ActiveU Project Report

Project title : ActiveU
Group Members: Tyler Schmitz, Jared Osterhaus, Sarthak Paithankar, Mateo Ruby, Jose Martinez

Project Description: A 200 word summary of the project
ActiveU is a comprehensive fitness tracker, built to simplify the way you record, track, and create workouts. ActiveU allows users to record workouts and store them within their accounts. Users can either create their own workouts or use our website to get suggested workouts. Users can use our “Create a Workout” feature which allows users to enter a type of training modality, muscle group, or difficulty to get a variety of different exercises to choose between. For those who want to capture their sessions in real-time, the “Log a Workout” tool enables users to time their workouts, name them, and save them for future reference. This ensures you always have a clear record of your fitness journey. To keep Users motivated we implemented a friends system where users can follow other users to see their progress and workouts. Users can search for others by their username or look them up by looking at other users’ friends. This social aspect turns fitness into a shared experience, making it easier and more enjoyable to stay on track. Future implementations into the friends system include a reaction feature where users will be able to react to others’ workouts with emojis and leave comments to share encouragement.

Project Tracker - GitHub project board: https://github.com/users/LuminFX/projects/3

Video: https://youtu.be/x3ZAL-9mUW0

VCS: https://github.com/LuminFX/ActiveU

Contributions:
Tyler: My biggest contribution was creating the friends page, and adding all of the functionality for it. This functionality includes adding and removing friends, canceling sent requests, accepting or denying incoming friendships, and linking the displayed friends to profile pages that can be opened by the user. To add these features, I spent a lot of time with Handlebars, Bootstrap, ExpressJS, and PostgreSQL. I also created and implemented the account page, where users can see their own workouts and friends. In addition to this, I did most of the merging required for our repository on github.

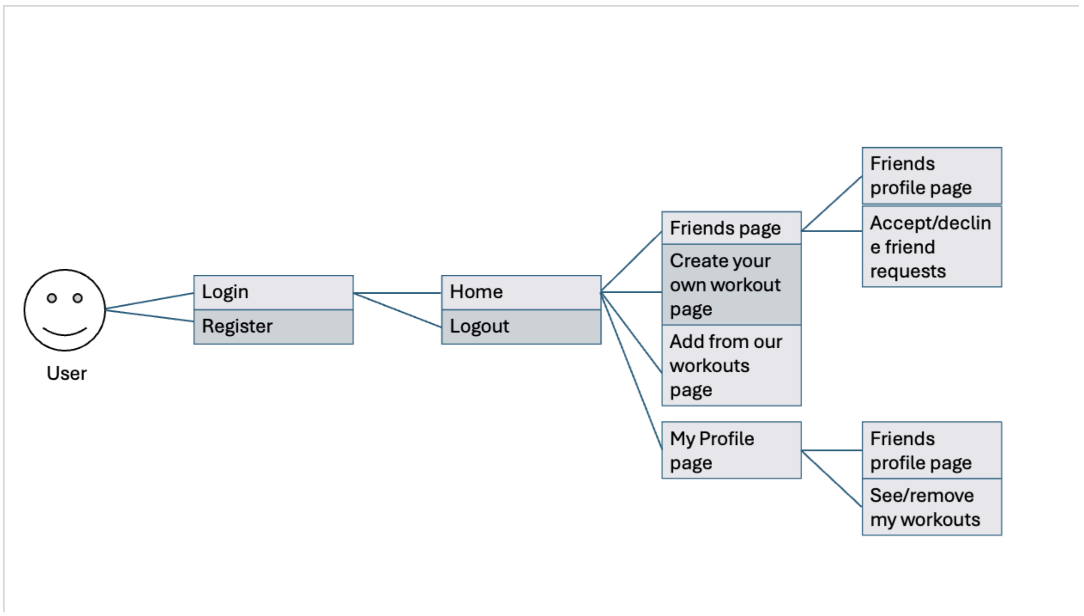
Jared: My most significant contribution was setting up the add a workout page by pinging ninjaAPI’s “Workout api” to fetch different workouts depending on the users input. This page was clean and appealing to the user such that they could find new or existing workouts they want to add to their profile. It worked by presenting the user with two categories which were required to complete a request by the API and I would put their results into a request and present the JSON I would receive back to the user. On the backend, I had also set up majority of the projects get and post requests to make sure each page was rendered correctly along with keeping the team on track with our handlebars/bootstrap format.

Sarthak: I implemented most of the front-end design for the website which includes the login, logout, registration, home, profile pages. I also worked to create the profile page feature which displays a user's workouts and friends. To implement these features I used tools such as bootstrap and handlebars for dynamic functionality in the front-end and express for nodeJS to create routes for querying the database to grab user information.

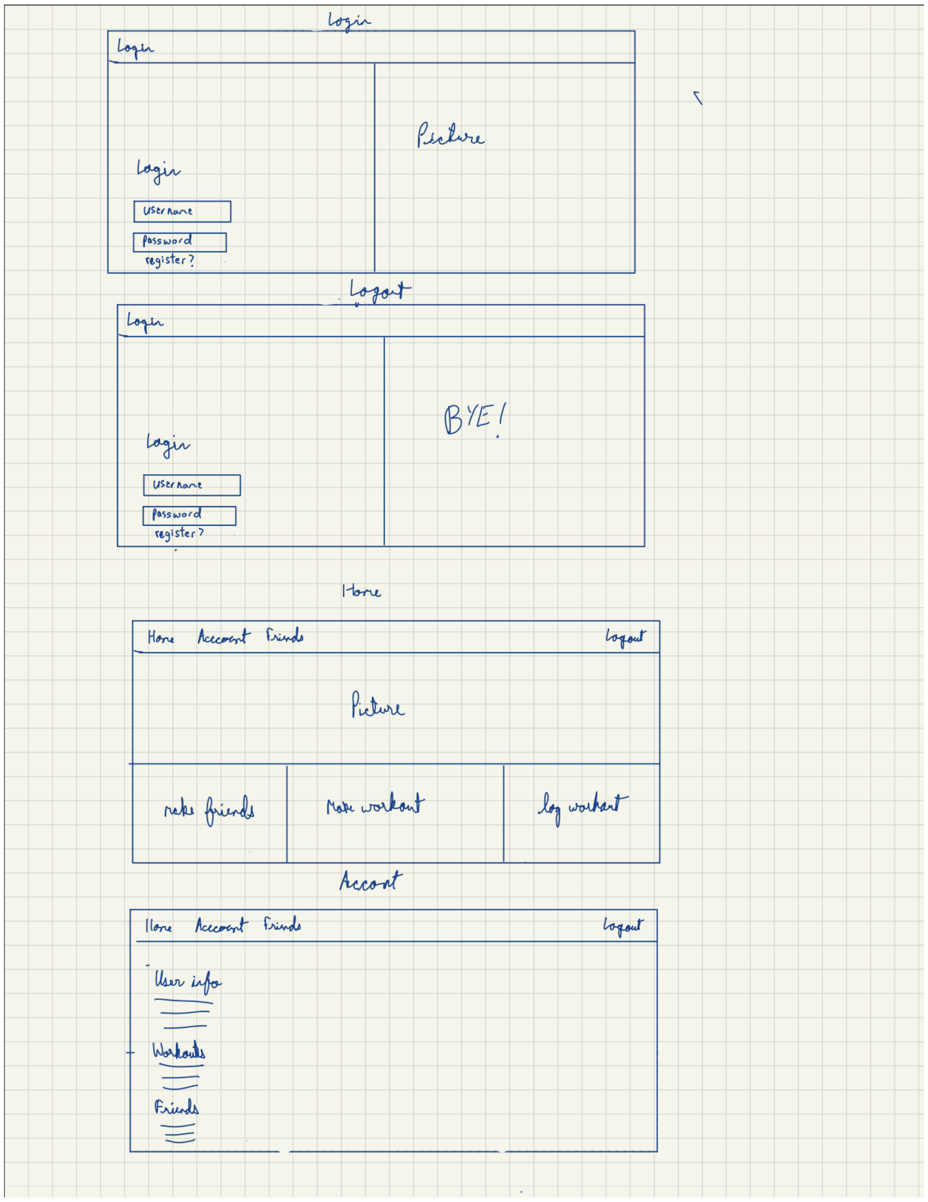
Mateo: I made the Create Workout page, I made a timer with Start, Stop, and Pause buttons to help users track workouts on their own. I used javascript to work with the user inputs so that they could name and save their workout. To style the page to look nice, and keep a black and yellow theme, I used HTML and CSS. I also used HTML and CSS to style the addWorkout page, and then javascript for when the user wants to add that workout and click the add button for a specific workout. I used SQL to actually store the workout data in a specific way in the database, with the name and how many sets and reps they did of that workout, which was later changed to the name of the workout and the type of muscle it hits.

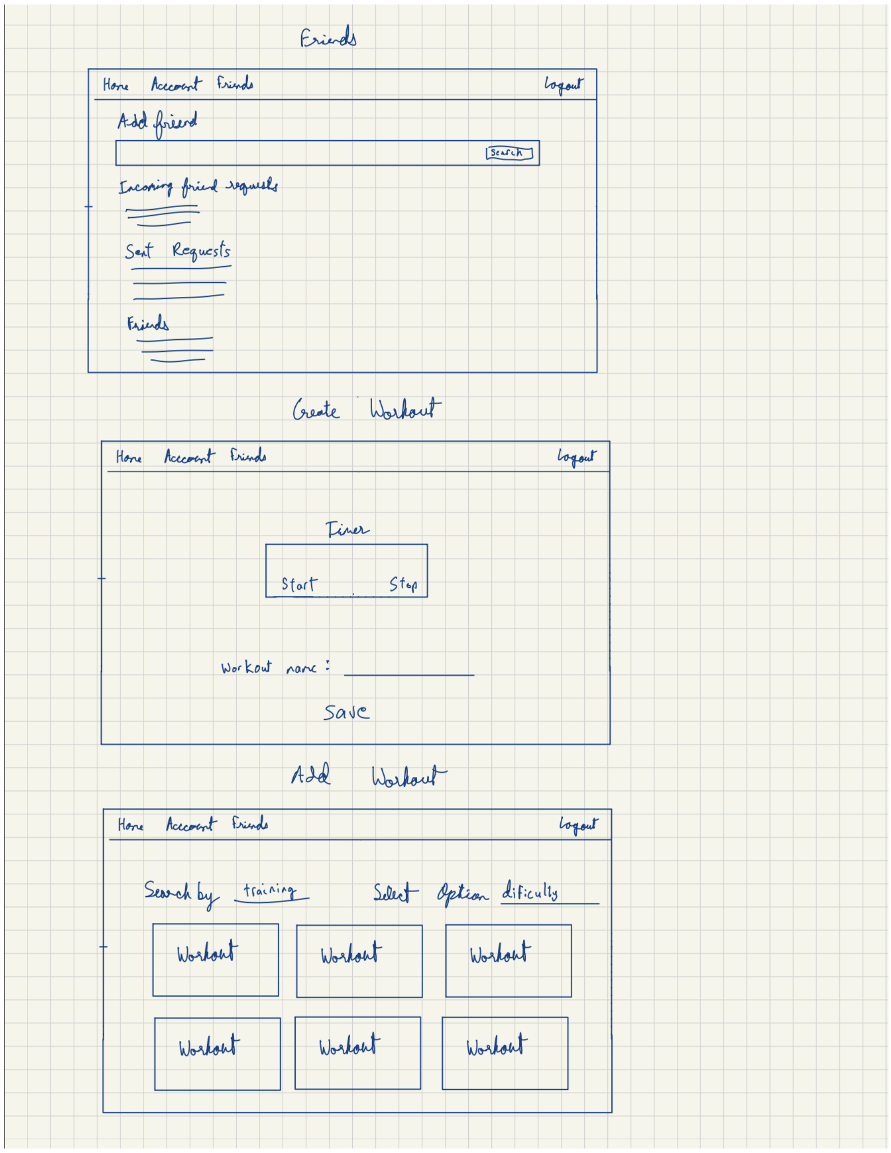
Jose: The bulk of my work consisted of the behind-the-scenes SQL database foundations, mainly the creation of important tables & relationships along with modifying their corresponding JS API routes. Storing user data and unique features such as friend requests, the creation of workouts, or other future tools depended on the implementation of these databases. I had also helped with the creation and the upkeep of the database Chai test cases that were introduced in a previous lab. These were important to ensure that everything in the backend was functioning, and any inconsistencies such as being able to register an invalid email were eliminated. Without these test cases, the project would've been led astray even if things appeared to be working properly.

Use Case Diagram: You need to include a use case diagram for your project. You can build on the use case diagram you created in the proposal. If you built a complete use case diagram for the proposal, you can include it as is.



Wireframes You must include all the wireframes that were created for the project. It is expected that you have at least one for each page. They can be photographs of hand-drawn images.





Test results: In Lab 11, you created a Test Plan. You need to include the test results and observations in the project report. Refer to [this] for more information

1) Test Logging a Workout in the log Workout page

Test Case 1: Used the dropdown menu to search for a strength workout and chose the chest category.

Test Case 2: Searched for bench press and added a 15-minute workout.

Test Case 3: Attempted to log a workout with an invalid input and checked that the error message displayed.

Results:

Test Case 1: The page displayed a list of chest workouts, and one of them was bench press.

Test Case 2: The bench press workout was added to the workout database and showed a 15-minute workout.

Test Case 3: An error message displayed when an invalid input was provided.

2) Test Create Workout Page

Test Case 1: Started the timer and paused it 15 seconds in.

Test Case 2: Saved the workout after pausing it as a 15-second running workout.

Results:

Test Case 1: The timer stopped and displayed the time of 15 seconds and 0 minutes and hours.

Test Case 2: The workout was added to the workout database and showed a 15-second running workout.

3) Testing Friends Functionality

Test Case 1: User_1 sent a friend request to User_2.

Test Case 2: User_2 accepted the request.

Test Case 3: User_2 checked User_1’s workout history, and vice versa.

Result:

Test Case 1: User_2 was able to see that User_1 sent a friend request, and User_1 was not able to see User_2’s workout history.

Test Case 2: User_2 was added as a friend of User_1, and vice versa.

Test Case 3: User_2 was able to view User_1’s workout history, and User_1 was able to view User_2’s workout history.

Deployment: https://activeu.onrender.com/