and authenticate users.

1.2 System Diagram



1.3 Actor Identification

Actors interacting with the system include:

- New Users: Users are individuals who have not yet created an account with COR.
- Registered Users: Users are individuals who have registered their accounts with COR.
- Platform Server: Firebase will be used as a database on the backend that also provides authentication services.
- Fitness Watch: The data from the user's watch will be incorporated into the system.

1.4 Design Rationale

1.4.1 Architectural Style

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

- UI Layer: This will consist of the User Interface where the user interacts with the app.
- Service Layer: This layer houses the classes of our program that provide the functionality for the workouts, events, messaging, and other parts of COR.
- Database Layer: Firebase will be used to store the backend data and will also provide authentication.

1.4.2 Design Pattern(s)

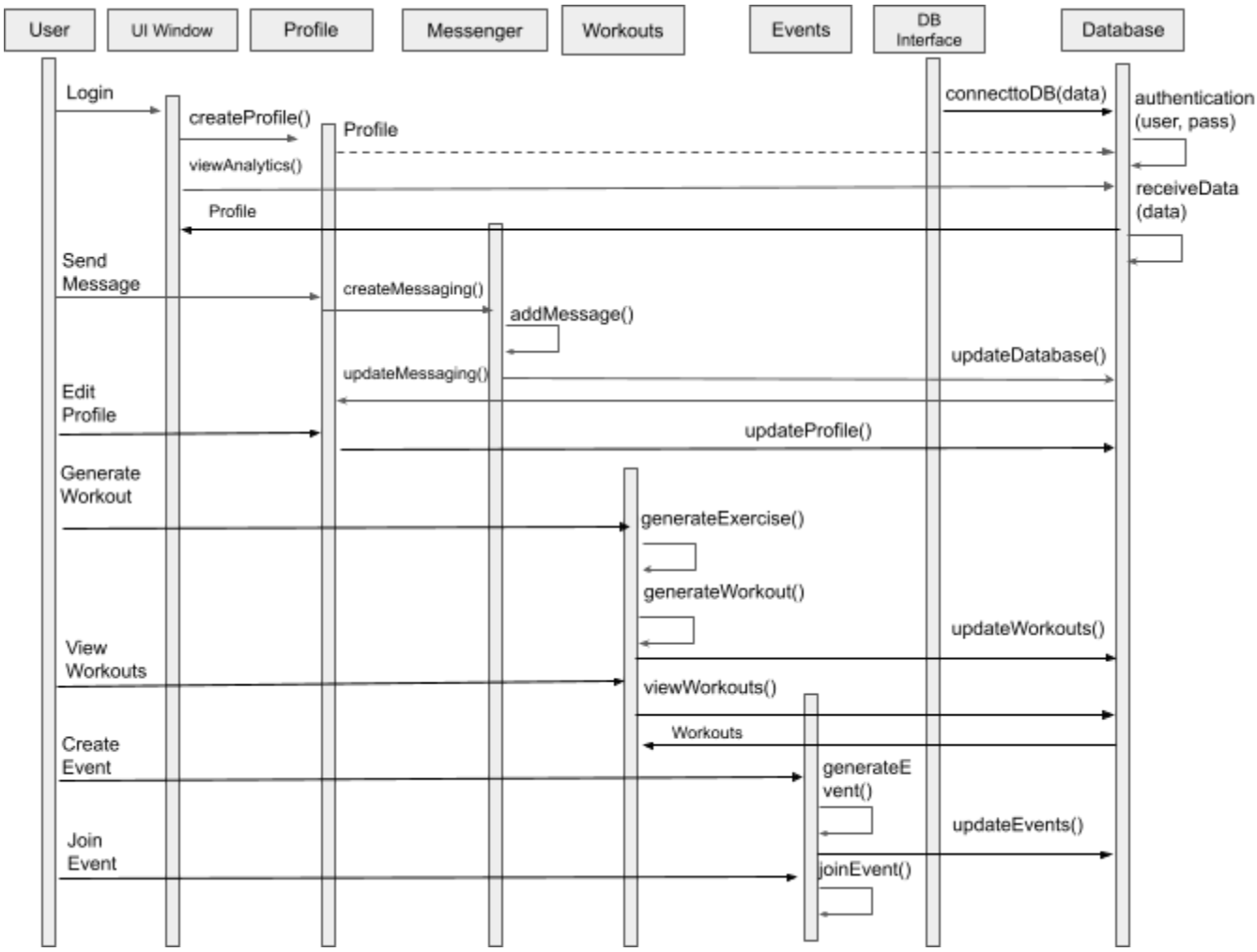
Our design requires the use of adapters between the service and database layers. The Firebase Database system stores their files as JSON files, so the adapter will convert the data into a format that will be compatible with the frontend Node.js framework. The fitness watch will also have an adapter to ensure that different brands of watches are providing their data in the same formats for handling by the service layer. These adapters will allow for smooth transitions of data to be handled by the classes in the service layer.

1.4.3 Framework

For the frontend of our program, we will be using Node.js. This is a common app and web development framework running on Javascript. Node.js does not have any buffering in its usage, so we can ensure that the users will have a seamless experience while using the app. It also has functionality for memory for users which gives the ability for profiles to be built and save data for easy access. On the backend, we will be using the Firebase Database for cloud storage of data. This was chosen so we do not have to have physical server space for the user data. Firebase also provides authentication for data safety. Since it is a cloud database, we will have the ability to sync across all the users of COR on their devices, syncing their watches and phones to one another. Firebase also allows for a basic cache of data to be available when offline, so users can still view workouts, events, analytics, and old messages.

2. Functional Design

2.1 User to Data



Description:

Login - When the user logs in it will either prompt them to make a profile, which will be authenticated by checking if the profile already exists in the database. If not, creating a profile will send a new profile to the database for future use. Otherwise, the system will allow the user to view their analytic data.

Send Message - The user can add a new message to either an existing or new conversation with another profile in the system, which either way will add a new message between the two users to the database.

Create Profile - The user can either create a new profile if they choose with different authentication or they can update their profile such as updating an email address or password.

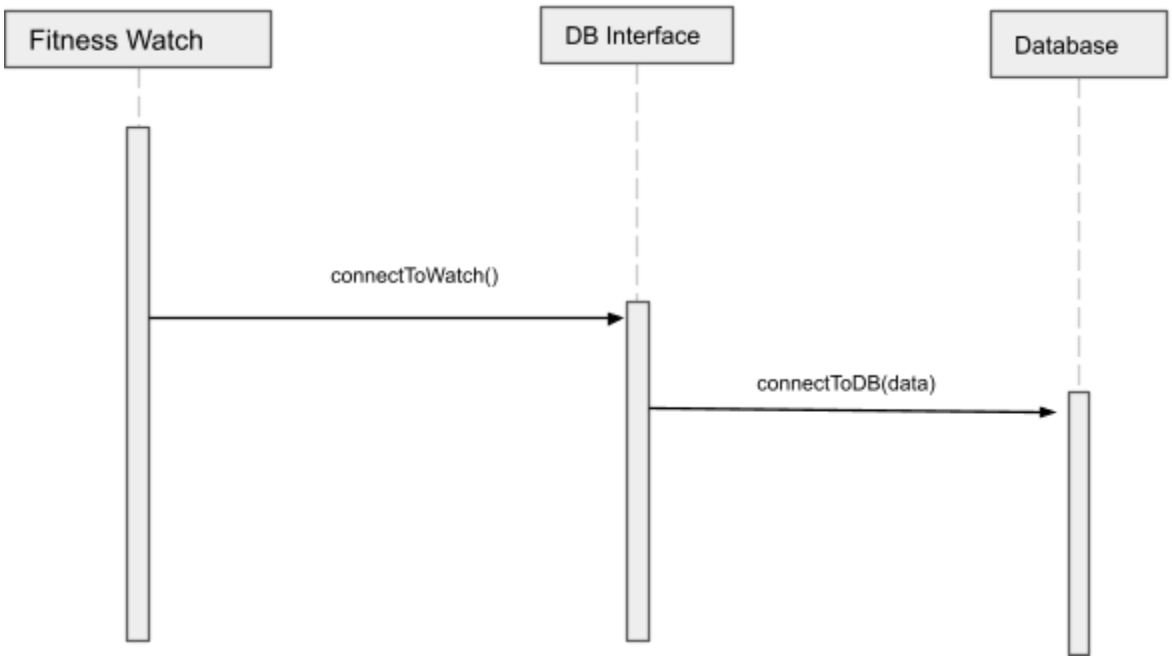
Generate Workout - The user can choose to either generate exercises or generate entire workouts. Exercises are individual pieces to workouts, such as “10 pushups”. Workouts are a curated list of exercises for the user to complete, which will then be sent to the database so the user can reference their old workouts in the future.

View Workouts - This pulls old workouts saved within the database for the user to see.

Create Event - After putting in event details, the user can officially generate an event which will save the event to the database for other users to see and join when they go to join events.

Join Event - A user can tell the system they plan on joining an event which will save the location, name, and time of the event to the user’s profile.

2.2 Watch to Database



Description:

Fitness Watch - After completing a workout, whether through workouts offered through COR or another fitness application a user chooses to use, the database will connect to the watch and pull new workout information to save into the database as workouts to be viewed in the app.

3. Structural Design