Title:

GymGo by GymTendo

Who (Name: github user):

Ethan Wang: etwa5465, Emily Zabron: ilike2code87, Isaac Mitchell: ismi5726, Kevin Wang: kevwang02, Emma Cotrell: EmmaCotrell

Project Description:

GymGo turns your workouts into a social and motivational game where you earn XP for every exercise, complete daily challenges, and track progress across both global and friends-only leaderboards. Through this app, we hope to promote healthy habits by gamifying regular gym visits and highlighting the social aspect of the gym by creating a community rooted in shared development and gym motivation. A built-in friend network lets you send and accept requests, compare stats, and encourage friendly competition through the friend's leaderboard. Working with other users, you can pool your XP to defeat fitness bosses and unlock shared rewards. Daily quests and suggested workout programs add variety and extra XP incentives, while providing a more structure to your daily workout flow. We've also implemented detailed exercise logging (reps, weight, duration) which creates a comprehensive workout history and stores data on your personal health journey. The clean, Bootstrap-powered Handlebars interface works seamlessly on desktop web applications, with intuitive registration, login, and easy navigation to Home, Leaderboard, Friends, Boss, Quests, and History pages. With minimal setup, GymGo keeps you accountable, engaged, and connected to a community that helps you reach your fitness goals together.

Github Project Board:

https://github.com/orgs/Gymtendo/projects/1

Video:

https://drive.google.com/file/d/1HVcSFSVzmCji_A5Wg1gb32QEzZpV9vzF/view?usp=sharing

Version Control:

https://github.com/Gymtendo/GymGo

Contributions:

Ethan Wang: Architected and implemented GymGo's core leaderboard and underlying PostgreSQL schema, defining tables like Accounts, Exercises, and UserExercises, and crafting efficient SQL queries to power both global and friends-only rankings. Using Node.js with Express and pg-promise, integrated these queries into Handlebars views for real-time ranking displays. Collaborated closely on cross-feature testing and refined our presentation and product flow, optimizing both backend performance and front-end rendering. Refined presentation elements, creating a logo for the application and created a detailed README file outlining the project.

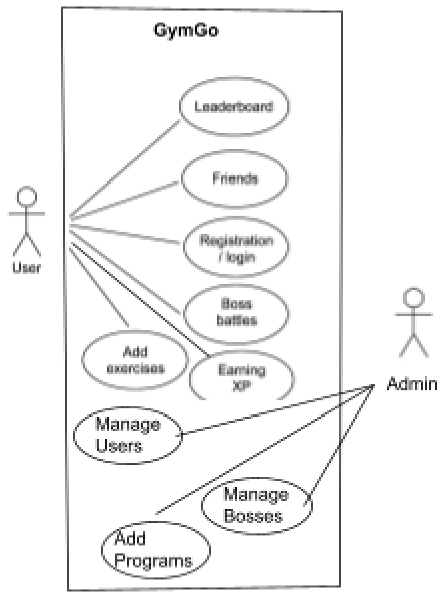
Emily Zabron:
I worked on integrating the PostgreSQL database with the register and login/logout features. It navigates users to the homepage upon login and stores their data in the leaderboard. I used partials and Handlebars to display information cleanly and JavaScript for the index file. I also designed different approaches to the frontend register, login logout UI. I helped debug errors like faulty leaderboard routes and database connection issues. I wrote all six unit tests to prevent duplicate usernames and ensure a smooth experience. The tests verify database storage and navigation after login. I used Mocha, Chai, Docker, and JavaScript for testing.

Isaac Mitchell: Responsible for much of the initial database, server, and website setup that we used to get things off the ground. Also created the friends page and system, which allows for pending requests in both directions and managing pending requests with minimal storage or processing overhead. Additionally authored many other small improvements, like improved navbar styling and redirecting the user back to their desired page after logging in. Finally, was responsible for repository maintenance, including project board improvements, managing branches, and squashing merge conflicts.

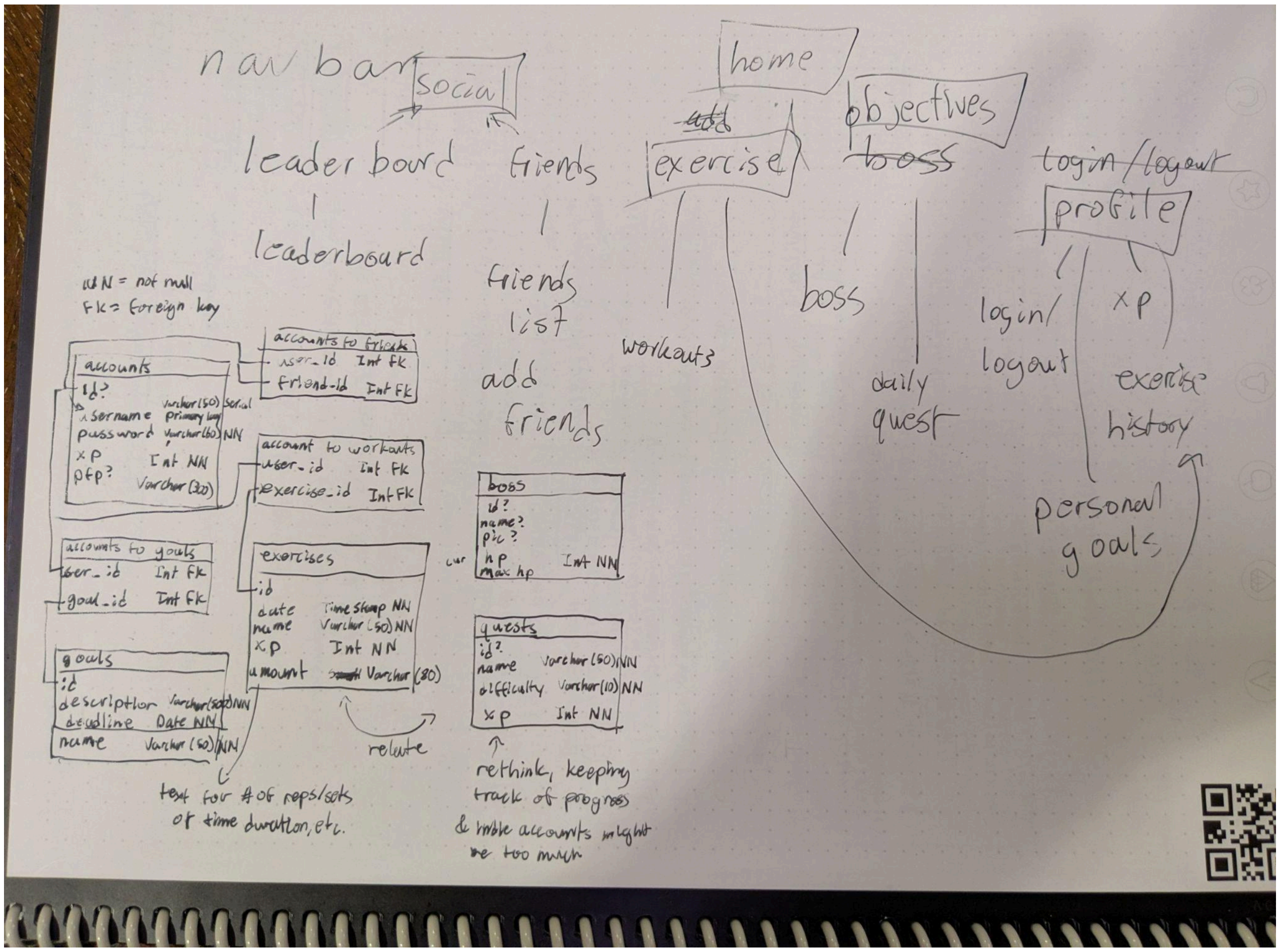
Kevin Wang: Drafted the initial navbar layout and planned early webpage features. Built frontend UI and backend logic for boss and quest routes, integrating users' XP from the database. Designed responsive progress bars and a dynamic navbar for smaller screens. Created reusable Handlebars partials (header, footer, message, navbar). Helped design the PostgreSQL schema and table relationships. Maintained informational comments on index.js server file for all routes and middleware utilized. Regularly debugged code and reviewed teammates' branches before merging. Deployed web and database servers on Render and wrote a function to reset the database as needed.

Emma Cotrell: Implemented the "Add Exercise" button and recommended workouts, with real-time updates to the user's XP on both the leaderboard and user profile. Integrated exercise submissions with the database to store workout history and track XP gains. Designed and built the user profile dropdown displaying the username, XP, workout history, and a logout option for easy access. Additionally, stylized the login, register, and logout account pages to improve the visual design and user experience.

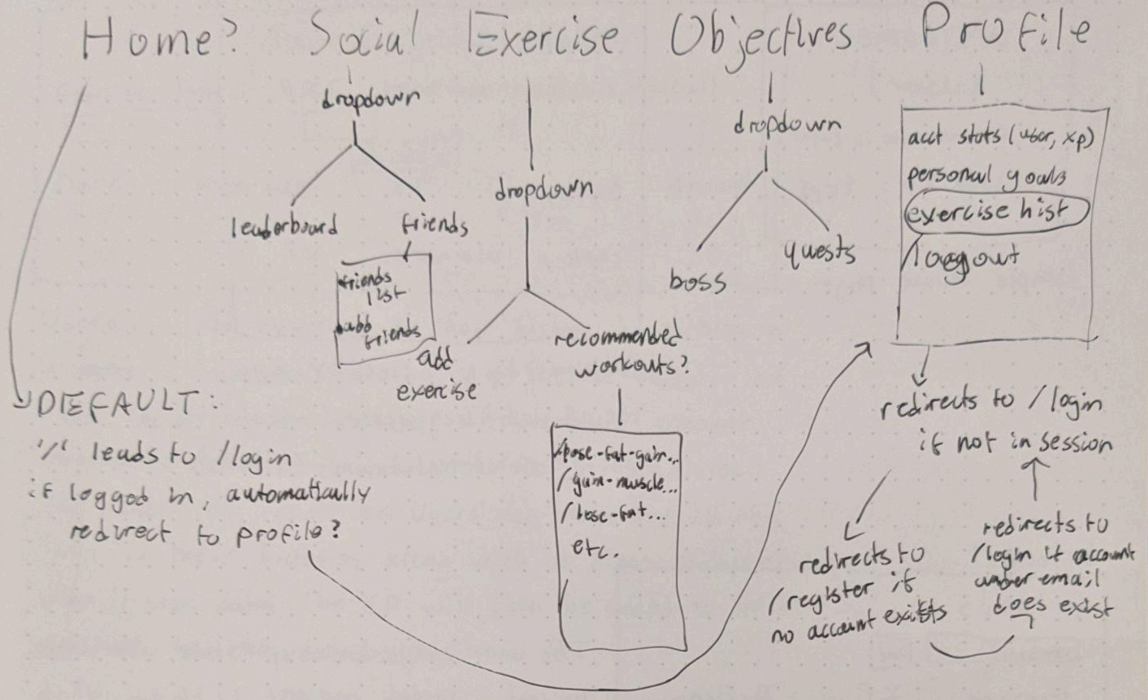
<u>Use Case Diagram:</u>



<u>Wireframes:</u>

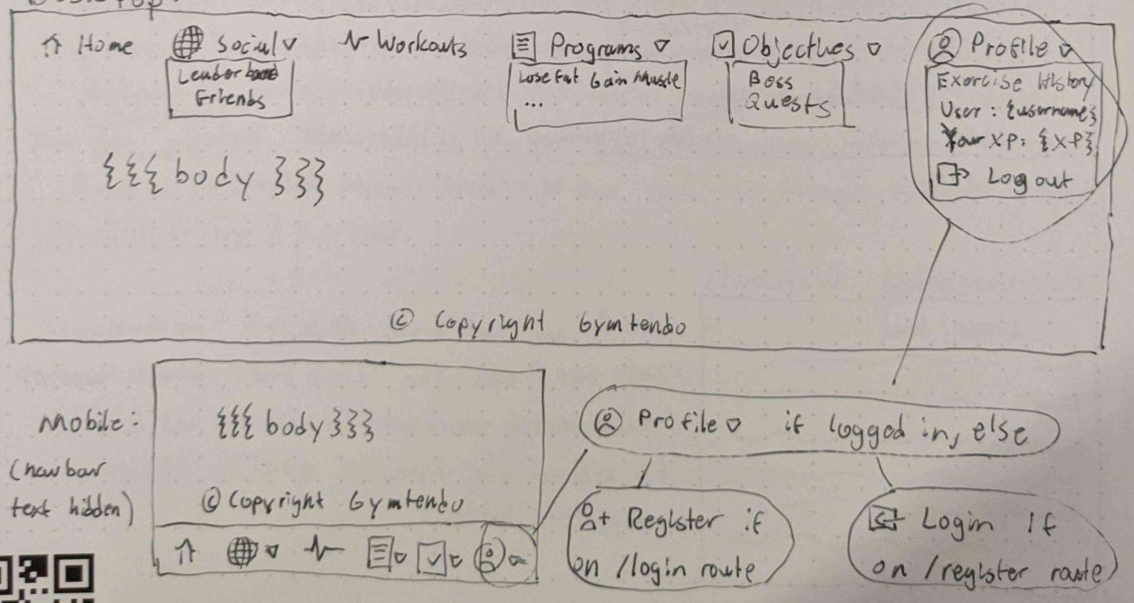


$\boxed{\text{boxed}}$ = functionality on webpage (ex: /friends has a 'add friends' form)

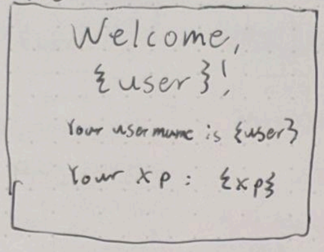


General Webpage Layout (header, footer, nav partials)

Desktop:

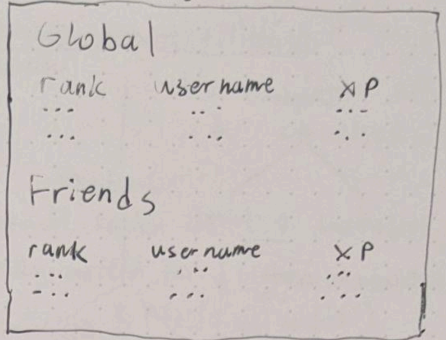


Home:



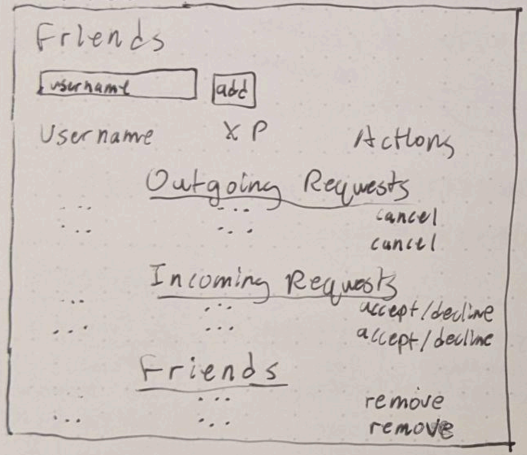
Simple welcome page

Leaderboard:



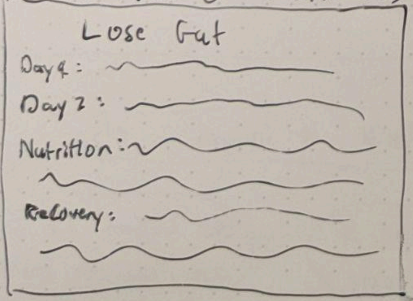
Displays top 10 users by xp for all users registered (Global) and for all current Friends (Friends)

Friends:



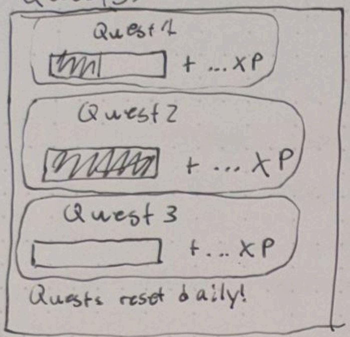
The user can search up other usernames to send a friend request to them, which will appear in the current user's "outgoing requests" and the prospective friends' "incoming requests". The request can be rescinded by the sender and accepted or rejected by the receiver. If accepted, they will appear under "Friends" and can be removed in the future

Recommended Workouts



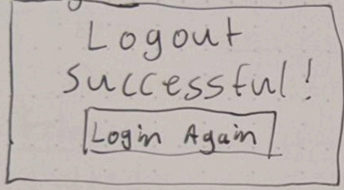
The "Lose Fat Gain Muscle," "Gain Muscle and Fat," and the "Lose Fat" pages are all recommended workouts, routines, and habits to achieve the specific goal in their respective titles.

Quests:



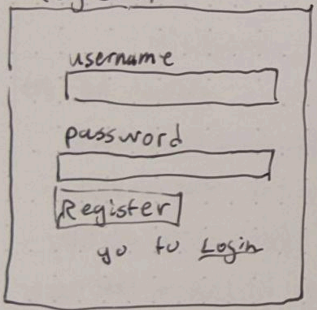
Users can get xp from quests by completing the quest directives. Whenever they make progress, the bars will update using info from the database. The user will only receive the quest xp after completion & visiting the quests page. If the user logs in the following day, all quest progress will be reset.

Logout:

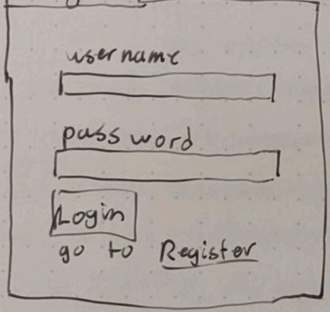


Displays a simple logout confirmation and destroys user session. Provides button if user wants to login again

Register:

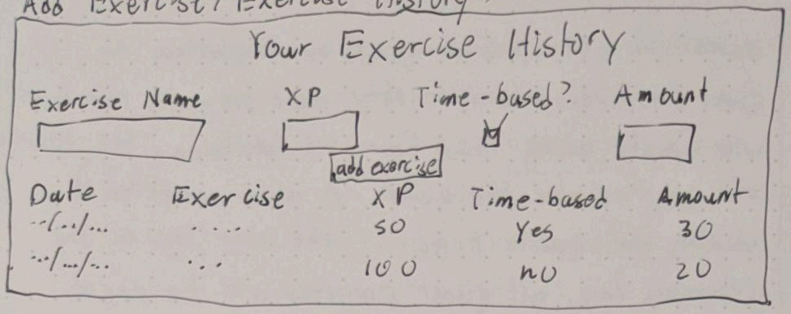


Login: (DEFAULT PAGE)



The Register and Login pages are visually similar. Register creates a new user in the database with the entered credentials (after hashing the password) while Login checks the entered credentials with those in the database and creates a session if they match.
Additionally, Register requires the username to be 3-50 characters and the password to be 8-50 characters. If a user attemps to register under an existing username, they will be prompted to login instead.

Add Exercise / Exercise History:



Your Exercise History

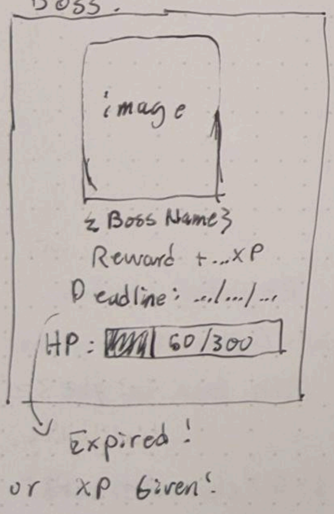
Exercise Name	XP	Time-based?	Amount
		☑	

add exercise

Date	Exercise	XP	Time-based	Amount
../../...	....	50	Yes	30
../../...	...	100	no	20

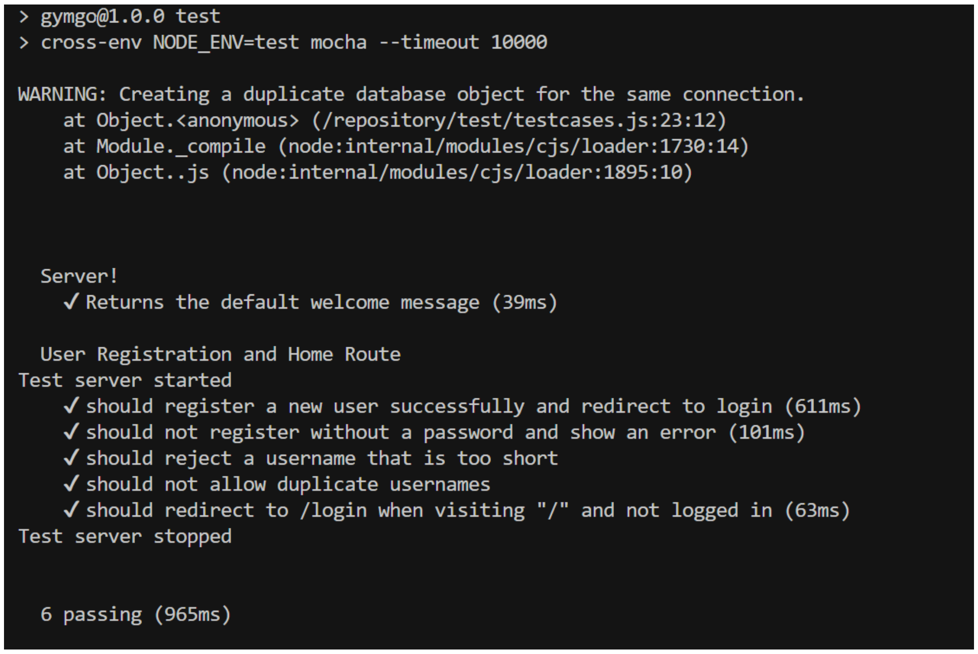
Users can add exercises to their history by typing in a name, XP amount, checking if it's time-based or not, then inputting an amount. If time-based, amount represents the # of minutes. Otherwise, it signifies the number of repetitions. Users can then see all this info in their histories along with a time stamp of when it was added. The XP will then be added to their account in the database.

Boss:



whenever users gain xp, the boss's global hp pool is drained. If the boss's hp reaches 0 before the deadline, all users will receive the reward xp. Otherwise, if the deadline passed or the boss is already defeated, a message will appear informing the user above the hp bar and the image will be grayed out.

Test Plan Observations:



```
> gymgo@1.0.0 test
> cross-env NODE_ENV=test mocha --timeout 10000

WARNING: Creating a duplicate database object for the same connection.
    at Object.<anonymous> (/repository/test/testcases.js:23:12)
    at Module._compile (node:internal/modules/cjs/loader:1730:14)
    at Object..js (node:internal/modules/cjs/loader:1895:10)



  Server!
    ✓ Returns the default welcome message (39ms)

  User Registration and Home Route
Test server started
    ✓ should register a new user successfully and redirect to login (611ms)
    ✓ should not register without a password and show an error (101ms)
    ✓ should reject a username that is too short
    ✓ should not allow duplicate usernames
    ✓ should redirect to /login when visiting "/" and not logged in (63ms)
Test server stopped


  6 passing (965ms)
```

The 6 tests that were written and implemented into our code test features as users log into our product. While writing these tests, edge cases were considered, such as duplicate usernames, weak passwords, etc. The first unit tests checked for duplicate usernames, too-short usernames, and weak passwords. Then, upon registering, we tested to make sure our users get sent to the login page immediately and are displayed the welcome message. Another test made sure in the case of no password created, an error will display. Finally, a unit test also checked that users were being redirected to the login page if not logged in and trying to access features.

Link to Deployment Environment:

https://gymgo.onrender.com

Link to README:

https://github.com/Gymtendo/GymGo/blob/main/README.md