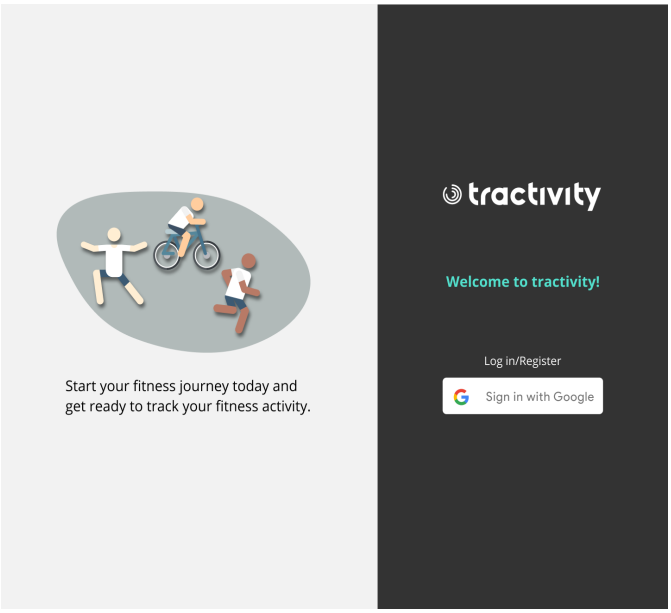
Home

Fitness tracker stage three - Due 10pm Wds May 12. Spring 2021



Tutor's video of the complete Fitness Log app.

Fitness 3 Design spec drawings

Tutor's solution to Fitness 2; fork if you want!

You may do this assignment in groups of at most three. You do not need to be in the same group you had for Fitness 2.

In this project we will make users login to the Fitness Log,using third-party login with Google. This is the final step required to make our app really usable. It will let us to have a profile for each user (not everyone is named "Joey"!). And, it will let us include the user the exercise log records in the database, so that when a user visits the app, they can see and add to their own exercise logs, and nobody else's.

We'll do this in two main stages. In steps 1-5, we'll get login working, get user IDs and basic profile information from Google, and make sure that a user can't get past the "splash page" unless they are logged in. In the

user can't get past the "splash page" unless they are logged in. In the remaining steps, we'll add userIDs to the database, change all our queries to use them, and change the name in the greeting on the main page.

Background

This assignment is related to the material in the lectures on Sessions, Login 1, and Login 2. There are textbook readings you can look at with those lectures; also the two comics on login might be helpful for getting an idea in your head before diving into the details.

Discussion sections in the week of 5/3 will both be pretty assignment-specific. One will go through the starter code in detail. The other will go through the tutor's code for Fitness 2. Once you understand these both well enough to combine them, you will be in very good shape.

The Net Ninja has a much more hands-on but still conceptually clear set of videos on how he set up OAuth2. Obviously, he is not doing our homework assignment (he has his favorite editor, his database, his style of how to set up Express, he's using Google+ which no longer exists, etc.), but he is doing something quite similar.

Details

1. To begin, let's get familiar with login code for Node.js by forking Fitness 3 starter code from "Teams -> ECS162 Spring 2021". It won't work unless it has credentials from Google. Rename the

   demo to be the name you'll use for your Fitness 3 project; this becomes part of its URL, and Google will need to know that URL when you get credentials in the next step.

2. For an app to use login with Google, the app itself has to "have an account" at Google. The credentials are like the userid and password of this account. To set up an account for your app at Google, follow these instructions. Only one partner needs to get credentials for the app.

3. You store the credentials on the forked demo server in a bit of private storage, which you can access through the "lock" icon on the left of your files in Replit. Put the Client ID and Client secret there, note the line of Javascript it gives you to access them from

your code, and find where you need to edit the login code (in index.js) to get them out. Hopefully once your code is using your credentials, the demo now works for you. You have to run the demo in it's own window(login does not work in the little browser in Replit, there is some kind of firewall). The demo uses Passport.js, a module that helps implement a variety of logins (not just login with Google).

4. Next, either get the tutor's solution to Fitness 2, or your own solution, into the project. If you feel at all confident with your work, you are probably better off using your own solution, since you understand it. All of your browser code from Fitness 2 should go into the "/user" directory in the login app, since it should be hidden until login. All of your server code has to be merged with the server code from the login app. Notice how the "isAuthenticated" middleware function tests to see if the user is logged in, and if so, allows them to access files in the "/user" directory, and answer AJAX queries of the form "/query". Also notice how it passes control to the next middleware function on the route by calling "next". You'll need to add "isAuthenticated" to all of the pipeline stages that handle your AJAX queries. Make sure you can log in and everything still works after you do.
5. This might be a good moment to add HTML and CSS to the splash page so that it resembles the spec drawing. Notice the mobile spec drawing, and make the splash page responsive.

6. For testing purposes, it will be helpful to allow the user to log out (many apps don't bother with this; if the user logs in, they stay logged in for hours or days). Add the logout button to the app, as shown in the drawings. Check the documentation for Passport.js to see how to teach your server to logout. Think about, should you get to the "/logout" route (pipeline stage) in the server by using "fetch" or by putting the button in an anchor tag? Check that you can log out and log in. (When you're logged out, you should not be able to get to the main page of your app by giving it's complete URL).
7. Now that we have login and logout, let's actually use the user profile information we get from Google. We'll need a new database

table to store profile information. Take a look at sqlwrap.js (we have not had to look at this yet). Add code to create a table "Profile" to contain profiles for users who have logged in. The profile can include just their Google userid and their first name. Find where the profile information comes into our app from Google, and add a database command to insert the user into the profile table, if they are not already there.

8. Now that we have a first name, we can get the app to greet the user by name! Add code in your server to answer an AJAX get query to the url "/name", and respond with the name of the user currently logged in. You can get this by finding the userid in the request object, and then looking up the first name in the database.

9. Next, let's store the userid along with the user's activity entries in the ActivityTable. You'll need to modify the activity database table creation in sqlwrap.js to include a userid column. Then, everywhere you enter a new row into the table, include the userid.

10. Now, let's use the userid when we take data out of the table. We'll need to match the current userid in every query, so that we only consider data from the current user when giving reminders or showing a progress table.

11. You're done! Please submit again in three places. First, on Replit, hit the "Submit" button. Replit will still let you edit your project, but PLEASE DO NOT. We will grade on Replit, and we want to see you project as it was when you submitted. Second, on Replit, download a timestamped version of the project, by downloading a .zip file (Files, three-dots icon, "Download as zip"). Upload this for "Assignment 3 - Fitness Log - stage 2" on Canvas. Third and last, fill out the form for this assignment on Gradescope. Be sure to add all your partners to the Gradescope submission.