**Time Tracker 23 Hand-Off Document**

# Application stack, programs used for development, and configuring Node/Angular servers to run

## Application Stack: SEAN (SQL, Express, Angular, Node)

This project is running on Angular 16 on the front end with a NodeJS backend paired with Express. The database is run using SQLite3.

## Applications Used for development:

* VSCode:<https://code.visualstudio.com/download>
  + Extensions: SQLite by alexcvzz (This allows you to run and test queries through VSCode).
  + If for some reason when you attempt to open the DB after installing this extension and you receive an error saying something along the lines of “Failed to open database '<path to main.db>': Parse error near line 4: no such column: table aster WHERE (type="table" OR type="view")”. Then you need to go into the extension settings for SLQite and change the setting “sqlite3” from being “sqlite3“ to instead being a executilbe located inside the files installed for the extentsion. The path to said file should be something similar to this “C:\Users\<Current user logged in as>\.vscode\extensions\alexcvzz.vscode-sqlite-0.14.1\bin\sqlite-v3.26.0-win32-x86.exe”. Once you have changed this setting, then you should be able to open the DB in VS Code.
* SQLite3:<https://www.sqlite.org/download.html>
* NodeJS:<https://nodejs.org/en/download/>
* Docker desktop (you don’t need this if you chose to use the option inside the “[Another way to install packages and start servers (running on local hardware, not a VM)](#_mv7bd6sdgvt9)” section):<https://www.docker.com/products/docker-desktop>
* RESTer chrome extension (to perform HTTP requests with any method, URL, body and headers): <https://chromewebstore.google.com/detail/rester/eejfoncpjfgmeleakejdcanedmefagga?pli=1>
* Command line installs:
  + npm install -g @angular/cli

## Configuring Node/Angular servers to run using Docker

### Steps to install the required packages/Docker images work:

1. Clone the repository

2. Install at least the [SQLite](https://www.tutorialspoint.com/sqlite/sqlite_installation.htm), [NodeJS](https://nodejs.org/en/download), and [Docker](https://www.docker.com/products/docker-desktop/) programs and [enable WSL2](https://www.omgubuntu.co.uk/how-to-install-wsl2-on-windows-10) on your system, and restart your computer after they are installed. Important note, for the SQLite program there are two things misleading in the instructions.

1. The name of downloads has changed, but not the bit version. So install both versions that have “win32” in their name
2. It says to add “C:\sqlite” to your PATH environment variables, but this is not valid because the tools are contained in a folder when you unzip the files to the folder “C:\sqlite”. For example, my PATH variable is “C:\sqlite\sqlite-tools-win32-x86-3430000”.

3. Make sure that Docker Desktop is running on your system, wait until it doesn’t say “Starting the Docker Engine…” when you first start it.

4. Once it is started, using your desired terminal, we used Powershell, and navigate to the directory that has the file docker-compose.yml file, “Root\_of\_the\_repository\TimeTrackerV2\” and run the command “docker-compose up --build”.

5. Now you just wait until it is finished installing the packages and Docker containers used to run the NodeJS and Angular servers. You will know it is done when it says “Compiled successfully.” in the terminal for the “Generating browser application bundles”.

6. Now the servers should be running on your system. You can check this by going to “[https://localhost:4200](http://localhost:4200/)” for the front end and “[https://localhost:8080](http://localhost:8080/)” for the back end.

7. To remove the Docker container, simply run this command in the same place as the docker-compose.yml file, “docker-compose down”

### Steps to start/stop the Dockers for the NodeJS and Angular servers (2 ways of doing this):

1. Using Docker Desktop

1. Go to the Containers tab and you should have a container named “timetrackerv2”. Then in the actions section, you will have a Play/Stop symbol to either start or stop the NodeJS and Angular servers. It will take a few seconds, 10-20 if I were to guess, to process it, so wait for it to start/end.

2. Using the Terminal

a. Make sure that Docker Desktop is running on your system

b. Navigate to the directory that has the file docker-compose.yml file. Root\_of\_the\_repository\TimeTrackerV2\

i. To start the NodeJS and Angular servers, you would run the command “docker-compose up -d” or “docker-compose up”.

ii. To stop the NodeJS and Angular servers, it depends on how you started it. If you added the “–d” to start it, you would run the command “docker-compose stop”. But if you didn’t, you can simply press the key combination “Ctrl + C” to stop the servers.

Once you have done one of the above, you can navigate to “[https://localhost:4200/](http://localhost:4200/)” for the front end and “[https://localhost:8080/](http://localhost:8080/)” for the back end if the servers are running.

### Another way to install packages and start servers (running on local hardware, not a VM):

Using Windows PowerShell:

1. Firstly, you need to change your execution policy from your defaults on the CurrentUser or LocalMachine level to be preferably “RemoteSigned” so you can execute the powershell scripts.
   1. First, it is recommended that you figure out what your current execution policies are by using the powershell module [Get-ExecutionPolicy](https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-executionpolicy?view=powershell-7.4) and typing in the following command “Get-ExecutionPolicy -List” so you know what your current settings are so you can revert the changes on a future date.
   2. Next, you can set the execution policy by using powershell module [Set-ExecutionPolicy](https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-executionpolicy?view=powershell-7.4) and using either of the following commands “Set-ExecutionPolicy RemoteSigned -Scope LocalMachine” or “Set-ExecutionPolicy RemoteSigned -Scope CurrentUser”.
2. Navigate to repository directory “Root\_of\_the\_repository\TimeTracker23\TimeTrackerV2”
3. To install packages run the following command “.\Install\_Packages.ps1”
4. To start the server run the following command “.\Start\_Servers.ps1”

# Configure PC for browsers to accept https connections for this application

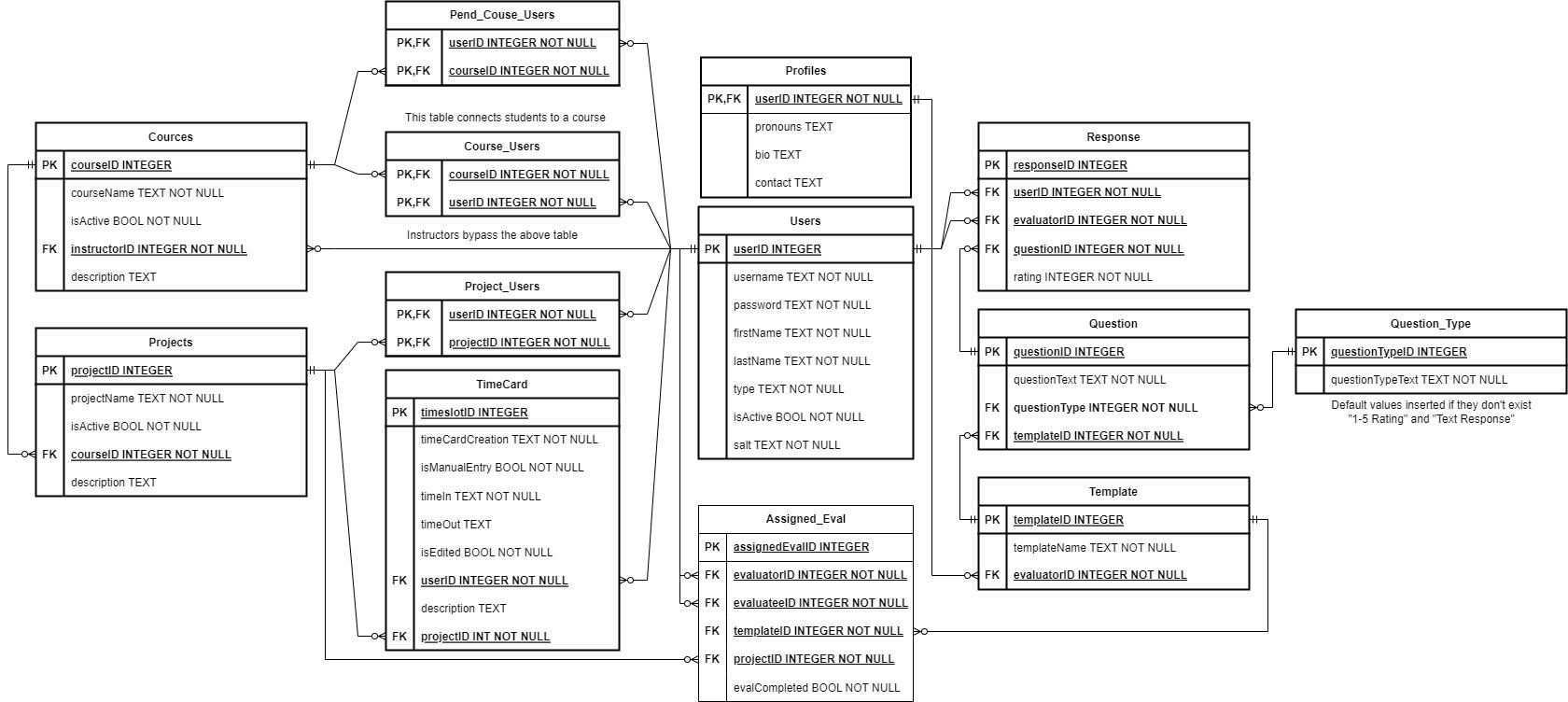
Before you start configuring the PC, an important note is that the certificates used in both node and angular servers are valid from 11/3/2023 to 11/2/2024. So if the current date is not in this range, you have to re-generate the certificates. To do so, you need to install OpenSSL on a PC ([how to install OpenSSL on windows](https://thesecmaster.com/procedure-to-install-openssl-on-the-windows-platform/)) and run the command inside the txt file in the folder as referenced in option 2 below to generate a new set of the certificates and private keys.

Now, because the application uses self signed certificates, you have to configure your PC to say these certificates are secure. To do so, there are two ways of doing this:

1. Go to each URL <https://localhost>:(8080 and 4200) and allow the connection by clicking the “Advanced” button and then the “Proceed to localhost (unsafe)”. This will be persistent in the normal browser, but it will not be in a guest browser.
2. This next option has two ways of doing this, but both require you to know where the certificates are located for both the Node and Angular servers. Each of these certificates files have the extension crt and they are are located at “Root\_of\_the\_repository\TimeTrackerV2\NodeAPI\certificates\cert.crt” and “Root\_of\_the\_repository\TimeTrackerV2\Angular\certificates\cert.crt” respectively. Each of the below options are assuming you are using Windows as your OS.
   1. For the first option, follow the below steps (has to be done once for each certificate):
      1. Locate both of certificate files referenced above and double click on them.
      2. In the resulting window, click on the “Install Certificate…” button.
      3. Now select the radio button “Current User” if is not already selected and press the “Next” button.
      4. In the next window, change the default radio button selected to be “Place All certificates in the following store” and select the “Trusted Root Certification Authorities” store folder and then click the “Next” button.
      5. Lasty, click the “Finish” button and the certificate will be added to the certificate manager inside your OS.
   2. For the second option, follow the below steps (has to be done once for each certificate):
      1. In the search bar, type “run” and in the run window type “certmgr.msc”.
      2. Next, navigate to “Trusted Root Certification Authorities” > “Certificates” and right click on the folder and select “All Tasks" > “Import…”
      3. Now, make sure that “Current User” is selected and press the “Next” button.
      4. Next, browse to one of the “cert.crt” as talked about above and select it and then press the “Next” button.
      5. Now simply follow the steps as defined in the first option starting at the bullet point “iv”.
      6. After you pressed the “Finish” button, you will get a “Security Warning” error saying something like “You are about to install a certificate from a certification authority (CA) claiming to represent: localhost 4200 <or> 8080”. Simply press the “Yes” button and it will be added as a trusted root certification authority.
   3. You can view these certificates that have been imported by using the same method as referenced in option ‘b’ in step i and the first half of step ii to navigate to the store where you saved them. Then in the column “Issued To” or “Issued By” look for “localhost 4200” and “localhost 8080”. These are the certificates that has been added so you can either remove or view them if you want. Such as being done developing this project and you are cleaning up your PC or replacing these self signed certificates with certificates that are signed by a Trusted Certificate Authority.

Both of the above methods work, but the second option makes it work for any instance of a browser, I.E. your primary or guest browser and it also treats it as a trusted certificate so the icon on the browser becomes a closed lock.

# 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 as shown in following link. Once you have uploaded it, you can open it inside the web page “[app.diagrams.net](https://app.diagrams.net/)”. But just as 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 contain the shapes to model the DB. Such as the tables, entries for the tables, and relationships between each of the tables.

# App Components

Under the “TimeTracker23\TimeTrackerV2\Angular\src\app” you will find the following components. The only files we edited in each component were the *app\_component\_name*.component.css, *app\_component\_name*.component.html, and *app\_component\_name*.component.ts.

* add-courses
  + This component is not used. Students can add/register for courses in the courses component.
* add-student-project
  + Under the project page for an instructor, there is an “Add Students” button that navigates to this page. In this page, it pulls all the students in the course, sorting by whether they are already in the project or not in the project.
* admin-evals
  + This component is not currently being used.
* assign-evals
  + This page is used by instructors to assign evals to projects. It can be navigated to under the course page for instructors, “Assign Evals” button.
* changepassword
  + This is implemented in the resetpassword component. This is not currently used and is unfinished.
* course
  + This page shows all projects and students in the course.
* course-reports
  + This page is referenced at the end of the course page. This shows specific report data about the student, what projects they are in and the total time the student contributed to each project.
* courses
  + This page is viewed by students allowing them to register, drop, and search for courses. Students navigate to this page through the navbar.
* create-course
  + This page allows an instructor to create a new course. Instructors can navigate to this page through the navbar or the dashboard “Create New Course” button.
* create-project
  + Under the course page for the instructor, there is a “Create New Project” button that redirects you to this page. This allows the instructor to add a new project to the course.
* dashboard
  + This is the homepage after a user signs in. There are switch cases for each different user in how it displays info: admin, instructor, and student.
* edit-course
  + This page is found by the “Edit” button on the course page for the instructor. It allows the instructor to make changes to the course.
* edit-profile
  + This page allows users to edit the bios, contact, and pronouns in their profile.
* edit-project
  + This page is found through the “Edit” button on the specific project viewed at the moment by the instructor. It allows the instructor to edit project details and set the project inactive or active.
* edit-timecard
  + This page is found through the “Edit” button by the timecards of the students in the history section of the project page. It allows the instructor to change the times entered by the student.
* eval
  + This page is viewed by the student and is found when there is an “Evaluation Due” button on their dashboard.
* group
  + This component is not used. Groups have been implemented under the project component.
* inactive-courses
  + This shows all the past courses that have been set to inactive by the instructor.
* login
  + This is the login page for all users. It is the landing page.
* manage-evals
  + This page is used by the instructor and is navigated to by the “Manage Evaluations” button on their dashboard. This page allows instructors to edit and create new evaluations.
* navigation
  + Used for debugging purposes to get the routes to all pages and components.
* pipes
  + Takes a number (time in milliseconds) and parses it into hours, minutes and seconds.
* project
  + This page lists the specific time log information for the group. Users can click on their projects or the “View” button under the course page by the specific project they want to see the time logs. If they are a part of the project, they will be able to add time entries.
* register
  + This is the register page for new users making an account, either a student or instructor account.
* resetpassword
  + This page can be navigated to by the profile page. Each user can reset their own password, unless they are an admin which can navigate to any user profile and reset the password for the user.
* user
  + This component is not currently being used.
* user-profile
  + This page displays the user’s name, pronouns, contact, and bio. Can be found through “Profile” in the navbar.
* users
  + Admins can view and manage all users (set users active or inactive, and change their user type).
* view-eval
  + This component is not currently being used.
* view-evals
  + This component is not currently being used.
* view-report
  + This gets all the timecard information for each student in the project.

# 

# Our requirements listed with priorities

Features ranked from 1 to 3

1. Highest Priority
   1. Instructors
      1. Create courses and projects (DONE)
      2. See time logs from each user (DONE)
         1. See time report from each group (DONE)
      3. See a chart of time spent by user (DONE)
      4. View time logs of all users in the course (DONE)
   2. Users
      1. See time logs from each user (DONE)
         1. See time report from each group (DONE)
      2. See a chart of time spent by user (DONE)
      3. Join a group (DONE, the project is the group for how we envision it)
      4. Track time spent on a project using Start/Stop buttons (DONE)
   3. Other Features
      1. Update to Angular 16 (DONE)
      2. Deployment (however this can be done)
2. Desired Features
   1. Admins
      1. View and Delete any user (DONE)
      2. All functionality of other users (To be broken into multiple steps)
   2. Instructors
      1. Assign users to projects (DONE)
      2. Approve users to join course (DONE)
      3. Add users to a group (DONE)
   3. Users
      1. Request to join courses (DONE)
      2. Drop a course (DONE)
      3. View project details in course (DONE)
      4. Leave a group (DONE)
      5. Manually enter times up to 5 times a day before reaching a limit (DONE)
   4. All Users
      1. Login (use hashing on the client instead of the server (DONE) and a certificate (DONE, but with a self-signed certificate, not one by a trusted Certificate Authority. It should be easy to convert, simply replace the certificates inside the servers with the ones from a trusted Certificate Authority)) [How to implement a certificate](https://www.youtube.com/watch?v=USrMdBF0zcg&ab_channel=MafiaCodes) (This process can be applied to the angular server as well with some [slight changes](https://betterprogramming.pub/how-to-serve-your-angular-application-over-https-using-ng-serve-240e2c2e0a5d))
      2. Register for an account (users and instructors) (DONE)
3. Additional Features
   1. Instructors
      1. Edit and delete courses they have created. (DONE)
      2. Edit and delete projects they have created. (DONE)
      3. Create and assign evals
         1. Sent to active groups only
      4. Search for courses, projects, and groups (DONE)
   2. Users
      1. Search for a course or project (DONE)
      2. Fill out assigned eval
   3. All Users
      1. Edit profile (DONE)

## Completed Items:

### Courses

* Delete course functionality
  + Delete course updated, confirmation dialog added
* Edit course functionality
* Search functionality for courses on Add Courses page for Students
* Search functionality on course page should be implemented for projects and students
* Students should show up in table on course page
* Instructor admin to course
  + *\*\*\* Students must request to join a course*
* Get rid of View Course buttons for those not in the course (Register Courses and Pending Courses sections should not have a view button)
* Improved UI under student section under View Course page, decrease the white-space
  + Changed student list to cards in a grid
  + Changed projects under student card into a table
* Instructors can set a course to inactive
  + Once inactive, courses do not appear for registration
  + Inactive courses do not appear on the instructor dashboard
  + Inactive courses appear on a Past Courses page for each instructor
  + New projects cannot be created in an inactive course

### Projects

* Delete project functionality
  + Delete project updated, confirmation dialog added
* Edit project functionality
* Student can join and leave groups
* Instructor can add or drop a student to a project
  + Back to Project button added to add students page
* Time tracking buttons only appear for users in a project
* Back to Course button added to project page
* Projects can be set to inactive on the edit page
  + Inactive projects do not appear on the dashboard
  + Timecards cannot be created for inactive projects
  + Inactive projects cannot be joined or left by students
* Search for projects implemented
* Students shouldn’t be able to join projects in courses not registered for.
* Students can now view projects they are not in
* Improved UI of projects page
  + Add a separator between My Activity and Team Times
  + Centered manual mode button
  + Fixed header styling
  + My Activity no longer appears for users not in the project.
  + Restrict My Activity to the 5 most recent timecards
  + Move pie chart key to the side of pie chart
  + Added max width to pie chart
* Instructor can update or delete times on project
* Instructors can manually enter times for students.

### Other

* *\*\*\* Upgrade to latest Angular 16*
* Can see a chart of time spent by student (summary)
* Reports page for individual user needs to be implemented showing time logs, and comments for each time log entry unique to the User (Accessible to project members and to instructor)
  + *\*\*\* time logs available to the whole class*
* User profile can be viewed and edited
* Remove all inline SQL statements with the ‘?’ insert statement.
* Make it so that if there are no admins in the DB, make Nodejs create a default admin and notify the first user that goes to the login page that an admin account has been created.
* Change button colors for register
* Fix hashing so it happens client-side instead of server-side
* Changed btn-primary to btn-purple to fix the colors
* Fixed spacing at top of page
* Move or implement the same logic for resetting a password and delete user/account buttons inside the admin’s Users page to the profile for the specific user. This will allow the user to reset their password and delete their account if they want to.
* User profile UI improved
* Users can view other users’ profiles
* Added pagination to:
  + View Reports
  + Project Timecard History
  + Past Courses
  + Pending Students
  + Add Students to Project
* Add a “Back to Course” button to the “Assign Evals” page.
* Add a “Back to Course” button to the “Create New Project” page.
* Admin dashboard created with recent users, courses, and projects

## Unfinished Items

* Enable evals
  + Allow instructors to create and assign evals to course users. Allow course users to fill out individual evals for themselves and for their project member
    - *\*\*\* assign an eval to a course only to the active groups at the time*
  + NOTE: There exists empty components in the repo for the future implementation of EVALS.
  + Student can complete eval assigned
  + Many of the components are partially built but with no functionality (type*script)* such as the 6 different eval component (admin-evals, assign-evals, eval, manage-eval, view-eval, and view-evals)
* Add student name in top right corner, make it a drop-down for profile and logout
* 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.
* 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.
* 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.
* Add admin courses page
  + Admin edit course functionality
  + Admin delete course functionality
  + Admin add/remove students in course functionality
* Add admin projects page
  + Admin edit project functionality
  + Admin delete project functionality
  + Admin edit/delete timecard functionality
  + Admin add/remove students in project functionality
* Instructors should get admin approval upon registration before having access to full functionality
* Instructor dashboard shows pending students for courses, has an empty description section
* Projects in inactive courses should be treated as inactive; currently, they still behave as active projects
* 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.
* Add logic to only display the “Evaluation Due” button   
  (dashboard component) to only be displayed when the current user is in the Assigned\_Eval table.
* 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.