Handoff Documentation

for

**Time-Tracker**

***Overview:*** *The primary use case for this application is to allow users to track their hours spent on projects. The application has three types of users: Admins, Instructors, and Users.*

- *Admins should be able to manipulate the type of each user along with all other necessary functionality.*

- *Instructors can create courses and projects, and accept/assign users to them. They can create and assign evals, and see the progress/statistics of each group/user.*

- *Users can request to join courses, view the details/projects of those courses, and create groups. A user can track their time spent on a particular project by clicking the Start/Stop buttons, or by entering the time in manually (some rules need to be enforced to avoid fabricated entries)*

*A secondary use case is to allow users to complete evals for their groups. Admins/Instructors should be able to create eval templates that they can assign out to groups. The evals will then contain questions by which users will assign Ratings for.*

# **Our Stack:**

This project is running an Angular 12 front end with a NodeJS backend paired with Express. The database is run using SQLite 3.

# **Prerequisites:**

These are the prerequisites for the project. It may run without some of them, but you’ll want to make sure you have each of these installed regardless. It will help you as you develop further.

VSCode:<https://code.visualstudio.com/download>

SQLite:<https://www.sqlite.org/download.html>

NodeJS:<https://nodejs.org/en/download/>

Docker:<https://www.docker.com/products/docker-desktop>

You must also ensure that WSL 2 is enabled. For more information on how to do that, look [here](https://www.omgubuntu.co.uk/how-to-install-wsl2-on-windows-10).

You can clone the repo here: <https://github.com/bradleypeterson/TimeTrackerV2>

# **Startup:**

Startup instructions are also detailed within the README.md file stored in the root of the project.

For first time startup, navigate to the directory containing the docker-compose.yml (this should be the root) and run the following command:

docker-compose up --build

For any subsequent startups, run

docker-compose up -d

The -d parameter will remove trailing logs, so omit it if you'd like logs to appear in the console. Only run with --build if you make changes to the package.json or config files on either project.

The front end should be visible by navigating to the url, “<http://localhost:4200/>”, and the backend will be running on “<http://localhost:8080/>”. You can verify the backend is running by checking that it displays “hello world” to the page when you navigate to the respective URL.

# **Current State:**

This project was initially developed using angular.js which is deprecated. Our objective was to fully migrate the project to Angular 12 using NodeJS as the backend. Here are the completed items in the new project as well as a limited/generic list of the outstanding items.

Completed Items:

- All pages have been added into the project (with limited styling and functionality)

- Bootstrap has been integrated into the project

- Login page is styled nicely

- Register page is styled nicely

- Dashboard pages are styled nicely

- Groups & Group pages are styled nicely

- Project page is styled nicely

- Courses & Course pages are styled nicely

- User page is styled nicely

- Login functionality exists (using hashing and salts)

- Registration is functional

- There are working links to navigate to each page (The nav bar changes depending on the type of user logged in. For example, the Instructor Dashboard only shows up for instructors)

- There are tables in the database for:

- Users

- TimeCard

- Courses

- Projects

- Groups

- AdminRequests

- CourseRequest

- GroupAssignment

- Courses Page

- For instructors, the add button (+) on the Courses page will allow an instructor to create a new Course which is then displayed in the list of courses, which are also links to each course’s page.

- For students, each course has a join, leave, or pending button (depending on whether they are currently in the course, requesting to take the course, or neither).

- For students, the bottom of the page will have a list of courses they have joined.

- Course Page

- For instructors, the add button (+) on the Course page will create a new Project and navigate to the new project’s page which updates the list of projects already in the course. These also act as links to each project’s page.

- There is also a list displaying all students currently taking the course.

- Project Page

- For instructors, the add button (+) on the Project page will create a new Group that is added to the list of groups, which are links to that group’s page.

- Each group will have a join or leave button (depending on if the user is currently part of that group) that is only visible to the students in that project.

- Group Page

- The Start Time and End Time buttons successfully start and stop a time entry (The time is displayed for each student in a pie and bar graph, each color representing a different student. The bar graph separates the data by date and amount of time clocked.)

- In addition to the buttons, the student can manually enter time with a description. The graphs and list of previous clock times are automatically updated.

- There is also a list of all the group members displayed

- Groups Page

- This page is visible only to students on the nav bar.

- Contains a list of groups that the current user has joined.

- User Page

- Accessed by clicking on the user’s username next to the logout button in the nav bar.

- Shows user’s username, first name, and last name, and allows them to change it.

- Allows user to change their password after entering their current password.

- Allows user to activate or deactivate their account by clicking the ‘Active’ checkbox.

- A user may also request to be an instructor.

- Dashboard

- The user is automatically redirected to this page once they log in.

- Contains cards for each class, group, and project the user has joined. The user’s projects are from what groups the user has joined.

- Each card is a link to each item’s page.

- Instructor Dashboard

- Visible to instructors only through the nav bar.

- Contains list of course requests from students, which can be accepted or denied.

- There is a list of courses taught by that instructor below that, with an inner list that contains the students in that course.

- There is a list of the instructor’s projects below that, with a list of groups inside. Each group is a link to the group’s page.

- Admin Dashboard

- Contains a list of all users which can be sorted depending on the user type or searched for in a search bar.

- Clicking on a user will bring up a modal that displays the user’s information and enables the admin to edit this information, reset their password, or delete the user entirely.

- There is also a button to the side which shows how many requests there are to elevate a user’s type to instructor or approve password reset requests.

- Clicking on the requests button will open an admin request modal that allows the admin to accept or deny user requests.

- User Account Modal (from the Admin Dashboard)

- Displays the selected user’s account information.

- The user’s account information can be edited by the admin after enabling editing and selecting edit

- The modal shows how many courses, projects, and groups the user is enrolled in as well as information about each course, project, and group.

- The user’s password can be reset to a default from this modal

- The user can be deleted from this modal.

- Request Modal (from the Admin Dashboard)

- Displays all user’s requests for an admin to approve or deny.

- The admin can individually select, approve, or deny on any request or approve or deny all requests.

- Requests are saved once the save button is clicked

- The list of requests can be filtered via radio buttons or by a username search

- Reset button resets all unsaved requests to pending

Outstanding Items:

- Limit which pages are visible depending on type of user logged in (hide links for pages a user doesn’t have access to)

- Project card link on dashboard is not redirecting to the correct page

- Courses page needs a search function

- Time Limits need to be set for manual clock entry

- Edit functionality for Courses, Projects, and Groups

- Right now there is no way to edit a course, project, or group after they are created

- Enable users to utilize search tool in course list

- Enable evals

- Evals were not fully implemented in the original project. There is some work in there (html), but it may need to be re-engineered depending on how you would like to implement them.

# **Tips and General Advice:**

- Since there is no built in debugger for this project, utilize console printing

- You can click F12 to open up the console and see console messages for the front end (Angular)

- You can use the Docker container console to see console messages for the backend (NodeJS)

- Feel free to look into implementing a debugger

- You can look into the angular chrome extension, it is a free install.

- **Don’t be afraid to jump in if you have no experience with Angular/NodeJS** (most of us didn’t). Here are a few links to help you learn:

- Straight from the source: <https://angular.io/>

- Pluralsight:<https://app.pluralsight.com/library/courses/angular-2-getting-started-update/table-of-contents>

- How it is setup with docker: <https://medium.com/bb-tutorials-and-thoughts/dockerizing-angular-app-with-nodejs-backend-85e9d332335d>

- Example Angular/NodeJS project: <https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-angular-app-with-nodejs-e24c40444421>

- Google is your friend

- All endpoints, database connecting/querying are written in the server.js file

- Call a function in server.js in the http.service.ts file

- All database table entries and seeding are in the seed.js file

**- How to access the database**

- The easiest way to have direct access to the database is to use VSCode. Install the extension SQLite by alex cvzz. Once installed, press ctrl+shift+p and search SQLite: Open database. Select ‘Choose database from file. An explorer window will open and navigate to the NodeApi directory in the project. Select the database directory and select main.db.

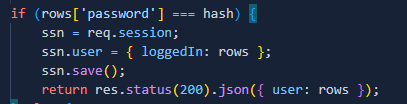
- This will open a new screen in VSCode that will allow you to write queries to the database. Right click on the query and select ‘Run Selected Query’ to run the query.

- A small window will open in the bottom left of the screen named “Sqlite explorer’ that will show more table information.

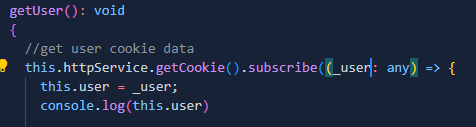
* **Backend Cookie management:**

-This site is set up to hold all user data on the node.js back-end server-side. This is accomplished using a cookie. When you login, you will see a cookie with the name “sid” in your developer console. This is where the user information is stored.

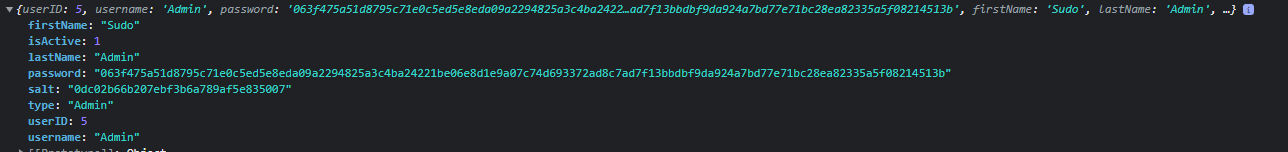
The user data is loaded into the server side cookie on the server.js /login route:



As each page loads, it calls /getCookie on the front-end to check the user server-side data.



Console.log user output:



Database Diagram:

