Shane Wilkerson

Dante Carrizales

CMPS 3390

App 2

Communication: Discord

Ideal users: Gym lovers

Main task: Review gyms

Must have features: User profiles, gyms, and reviews

Framework and database: React Native with expo and sqlite

We want to create an app where gym lovers can review different gyms and talk about all the different amenities that the gym offers. Users will be able to rate it out of 5 dumbbells (stars), and any comments they have about the gym.

First page - be able to look up gyms and leave reviews

Second page - see all the reviews you have left

Third page - profile settings

Data base - user, gyms, reviews

User - username, userID(Maybe), email, password

Gym - Name, address, number of reviews(?), avg review rating

Review - string, master key from userID

Environment:
```
# Install Homebrew (if not already installed)
 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Node LTS (20.x) and Watchman
brew install node@20 watchman

# Check versions
node -v
npm -v
watchman --version
git --version
```

I used this to create structure for the app
```
npx create-expo-app@latest .
```

React Navigation
```
npm install @react-navigation/native @react-navigation/bottom-tabs
```

Required dependencies for navigation
```
npx expo install react-native-screens react-native-safe-area-context
```

SQLite database (Expo-managed)
```
npx expo install expo-sqlite
```

So this gives us:
Expo SDK 54 (React Native core)

React Navigation Tabs (for Home / My Reviews / Profile)

SQLite local storage (for saving gym reviews)

Consistent Node/npm versions across devices

Use this to run: npx expo start, then i hit i for ios simulator

For you just downloading it,

```
git clone https://github.com/ShaneWilkerson/3390Project2.git
cd 3390Project2
npm install
```

npx expo start

This SHOULD work

Added 3 screens inside the app/tabs folder with simple headings and colors
- index.tsx (this is like the home tab)
- reviews.tsx
- profile.tsx

Set up for SQLite

Install the SQLite package
- npx expo install expo-sqlite

Created a database.js builds the data structure of the app where it stores and reads information

Edited inside the app/_layout.tsx with some imports and exports.

Exported functions from the database.js file to each screen at the top

Finished Features

Login

- Functional login screen that is attached to the SQLite database file
- Includes working sign-up and sign-in buttons

Profile

- Displayed username changes depending on who is logged in
- Functional logout button

Home

- Welcome message changes to fit logged in user
- Gyms in database appear as clickable cards that display the number of reviews attached to each specific gym

Reviews

- Review creator (layout only)

Planned Features

Profile

- Button that allows users to change their username
- Button that allows users to change their password
- Delete button that wipes a user from the database
- A dynamic join date that is defined when users sign up
- A gym review counter that updates depending on how many reviews a user posts

Home

- Gym cards capable of opening up their reviews when clicked either in the same view or a separate view that take over the home tab
- Review number for each gym updates according to total reviews posted

Reviews

- A function that matches the inputted name with the gym's name in the database to attach the review to that gym's entry
- A post review button that attaches the review to the chosen gym's database entry (potentially saved as a string)
- A text tag that displays the name of the user who posted the review

Bottom of database starting from load Reviews

10/19

- Downloaded async storage
    - npx expo install @react-native-async-storage/async-storage

Updated profile tab -

- The profile screen is responsible for showing which user is currently logged in and allows them to log out.
- When the screen loads, it runs a useEffect function that reads the currentUser data from AsyncStorage, which was saved when the user logged in.
- That information is stored in the username state variable and displayed on the screen.
- If the user taps the Logout button, the app removes their saved data from AsyncStorage and uses router.replace("/login") to take them back to the login screen.

- The layout uses a green header, a light gray background, and matching colors from the global colors.ts file for consistency. The code is written in a simple and clear structure using React Native functional components.

- useEffect()
    - Loads user info from AsyncStorage when screen opens
- username state
    - Stores and displays the current username

Login page -

The Login screen is the first page that appears when the app is opened.
It allows the user to either log into an existing account or create a new profile.
This screen uses two text input fields for the username and password, along with two buttons — one for Login and another for Create Profile.

When a user enters their information and presses Login, the app calls the profileLogin() function from the database file. This function checks the local SQLite database to see if a profile with that username and password exists.
If a match is found, the user data is saved to AsyncStorage under the key "currentUser", and the app navigates to the main tab section using the router.replace('/(tabs)') command.
If no match is found, an alert appears saying that the profile could not be found.

When the user presses Create Profile, the app runs the createProfile() function, which adds the new username and password to the database. After that, the user can log in using their new credentials.

The screen uses the shared color palette from colors.ts, which gives the page a consistent look across the app. The background is light gray, the buttons and title use the mint green color, and the text inputs have white backgrounds for clarity.