Lift X

System Architecture Specification

(SAS)

Team 3

Kyle Biggs	Front End Developer
Moose Griffth	Software Engineer
Christopher Rodriguez	Lead Architect
Jerome Fleischman	Team Leader & Scrum Master

Version History

Version	Date	Author(s)	Change Comments
1.0.0		Kyle Biggs, Moose Griffith, Christopher Rodriguez, Jerome Fleischman	First Draft

Table of Contents

Table of Figures

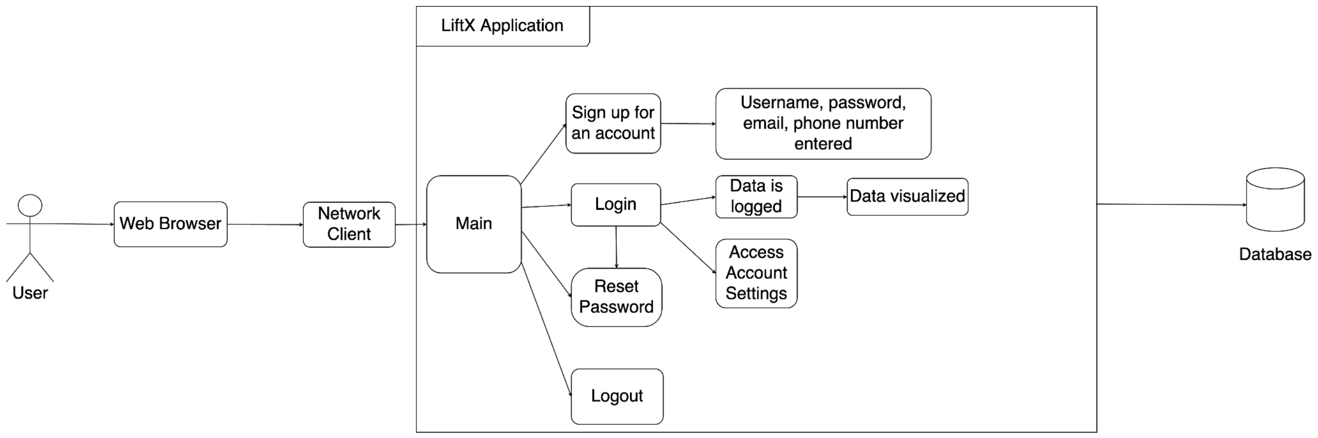
1. System Analysis

1.1 System Overview

This document describes the architectural design and specifications of LiftX The system uses a three-tier architecture, which includes a UI layer, business logic layer, and a database-centric layer For the UI layer, the interface that the user will be interacting with, we will mainly be utilizing the Code Sandbox environment which lends itself well to frontend development . The business logic layer includes the logic behind the application that allows for the inputted exercise data to output the proper progression The database-centric layer is used to store inputted information as well as interact with the business logic layer to create the suggested progression.

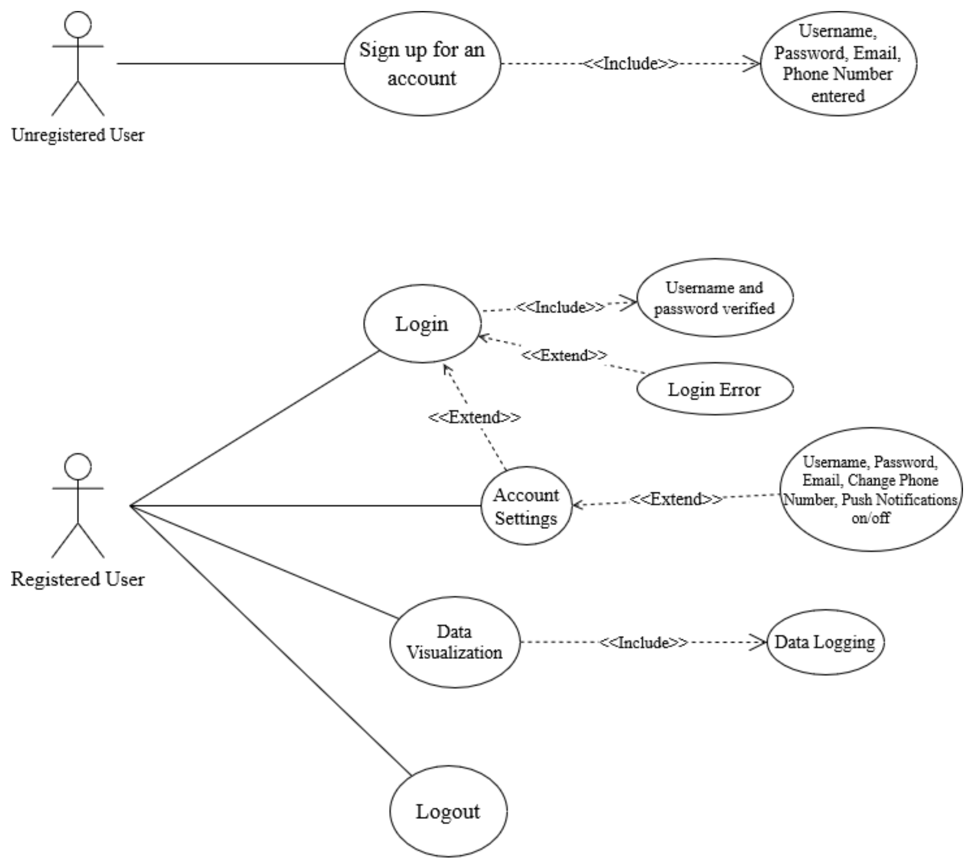
1.2 System Diagram

1.2.1 System Diagram for LiftX



- This system diagram gives an overview as to how the system components will work with one another along with the functional requirements from the LiftX Application.
- The user will connect to a web browser through either their smartphone or computer and they will require connection to a network client in order to access the application's website.
- Main is the general root of the system and that leads into the functional requirements that make the application what it is such as the ability to sign up for an account, login to an account, reset the account's password and logging out.
- The data from the application such as an account's username, password, email, and phone number will be stored into a database.

1.2.2 Use Case Diagram for LiftX



- This Use Case Diagram gives an overview of how two kinds of users will interact with the application and an overview of the functional requirements of the system.

1.3 Actor Identification

- There are two types of actors as of now. The first actor is the unregistered user. The unregistered user only has the ability of viewing the LiftX homepage and the ability to sign up for an account. However, they will not be able to access the features that a registered user would be able to access in LiftX.
- The Registered User will be able to access the services in LiftX such as the ability to log their weight data into the application and receive a visualization of their progress over time. They will also be able to edit their account settings at any time and have the ability to logout as well.

1.4 Design Rationale

1.4.1 Architectural Style

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

- UI Layer: This layer uses Code Sandbox for the front end of the application.
- Business Logic Layer: This layer includes the logic for the progression of weights for the user.
- Database-centric Layer: This layer uses the users' stored inputted data to interact with the business logic layer.

1.4.2 Design Pattern(s)

For our application we will primarily use the Adapter Design Pattern. Data from the database-centric layer will be pushed into a class in the business logic layer using an adaptor class.

1.4.3 Framework

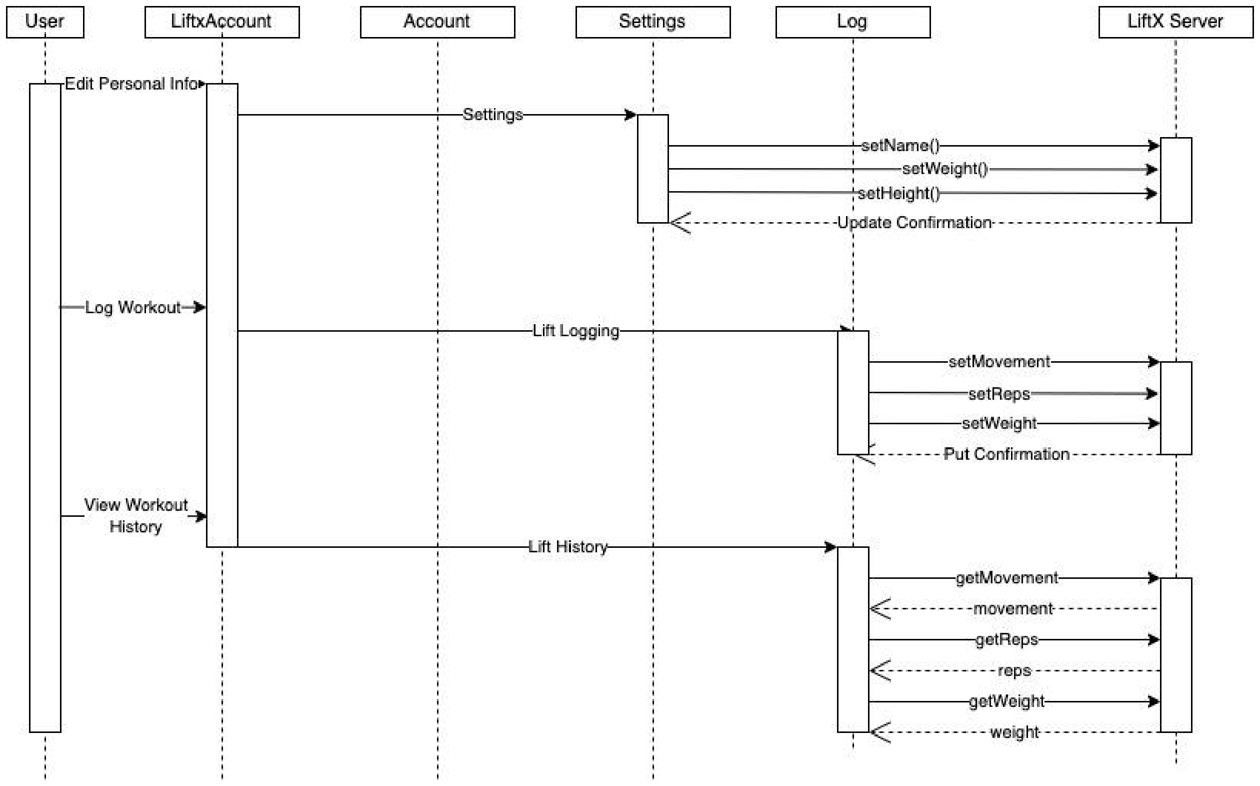
The Lift-X front-end will consist of a combination of HTML, CSS, and Javascript. Using these languages will ensure the cross-browser compatibility of our web application, a non-functional requirement of ours.

We will be implementing our backend in Java to allow for multi-threading of our server. It's important that our server can perform multiple tasks simultaneously without querying to allow for simultaneous data logging and display. Java is also very secure and has a plethora of open-source libraries to call upon which will allow for accelerated development.

JS
HTML
CSS
Java

2. Functional Design

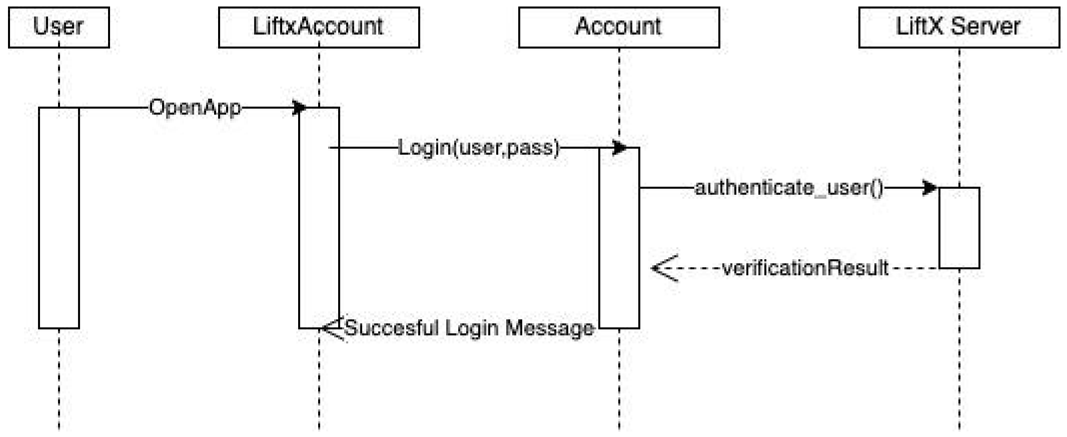
2.1 Updating Personal Info, Logging Workout and Viewing Workout History



- When the user edits their personal information on the Liftx page, the LiftxAccount, using its relationship with the Settings class, calls Settings.
- Settings holds most of the personal information variables attached to the user and the methods for the get and put calls to store and retrieve them.
- Settings executes its setName(), setWeight(), and/or setHeight() methods which make put calls to the LiftX Server. The server then returns a confirmation message.
- When the user wishes to log a workout in Liftx, the LiftxAccount class calls the Log class. The Log class holds the variables and methods specific to the storing and updating of workout information. The Log class executes its setMovement(), setReps, and setWeight() methods which execute the necessary put calls to store the new workout information. The Lifx Server then returns a confirmation message.

- Finally, when a user wants to view a previous log, the LiftxAccount class calls the Log class. The Log class executes its getMovement(), getReps, and getWeight() methods which execute the necessary get calls to retrieve the desired workout information.

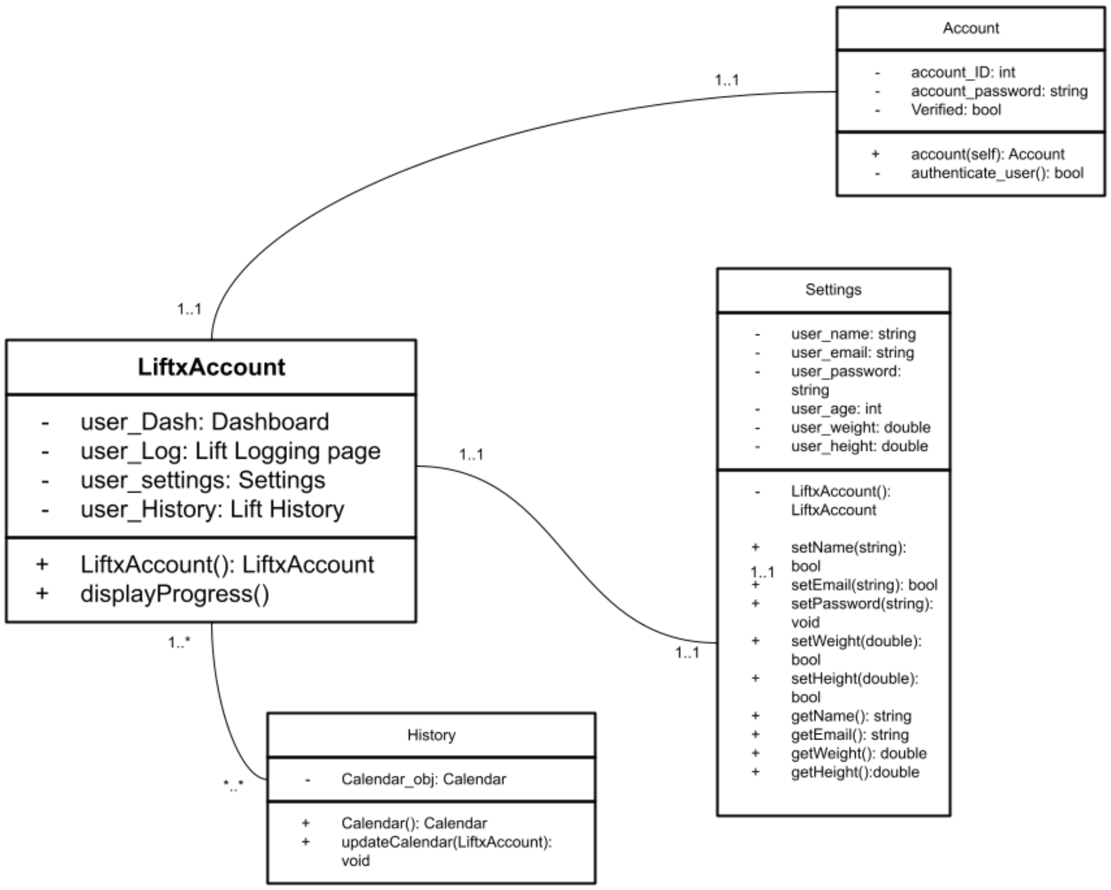
2.2 Logging into the Web Application



- When the user opens the web application on their browser and enters their login and password, LiftxAccount class calls the Account class which executes the authenticate_user() method calling the server to verify the user's login credentials.
- The Liftx Server returns a verificationResult to Account and Account returns a ‘successful login’ message to the LiftxAccount and user.

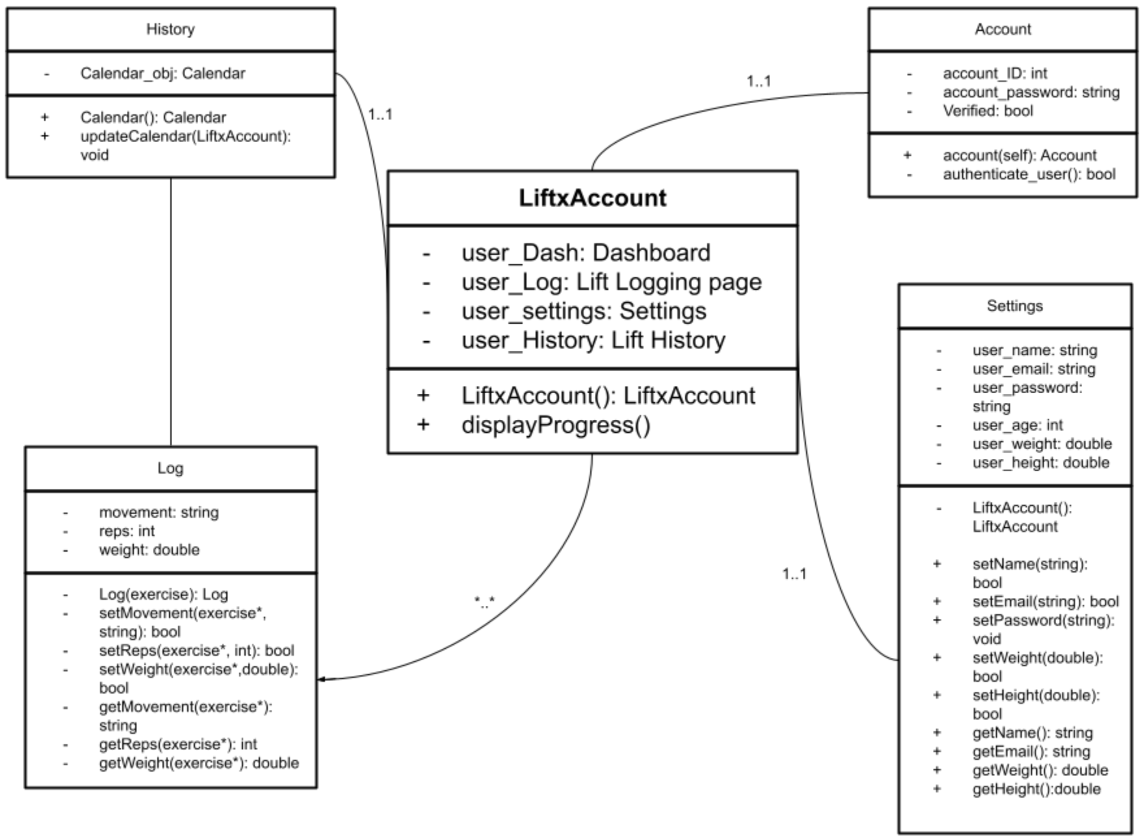
3. Structural Design

3.1 The following is a UML class diagram for the LiftX Account class which inherits the Account class and has association to the Settings class.



Structural Design Diagram

3.2 The following is a UML class diagram showing the associations of the LiftX Account class to History and Log



***UML* Class Diagram**

- As shown in the System Overview, recording lifts (Log) and tracking progress (History) is assumed to be requested by users with registered accounts (users).