# Complete Documentation

CSCI 201 Group: Fitness Web Tracker

Liana Zhu
Josh Carlito
Grant Yabuki
Scott Tang
David Cue
Kevin Vo
Nathaniel Lee

**Project Proposal:**

Names: Nathaniel Lee (nlee5082@usc.edu) , Grant Yabuki (yabuki@usc.edu), Scott Tang (ruit@usc.edu), Josh Carlito (carlito@usc.edu), David Cue (dcue@usc.edu), Liana Zhu (lczhu@usc.edu), Kevin Vo (kbvo@usc.edu)

Proposal: Our project proposal is to create a fitness app where users can log their workouts, set goals, and track their progress over time.

## High-Level Requirements:

Log-in:

- Users can create accounts and sign-in with email
- Guest users can use the application to test the features
- Include **forgot password** and **account recovery** functionality.

Features:

- Log Workouts
- Log Weight/Reps/Rest
- Notify/recommend users when to progressively overload
- Log Calories
- View Charts/Graphs of body metrics to view trends
- Table section to view metrics with the dates of when data was input
- Create goals and be able to view a goal line on a graph if applicable

Sign-in/Registration Page:

Simple login page with clear USC/Guest login options.
Option to link accounts with Google or Apple ID for convenience.

Dashboard:

3 Core Sections:

Body Weight Section: Input weight and view recent logs.
Calorie Tracker Section: Log calories and see daily progress.
Workout Section: Log workout sessions and view summaries.

Graph & Data Table Section:

Graphs: At-a-glance trends for weight, calories, and workouts.
Data Table: A detailed view of all logged metrics by date.

Simple Design:

Clean, responsive design with minimalist styling.
Intuitive navigation across key sections (Dashboard, Profile, Settings).

Technical Specifications:

Web Interface (12 hours)

- Login Page: This will contain a username field, a password field, and a login button. Forgot password button, create account button, and continue as guest button will also be included. Buttons to link to another account as a login option will also be added such as google and google.
- Guest users will be able to view all website sections, however they won't have the option to input any data to track their workout history.
- Upon verification, users will be taken to the Graphs and Data section, where a graph and a data table will be displayed for each of the corresponding categories of weight, calories, and workout. These tables and graphs will be based on the user's inputted/saved data if logged in or empty if the user is a new guest and has not entered any information in yet. Graphs will show trends in these categories e.g. weight increase/decrease, calorie intake increase/decrease by day, and workout intensity/occurrence (can be combined to be calculated by a workout "score") within certain timeframes. Data tables will show all logged data/inputs by date from a certain time selected by user e.g. ytd, 1 week, 1 month, 6 months, all.
- Dashboard/sidebar will be shown to user so that they can use it to navigate between different sections of the application. It will also display the current logged-in user/guest via username/guestname and an option to log out. Upon logout, user will be taken back to login page. Dashboard/sideboard will also include options to go to other sections of the program which will be the body weight, calories, and workout sections and will also include the graphs and data section too. The section user is currently on will be highlighted on dashboard.
- Body weight section will display an option to log user's weight to a date/time (either manually input by user or timestamped by application). There will also be a table displaying all of the user's previous logs of body weight. There will be edit and delete options for every entry in the table in case of user error. Table will be sorted from most recent to oldest logs by default. Options to sort oldest to newest, lightest to heaviest and vice versa can also be options. Users will be able to search for logs based on date/time (either one log at a specific date/time or a series of logs between two dates/times). Table will be separated on multiple pages to prevent overflow if too long. By default, table will only show logs up to 1 month prior, but can be adjusted by user to show later logs or more recent logs only.
- Calorie section will be implemented similarly except weight will be calories and default setting will show logs for current day only. Table will not expand to show earlier logs but earlier logs can be viewed by searching the day they wish to view and having that day's logs appear on the table. Can also log food eaten alongside calories and date/time as well as macros of said food such as protein/carbs/fat/fiber (will be optional fields). Summary of total calories/macros for the day table is on will also be displayed. Input option can also be streamlined so user doesn't have to input all fields when logging food/calories/macros/date by having custom foods and recipes user can add/save.

- Workout section will implemented similarly too except default setting for the table will show logs for current week and workouts will be logged and displayed as exercise/weight/reps/sets/duration/rest/date (some fields will be optional like duration and rest). Can also implement goal feature within this section so that users can input goals for an exercise whether that be reps/weight/set/duration or all. Thus, when the user is able to log the goal consistently/or for the first time user can be notified to increase the goal.

Database (10 hours)

- The database will consist of 7 tables: User, Exercise, Workout, WorkoutGoal, Calorie, CustomCalorie, WorkoutProfile
- User: userID, username, password, firstName, lastName
- Exercise: exerciseID, exercise name
    - Table for normalizing exercises
    - Fixed table, not updated by users
    - Maybe sort exercise alphabetically when inputting into table, so workouts in Workout table can be queried in a numerically sorted order through exerciseID
- Workout: workoutID, userID, timestamp, exerciseID, reps, sets, duration, rest, distance, speed, avgBpm
- WorkoutGoal: workoutGoalID, userID, exerciseID, isAccomplished, reps, sets, duration, rest, distance, speed, avgBpm
    - Every time a workout is inputted and the userID and exerciseID match, check if any of the features (reps, sets, duration, rest, distance, speed, avgBpm, etc.) match the goal set
- (body) Weight: weightID, userID, timestamp, weight
- Calorie: calorieID, userID, timestamp, totalCalorieIntake, proteinCalorieIntake, fatCalorieIntake, carbCalorieIntake, fiberCalorieIntake, customCalorieID
    - Everything besides calorieID, userID, and timestamp can be null. i.e. can select customCalorieID and leave every other calorie field blank
- CustomCalorie: CustomCalorieID, foodName, totalCalorieIntake, proteinCalorieIntake, fatCalorieIntake, carbCalorieIntake, fiberCalorieIntake
    - Not user specific, so a user can utilize a custom food item made by another user

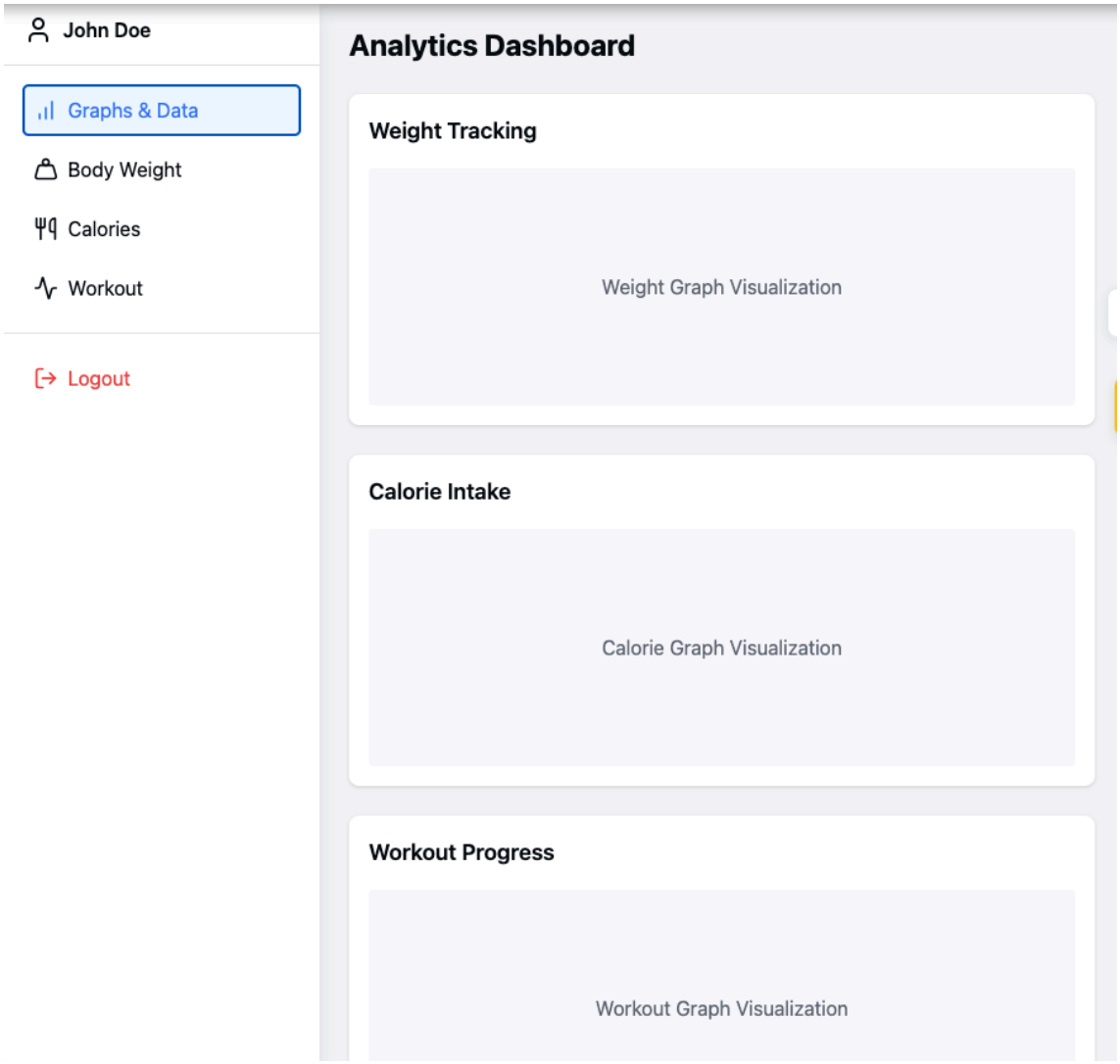
## Detailed Design:

**Hardware and Software Requirements**: The hardware requirements include an updated and working OS, such as Windows/Mac Laptop/PC, with enough processing power and storage. The software requirements to host the website include having an updated and working browser with a working web connection. To build the web application, the software requirements include having a working IDE like VSCode and Eclipse, or being able to run a Java program, specifically having a working Java Runtime Environment, with a working installed JDK and JVM, an installed and working Apache Tomcat Java Servlet software, a SQL database management system, and a working version control system like Git or Github.

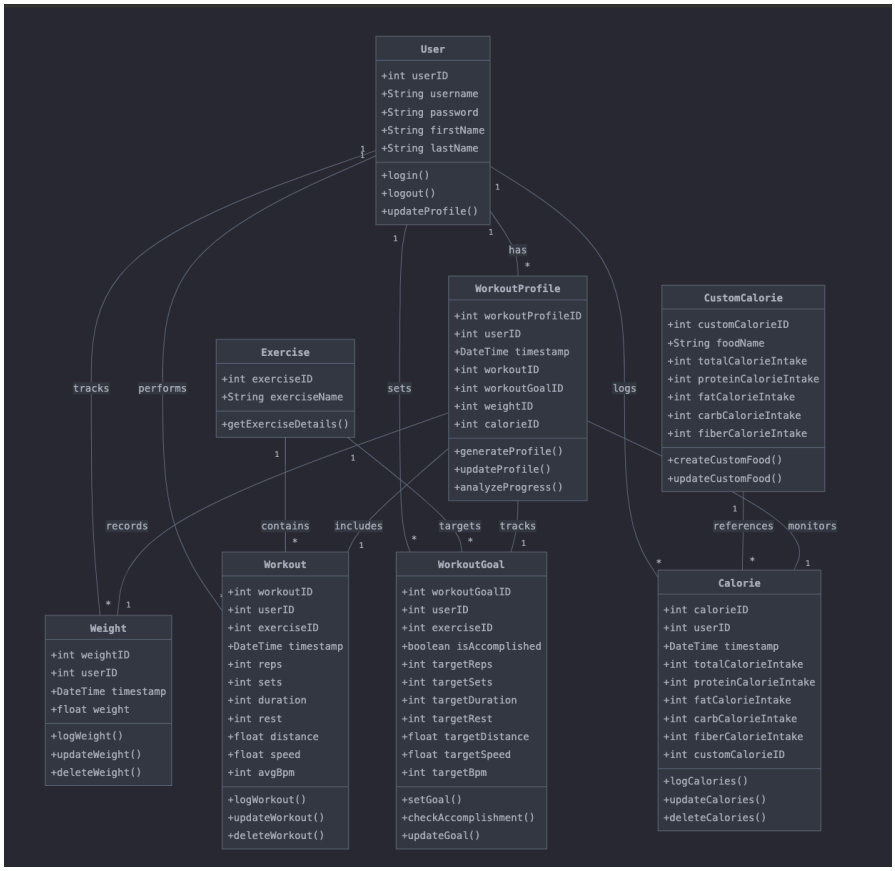
**Languages**:

Back-end - Java, Spring, Tomcat
Front-end - React.js, HTML, CSS
Database Management: SQL (MySQL), Microsoft Azure

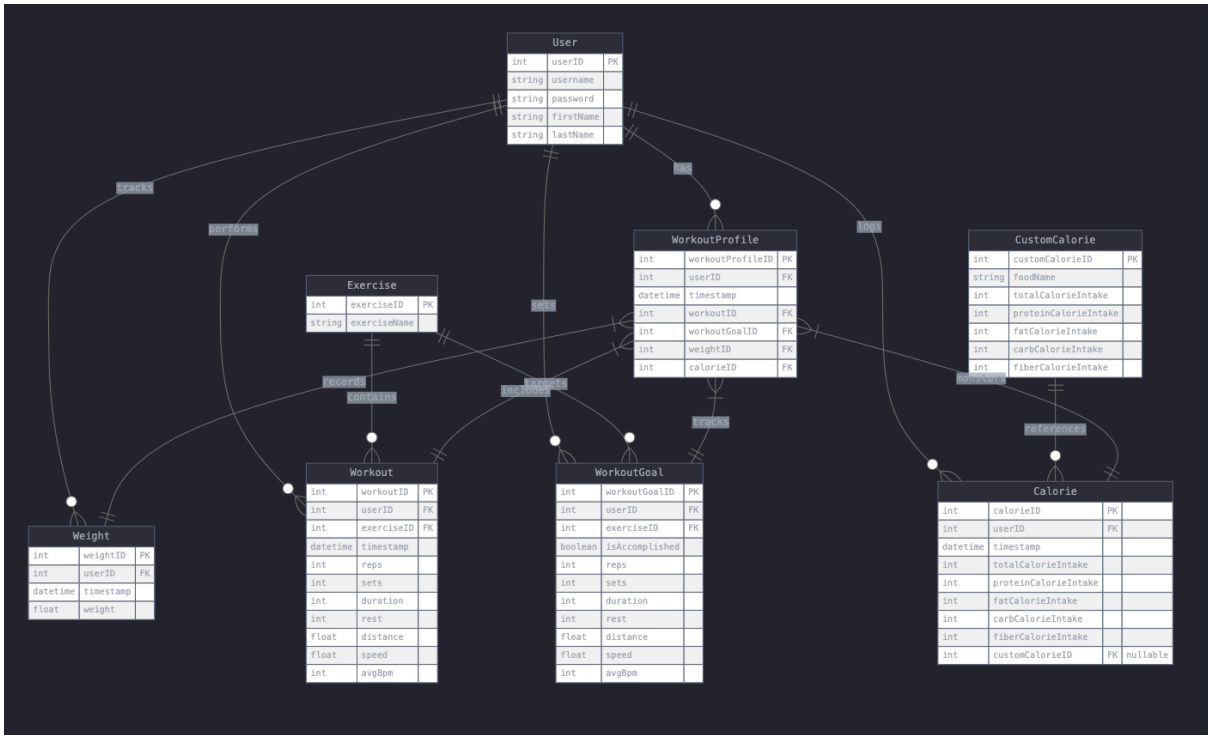
**GUI Design:**



## Class Diagram:




## Database Schema:



## Testing:

Name - Test name (Test type) - Description (roughly 1 sentence)

Liana - White Box Testing — test login functionality by trying to log in without inputting a username or password entry and not selecting the guest login option, an error message should be displayed
Liana - White Box Testing — test login functionality by trying to log in with a valid username but an invalid password that does not match, an error message should be displayed
Liana - White Box Testing — test data input functionality by inputting a negative body weight value, an error message should be displayed
Liana - White Box Testing — test data input functionality by inputting a negative calorie intake value, an error message should be displayed
Liana - Unit Testing — test the security of the program by attempting to bypass login or retrieve user data with an SQL injection, verify that the SQL injection fails

David - Black Box Testing - Try valid login and check that the app correctly advances to next screen.
David - White Box Testing - test login functionality with Google/Apple ID, verify that session is correctly started
David - Black box testing - add workout, check that it is correctly stored in user history
David - White Box testing - Enter forgot password section, check that app correctly stores newly created password.
David - Unit Testing - Write custom test case in Java to query for a nonexistent workout and ensure nothing is returned. Create a workout, then check that it is correctly stored and retrieved by program.

Nathan - White Box Testing - Users will correctly be notified when they should increase reps/weight with the implemented logic
Nathan - White box testing - plotting functionality should correctly calculate and visualize progress based on the user data
Nathan - Unit testing - the delete function for body weight and calorie tracking should correctly update the database and the corresponding results
Nathan - Unit testing - Filtering the dates for viewing/plotting data should correctly use the selected timeframe
Nathan - White box testing - Searching for workouts based on exercise type and all relevant filters return the relevant entries.

Josh - Regression Testing - After a system update, verify that all previously stored user data remains accessible and accurate by comparing it against a backup database.

Josh - Regression Testing - Verify custom foods remain usable for meal logging after database updates

Josh - Stress Testing - Simulate 100+ simultaneous users performing various operations to verify system stability and response times remain acceptable.

Josh - Stress Testing - Load test the database with synthetic workout data for 100+ users and verify that graph generation and data filtering operations are completed promptly.

Josh - Stress Testing - Test system performance when multiple users are simultaneously syncing workout data across multiple devices while performing database edits.

Kevin - Black Box - Test that the sidebar correctly highlights page user is currently on by switching between different pages and checking sidebar
Kevin - Black Box - Test that the sidebar correctly allows user to move to the page they select by using sidebar to move between pages
Kevin - Black Box - Check that the correct user/guest is displayed in sidebar upon login/logout by logging in and out with different users
Kevin - Black Box - Check that tables are split onto another page after extending past a certain length by adding lots of entries and setting table to display all entries
Kevin - White Box - Ensure user's data is saved upon logout or exit of application by checking database after logging out or exiting application

Scott - Black Box Testing — test the application's log-in page to ensure that both registered users and guest logins are handled correctly, validating the transition to the main dashboard.
Scott - White Box Testing — verify the functionality of the calorie tracking system to ensure users can only log valid calorie values, and incorrect entries prompt an error.
Scott - Unit Testing — write custom test cases to check the integrity of user goal tracking, confirming that each goal updates accurately based on logged workouts or calories.
Scott - Stress Testing — simulate a high number of entries in the workout history for a single user to evaluate the application's ability to filter and display data without lag.
Scott - Regression Testing — confirm that all user account data, including goals and workout logs, remains accessible and unaffected after implementing an application update.

Grant - Component Testing - Test the application for the entire process of a registered user to login and create 1 entry
Grant - White Box Testing - Test the application when a user inputs unexpected calorie values, making sure the code handles unexpected input values correctly
Grant - Unit Testing - Test the dashboard/sidebar to ensure that navigation between different section of the application works and user gets directly to the correct page
Grant - Unit Testing - Test for when the user inputs a goal for the workout section to ensure data is saved correctly and input is correctly reflected on the page
Grant - Black Box Testing - Test for when the user inputs more than 1 months worth of data and ensure that the user sees the correct dates when they adjust for dates they wants to see

## Deployment:

General Steps:
Step 1a: Deploy the application to AWS
Step 1b: Push changes into GitHub page
Step 2: Verify the data migrated correctly to the server
Step 3: Perform a full test of the application
Step 4: Redirect DNS and notify users about the updated server

Creating a Spring Boot Project with Spring Initializr:
Step 1 : Go to spring initializr to quickly bootstrap a simple spring boot web application.
After filling all details and selecting web dependency, click on Generate or ctrl + Enter. The zipped project will be downloaded in your system.
Before deploying the application, it is essential to configure your AWS account and set up the necessary tools, such as the AWS CLI (Command Line Interface).

Step 2 : Unzip the downloaded project and import it in your IDE.

Step 3: Adding a RestController to test our project.
Go to src > main > com > example > demo
Now create a new file named TestController.java beside DemoApplication.java file.
Create a basic controller with minimal code and endpoint to "/."

Step 4: Test the Application locally .
Just click on Run Java Button on top right on the VS Code. Now, go to browser and open localhost:8080.

Step 5: Prepare the final Jar file
Open the terminal and type (maven) mvn package to build a jar file
Once your build is successful, you will find the jar file in target folder. Copy the jar file and keep it in your system safely.

Deploy our Spring Boot Application on AWS:
Step 1: Open AWS Console and Create and Elastic BeanStalk Application
Click on Create Application.
Provide Application name and Application Tags(Optional).

Step 2: Scroll Down and Provide platform, platform branch, and platform version.

Step 3: In Application Code , Select upload your Code :
Give a Version Label, and select Local File and select the jar file saved earlier and click on NEXT .

Step 4: Configure Service Access.

Select existing service role as aws-elasticbeanstalk-service-role.

Select your EC2 key pair and EC2 instance profile to securely log in to your EC2 instances.

Now Click on Skip to Review.

Step 5: Adding Environment Variable

Make Sure to add SERVER_PORT and JAVA_HOME values and java_home variable.

Now Click On Submit.

Step 6: As soon as you click Submit, Elastic Beanstalk starts setting up your application environment.

It will take a few minutes.

Upon clicking on the link under domain, we will be directed to our deployed app.

<u>Deploy on github pages (Alternative option)</u>

Publishing from a branch:

1) Make sure the branch you want to use as your publishing source already exists in your repository.

2) On GitHub, navigate to your site's repository.

3) Under your repository name, click  Settings. If you cannot see the "Settings" tab, select the  dropdown menu, then click Settings.

4) In the "Code and automation" section of the sidebar, click  Pages.

5) Under "Build and deployment", under "Source", select Deploy from a branch.

6) Under "Build and deployment", use the branch dropdown menu and select a publishing source.

7) Optionally, use the folder dropdown menu to select a folder for your publishing source.

8) Click Save.