Pre-requisites And Project Set Up

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 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
- To upload to the server use WSL and the following commands, -r is for recursive meaning folders/directories
    - pscp [-r] /mnt/[absolute path on windows] timetracker_u@137.190.19.215:/home/timetracker_u/TimeTrackerV2

Database

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”. 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)
- NOTE: The password for dummy data users is ‘ChangeMe123’

Tips and General Advice

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 sm

Spring 2025 Hand-Off

Our Priority List:

1 - Highest Priority:

- Fix nginx errors so the website can be hosted.
- Get the site up and running on the VPN
- Create an automated process that tests the site by creating two admins, several teachers, many classes, and dozens of students.
- Verify that all screens are working as expected with dozens of students worth of data.
- Get all requirements for evaluation pages, including a UI storyboard.
- Add the ability for an admin to create a course. Verify students can join the course. Admins are teachers by default.
- 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.

2 - Desired Features:

- Get all unit testing working.
- Find a mechanism for unit tests to run often, such as per commit, or a separate website or executable that can be manually invoked.
- Finish the eval component with all of its features.
    - Further testing
- ~~Look into either encrypting the database, or having Weber State credentials to access.~~
- ~~Verify that no user data is stored in browser local storage on the user's computer.~~
- Fix the background when clicking the hamburger button on mobile/tablet mode
    - Fixed the navbar
- Verify mobile/tablet layout for all pages and all user roles. (Previous semester indicated admin views was a problem.)
- Fix button spacing in forms.

3 - Additional Features:

- Add an audit log that is a part of a separate table in the database. It stores major actions that occurred like a normal log would. It does not simply store deleted items.
    - monitors creating and deleting accounts

What's been completed:

- Fix nginx errors so the website can be hosted.

- Get the site up and running on the VPN

- Create an automated process that tests the site by creating two admins, several teachers, many classes, and dozens of students.

    - Probable next step: Take that dummy data for testing purposes

- Add the ability for an admin to create a course. Verify students can join the course. Admins are teachers by default.

- 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

- 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.

- Fix the background when clicking the hamburger button on mobile/tablet mode

    - Fixed the navbar
- Verify mobile/tablet layout for all pages and all user roles. (Previous semester indicated admin views was a problem.)

- Fix button spacing in forms.

- Add an audit log that is a part of a separate table in the database. It stores major actions that occurred like a normal log would. It does not simply store deleted items.

- Fixed API endpoint call for the Course component

    - After user enrolls in a project, project was no longer being displayed inside the course component. Issue was because they had set up the wrong API endpoint to be called, so after the user joined a project, no data was being returned to load the projects
- Unit Tests That Have Been Completed:

    - Add-courses:
        - Tested navigate to course page
        - Tested initialize the component
        - Tested create the component
        - All worked and no errors
    - Add-student-project:
        - Tested that component is created
        - Tested that project info loads on initialization of the component
        - Tested that we get info on students in project and not in project
        - Tested that we are able to add students to project
        - Tested that we are able to remove students from project
    - Course:
        - This has errors:

        - Error: NG02200: Cannot find a differ supporting object '[object Object]' of type 'object'. NgFor only supports binding to Iterables, such as Arrays. Did you mean to use the keyvalue pipe?
        - also a page reload issue.
- Courses:
    - Test component creation Test
    - Initialization Test
    - Course Loading Test
    - Successful Registration test
    - Search courses Test
    - Run the test with ng test
- Create-project:
    - Tested that component is created
    - Initializes an empty submission form test
    - Tested project creation with a valid form submissioon
    - Project should *not* be cretaed when form is invalid test
- Edit-timecards:
    - Tested that component is created
    - Tested that name and last name of student of existing timecard is loaded when editing
    - Tested that an existing timecard can be updated
- Inactive-course:
    - Tested component creation test
    - Tested initialization test
    - Tested inactive courses loading test
    - Tested navigation test
    - Tested run the test with ng test
- Login:
    - Tested that component is created
    - Tested a successful login
    - Tested form submission with invalid input
- Project:
    - Tested that component is created
    - Tested that we can load users in specific project
    - Tested that we can load project info
    - Tested that we can load project timecards for project users
    - Tested that we are redirected to the edit-timecard component when clicking ‘edit’ on a timecard
- Register
    - Tests component creation
    - Tests form initialization
    - Tests form Validity
    - Tests successful Registration
    - Tests failed registration

    - User-profile
        - Tests component creation
        - Tests component variables are set
        - Tests default values for some variables are set
        - Tests that currentUserID and viewingUserID are set and compared
        - DOES NOT test http requests with loadProfile() using httpMock
    - View-report

Bugs:

- The Dashboard UI for an Admin when creating a course differs from that of an instructor
    - When a student registers for a course, their request should appear on the instructor for the classes dashboard under “Pending Requests to Courses”, for an admin, this does not appear.
- When viewing a project, if the user clicks the “Back to Course” button, the page is a blank course page when it should fill with the course details such as students, projects, and the course name.
- UI/UX Consideration: When timer is running, “Start” button stays as is. Consider switching this to be a “Stop” button to make it more intuitive for users.

Incomplete:

- Missing Unit Tests:
    - Assign-evals
    - Edit-course
    - Create-course
    - Dashboard
    - Edit-profile
    - Group (Unsure if this functionality has been fully built)
    - Reset-password
- Find a mechanism for unit tests to run often, such as per commit, or a separate website or executable that can be manually invoked.
- Ask client what actions can be monitored for the audit log.
- Undockerize the project
- Further testing with the eval components especially while the site is running on server.
- Expand the dummy database to integrate courses, projects, timecards, etc.

Recommendations

On Unit Testing:

- Command to run all tests at once on every commit.

- Complete mssing unit tests

On Database:

- SQLite is currently used as the project's database engine. While it's lightweight and easy to set up, it's important to be aware that SQLite **only supports one write operation at a time**. This can pose a **risk of blocking or failure** if multiple users try to perform write operations simultaneously.

- This limitation may not affect small-scale usage but could become problematic if the application grows in traffic or concurrency (e.g. multiple admins modifying data at once).

- **Reference**: [SQLite High Concurrency - When to Use SQLite](https://sqlite.org/whentouse.html#:~:text=High Concurrency,at any instant in time.)

- **Recommendation**:

  If concurrency-related issues arise during testing or production, consider:

  - Implementing **retry mechanisms** for failed write operations.
  - Migrating to a more concurrency-friendly database engine (e.g. PostgreSQL or MySQL) if scaling becomes necessary.
- Currently the database can fill with dummy data when the admin clicks ‘Test Database’ from their profile page.

  - Check with the client if this is something that should stay on the main database or if it should have a database of its own, possibly a copy.
  - If they don’t want a different testing database, should the program back up the current database info before adding to the database so there is not mix ups?
  - The dummy data generation only fills the user table meaning you are able to login. It could be worthwhile to add more integration into the database with projects and courses and evaluations and timecards and groups that the unit tests can use to check for validity.

On Angular Versioning:

- Currently on Angular v18
- Angular v20 to be released later in May
- **Reference:** Angular versioning and releases