**TimeTrackerV3 Handoff (2024)**

**TimeTrackers**

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 manually view Users and Delete Users from the System*
* *Instructors can create courses and projects, and assign users to projects. They can create and assign evals, and see the progress/statistics of each group/user.*
* *Users can join courses, view the details/projects of those courses. A user can track their time spent on a particular project by clicking the Start/Stop buttons, or by entering the time manually (some rules need to be enforced to avoid fabricated entries).*

# Our Stack:

This project is running an Angular 18 front end with a NodeJS backend paired with Express. The database is run using SQLite 3. For server configuration, nginx is installed but needs to be set up with the certificates.

The application has an admin account. The username and password is admin.

# Our Project is Live:

The project is currently live on the schools network at the following IP address: <https://137.190.19.220:4200/>

In order to get full functionality you will need to touch

[https://137.190.19.220:8080/](https://137.190.19.220:4200/)  
And the server will allow you to navigate the project

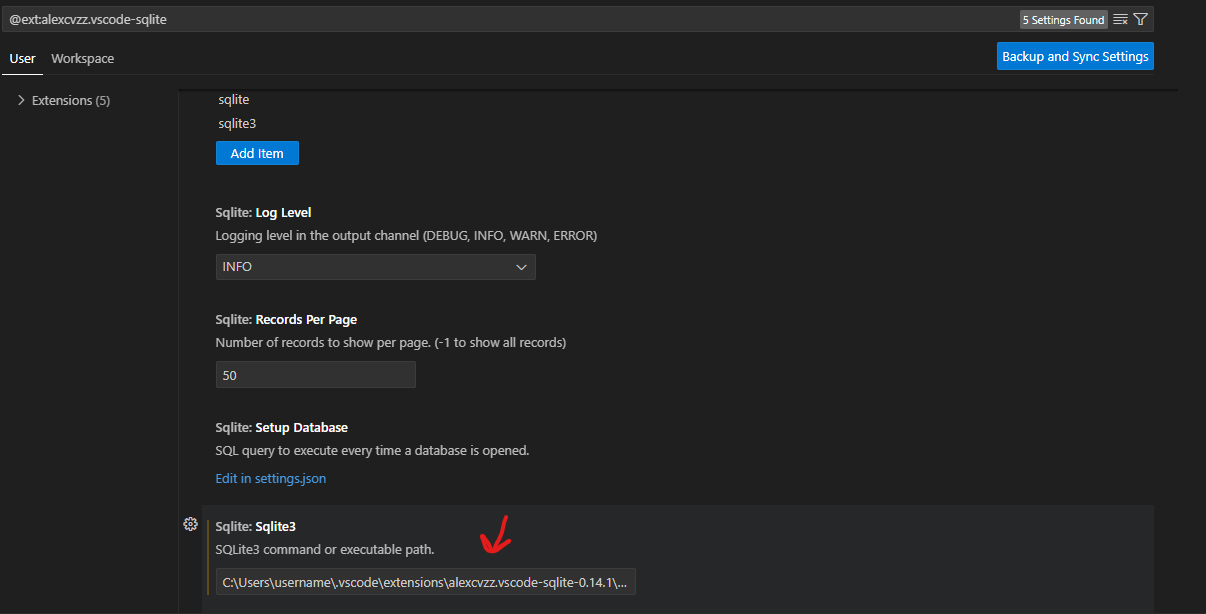
# Prerequisites:

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>

SQLite Extention in VSCode

* Make sure to update the settings inside the SQLite extension
  + C:\Users\username\.vscode\extensions\alexcvzz.vscode-sqlite-0.14.1\bin\sqlite-v3.26.0-win32-x86.exe

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

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

Putty: <https://www.putty.org/>

Testing Locally:

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. I

If you add the certificates locally to the keystore you wont have to navigate to [http://localhost:8080](http://localhost:8080/) every time you start

In order to test the application locally on your machine, you will need to update the following files. I would recommend working and doing as much as you can locally before testing onsite. At this point your local environment (should it be working) will match the setup found on the server.

* TimeTrackerV2/Angular/src/environments/environments.ts
  + Update the ip address to instead say https://localhost:8080
* TimeTrackerV2/NodeAPI/server.js
  + Change the origin to origin: "<https://localhost:4200>"

Before deploying your latest build onto the server please make sure you update these files to point back to the server IP, otherwise it wont start!

Testing on Server:

Currently the project lives under the /home/timetracker\_u/TimeTrackerV2 in order to start and stop the docker image, you will need to be in the project directory and perform the following commands. docker-compose down and then docker-compose up --build doing these two will bring the project up

The server deployment had us facing many different challenges. The largest being networking. Normal docker setups will default to an ip address range that is not typical of others. Once docker was installed on the server, we were no longer able to access the server via the VPN as the VPN uses the same default ip settings and created a conflict. In order to get around this issue, we ended up having to create a /etc/docker/daemon.json file that pointed us to a new range. Because of this custom setup, anytime you wish to restart the docker image you will need to stop it and start it. I will include a command cheat sheet.

If for any reason you can't access the server though the Schools VPN but can when present at school on the network. It will likely be due to the server

Should this ever happen you will need to update the following files:

* /home/timetracker\_u/TimeTrackerV2/Angular/src/environments/environments.ts
* /home/timetracker\_u/TimeTrackerV2/NodeAPI/server.js

Where the previous IP address is, you will simply replace it with the new one.

Command Cheat Sheet:

* sudo su -
  + This command makes you root. When running our project we would run the docker containers as root.
* docker-compose up --build
  + This command brings the docker containers up and performs all the necessary build commands in order to run the project
* docker-compose down
  + Because of the networking setup that we were required to figure out. Every time you wish to restart the docker image you will need to bring it down.
* docker ps -a
  + This command shows you all the running containers. If you are unsure if you need to perform the docker-compose down command you can run this to see if any are present.
* systemctl stop nginx
  + Sometimes when you go to build you may see that nginx is already bound to Host is already in use 0.0.0.0/127 or something to that tune. This is because nginx is still running.
* systemctl start nginx
  + This simply starts the service, should you need it running alone this is what you would do.
* docker volume prune
  + If you end up trying to start up the docker image and it tell you that you are out of space, you may need to run this in order to reclaim space. Be careful with this command. If you are starting and stopping the image a lot this can fill the server up. Here is what the error looks like
    - ERROR: for timetrackerv2\_angular\_1 Cannot create container for service angular: failed to copy files: write /var/lib/docker/volumes/7c7a48923e59761bb4ddcdea4441eed3d1b97210df17c7a18dc5e13a4695cb3e/\_data/@babel/template/README.md: no space left on device

# Our Semester’s Journey

## Completed Tasks

## Deployed and Running on Server:

* During this semester we were given a server from the school. We have taken the project and have it running. There are still items that need to be completed in order for this to be a completely finished project.
* Added ngnix to the project.
  + Nginx is pointing to the certificates so you no longer are warned by your browser that the Cert Is Not Authorized. However, you will need to look into obtaining a self signed certificate to become a trusted site.
* Installed Docker service on server
  + Worked with IT to allow for custom Docker setup to allow for server access with VPN
* Updated api fetch logic to no longer contain a hardcoded value for localhost and instead point to the servers backend at the appropriate address. Please review the previous section for more information.
* We renewed the certificates. During this semester the certificates expired and we renewed them with the provided command in both certificate folders. At some point you will want to obtain a CA Authorized certificate.

## Frontend:

* Updated Angular to version 18 (it looks like a new version is on the horizon)
* Admin
  + Finished the admin manage users page to allow the admin to approve and disapprove users to become instructors.
  + Added “Filter by” label to admin manage users page
  + We updated the Admin view for the users page to conditionally render if no users have been modified
  + Integrated a checkbox in the admin interface and implemented alerts for pending approvals.
* Dashboard buttons
  + We updated the drop banners with a contemporary style, replacing an outdated drop shadow.
  + We added the ability for the user to logout from the dashboard instead of needing to go to their profile
* Evaluation Page
  + Updated the save button to rest at the bottom of the form
  + Improved logic for assigning evaluations to students. Prior it was one single array and this was broken down and completed the logic for displaying each form to each student.
  + Added user feedback to the manage-evals component to tell the user they successfully saved the form.
* Removed the changepassword component and implemented the reset password component with better logic.
* Commented out Unit Tests
  + Every time you create a component in angular it comes with a boiler plate unit test. Though every component has one, it is not working with the project and likely never did. It was then decided to comment out each one.
* Added underlined tabs to highlight the active page for improved navigation.
* Enhanced the "View Profile" page layout by increasing spacing between elements and the banner.
* Updated the "Name Dropdown" menu with profile and logout links, icons, and a more noticeable border/shade.
* Adjusted UI elements to ensure all buttons and dropdowns appear clickable, with added hover effects like underlines and background color changes.

## Backend:

* We have confirmed that the backend is secure in its interactions with the frontend.
* Added an isApproved value to the user table. This is used as a flag to help the admin determine who is an instructor and who is not.
* Addressed backend communication issues by dynamically setting hostnames.

Outstanding Items:

Server Side

* Add certificates to the keystore and set up SSL integration with ngnix.
  + There are a few ways to tackle this. The main reason we introduced nginx is because similar deployments use nginx to help handle this portion.
* Add an audit log for database changes. Knowing who is updating and when would be helpful in maintaining security.
* Get a proper DNS, instead of the users navigating to an ip address a proper domain would be good. You will likely need to look into getting proper certificates instead of using self signed ones.
  + Once this is done you will need to update the environment.ts to point to the new DNS
* Address nginx errors
  + When doing the docker-compose up -–build docker is giving the following error
    - | 2024/12/05 04:27:52 [emerg] 1#1: cannot load certificate "/etc/ssl/ certs/cert.crt": BIO\_new\_file() failed (SSL: error:80000002:system library::No s uch file or directory:calling fopen(/etc/ssl/certs/cert.crt, r) error:10000080:B IO routines::no such file)

nginx\_1 | nginx: [emerg] cannot load certificate "/etc/ssl/certs/cert.crt": B IO\_new\_file() failed (SSL: error:80000002:system library::No such file or direct ory:calling fopen(/etc/ssl/certs/cert.crt, r) error:10000080:BIO routines::no su ch file)

* + - The certificates do exist at these locations. One will need to investigate why this error is being thrown and also determine if the nginx.conf file is correctly set up. Most of the settings in the nginx.conf are boilerplate.

Frontend

* Functionality for instructors to be able to add students to a course when or after creating a Project.
  + Students should show up in table on course page
* Currently there is a search bar in the add courses page that should allow for the student to search for specific courses that are not yet functional.
* Finish the eval component
  + We have the eval form assigned to others, and the form allows for data entry. However, when you press submit. It currently does not store in the database.
  + It calls [api/submitResponses](https://localhost:8080/api/submitResponses) but it doesn't actually submit the data.

UI Touch Ups

* + The admin manager user page, when updating, adds a save changes column to the far right and messes with the layout and order of the data displayed.
    - It needs to be added to the bottom maybe or displayed better so that the user does not lose focus of the current work
  + Button placement for a lot of the components need help. Most are working but need work for setup. A lot of them are mixed in with each other and the user experience feels clunky.
  + Report component, the layout is a bit funky and needs to be reworked.
* Unit Testing
  + Currently they are all commented out as they were not working and will require additional coding in order to work. Once you have them

Backend

* Storing Data: The local storage currently stores everything about a user, such as their hashed password, salt, username, ect. This is a massive security issue. Need to see about removing the user, or at least the things important for authentication, from localstorage. This issue can be viewed when the console writes out the information while on a project’s page
* Search functionality on course page should be implemented for projects and students. Currently this is not finished. See Goal Priority list for details. The main idea is the user should be able to filter projects and students to provide organization.

**GOAL PRIORITY LIST (FALL 2024)**

PRIORITY 1:

* **DONE** **-** Get the server running live though some platform
  + Server is running through the main Weber Server. (provided by school)
* **DONE** **-** Possibly figure out a new Database
* **DONE -** Add student name in top left corner, make it a drop-down for profile and logout
* **DONE -**  If continuing with SQLlite, protect against SQL Injection
* **DONE -** The back-end server is not secured, interaction with the db through API calls, there should be some method of authentication protecting the db data.
* Polishing major bugs
  + **TBD** - Have sorted several, keep an eye out for any we missed or that may appear with new code.
* **DONE -** Follow through with Unit Testing in frontend
  + Add unit tests if it's determined they should be included. Right now all unit tests are just commented out for each component, as they do not have proper functionality.
* **DONE -** Updating Angular to the latest version (18)

PRIORITY 2:

* Clean up any unused routes and their relevant components.
  + **DONE** - We’ve weeded out several of these but some may still exist.
* IMPORTANT: There exists empty components in the repo for the future implementation of EVALS
  + **DONE-ISH** (see below for additional changes needed)
  + Many of the components are partially built but with no functionality (typescript) such as the 6 different eval component (admin-evals, assign-evals, eval, manageeval, view-eval, and view-evals)
    - Students can fill out their assigned evaluations, but the numbers for each question are incorrect and they cannot actually submit the form.
    - Assign an eval to a course only to the active groups at the time
    - Allow groups to be created within each project, and ensure that any evals are only assigned to active groups
* Add admin courses and evaluations pages.
  + **DONE-ISH** (see below for additional changes needed)
  + Currently admin is just routed to the instructor courses components, so there needs to be a switch case added for all course and eval related routes to enable admin specific functions
  + The following are not done:
    - Admin create/delete/ edit course
    - Admin add/remove students in course
    - Admin create/delete/edit project
    - Admin add/remove students in project
    - Admin edit/delete timecard
    - Admin create/delete/edit/assign evaluations
    - **Relevant components to check**
      * app
      * dashboard
      * create-course
      * course
      * courses
      * course-reports
      * edit-course
      * create-project
      * project
      * edit-project
      * add-student-project
      * admin-evals
      * assign-evals
      * manage-evals
      * edit-timecard
      * view-eval
      * view-evals
      * view-report

PRIORITY 2 Cont:

* **TODO -** The local storage currently stores everything about a user, such as their hashed password, salt, username, ect. This is a massive security issue. Need to see about removing the user, or at least the things important for authentication, from localstorage. This issue can be viewed when the console writes out the information while on a project’s page.
* **DONE -**  The changepassword component seems to be not in use, because this component is replaced by the resetpassword component. Need to see about removing the changepassword component from the project to clean it up. The same goes for the file located here “.\TimeTrackerFall2023\TimeTrackerV2\Angular\src\app\project\project.component.original.html”. This file is never used inside any of the files inside the project, so we are assuming it was a backup of the file “project.component.html” before it was modified.
* **Likely won't happen -** Figure out Weber CS Active Directory Integration.
  + Brad came up with this idea. It's something that could still be done but would require a major overhaul of the systems database. The current work is working just fine.
* **DONE -**  Clean the user's page for admins so that the "User Controls" column is either removed or conditionally rendered if no users have been modified.
* **DONE -**  Instructors should get admin approval upon registration before having access to full functionality
* **DONE -** Add logic to only display the “Evaluation Due” button (Only make this visible if the student is assigned an evaluation due, make some sort of pop up or pull the attention) (dashboard component) to only be displayed when the current user is in the Assigned\_Eval table.

PRIORITY 3:

* **DONE -** Instructor dashboard shows pending students for courses, has an empty description section
* **DONE -** Display Work Description for projects in the history table
* **DONE -** Projects in inactive courses should be treated as inactive; currently, they still behave active projects
* **TODO -** Add an audit log that is a part of a separate database, so any changes made in the current database are kept record of. For example, a deleted/edited course or project, the log will show who made this change, the time and location.

# 

# Database Schema

* An editable version of this chart can be found inside the repository in the “docs” folder with the name of “Time Tracker database schema.drawio”. This file you can then upload to google drive and have multiple people working on it at the same time. We used google drive, but there are more options available.
* Once uploaded, you can open it inside the web page “[app.diagrams.net](http://app.diagrams.net)”. An important note:
  + You have to click on the “More Shapes” button and under the “Software” section, you have to click the checkbox for “Entity Relation” and click the “Apply” button. This will add a new dropdown on the left side of the screen named “Entity Relation” that contains the shapes to model the DB (tables, entries for the tables, and relationships between each of the tables).

Database Table explanation:

* There are tables in the database for:
  + User
    - Used to store user information.
    - The type column in the table is used to distinguish between student, instructor and admin accounts.
    - Contains an isApproved value that requires the admin to determine if they are allowed to be an instructor or not.
    - Contains an isActive value that admin can set to true/false.
  + TimeCard
    - This is used by student users to keep track of time that is worked on a project.
    - Each entry serves as a Time Log, that has a students start time, end time and a description to enter what was worked on during that time entry.
  + Course
    - This table is used to store course information. An entry for this table is created when an instructor adds a new course, through the dashboard.
  + Course\_User
    - This table serves to connect student users to courses. An entry for this table is created when a student registered for a course through the add course tab on a students navigation view. This table references the students UserID and the courses CourseID
  + Project
    - This table is used to create projects for a course. An entry for this table is created when an instructor creates a project for a course.
  + Project\_User
    - This table is used to connect a number of student users to a project. An entry for this table should be created by an instructor adding students to a project. When a student is assigned to a project by an instructor, a clickable card should appear on a students dashboard page that will navigate to the appropriate project page.
      * (NOTE - Functionality of having instructors assign students to a Project is not yet completed. Entries for this should be hard coded into the database to view the clickable links on a students dashboard until implementation is complete)

# Tips and General Advice: 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>
* I wish I knew how difficult updating the project to the latest version of Angular would be. We all had our own forked project with separate branches and getting them merged was quite the ordeal. This was not something that was easy to do and I would give your team proper time to get things merged and working.
* If you dont know linux very well, I would check out a few videos on YouTube on how to navigate the server. It would be worth your time to get good with moving around the terminal and understanding the setups.
* When troubleshooting the server, I would use a tool called screen it is a very nice tool. Here is a link to some documentation: <https://www.gnu.org/software/screen/manual/screen.html>
* The project overall can feel a bit daunting, especially when diving into the type script but after you play with a component or two itll make a lot more sense in how things are organized.
* VSCode will throw a lot of false positives, unfortunately we did not find a way to suppress them and resorted to ignoring and compiling to see where out mistakes were.
* Backend server side took a lot of time to learn, I recommend getting comfortable with this as soon as possible to help your latest deployment go smooth.