Startup

Download Node: https://nodejs.org/en/download/

Download and setup MySQL: https://dev.mysql.com/downloads/installer/
- Make sure you select "Legacy Password" instead of the recommended password.

Change the MySQL username and password in the code to the username and password of whoever's computer the code is being run on.

- Leaving the username as "root" is ok.

In a terminal, navigate to the directory titled "backend" and run **node builddb.js**

- This command only needs to be run when the database needs to be built or rebuilt.
- If it fails, try running it a second time. (Usually, you do have to run it twice)
- You should see a stream of output with messages like "Preference Added!" as the queries are executed.

In the same directory, run **node server.js**

- You should see:
  Server is running
  Connected to MySQL!
  listening on 8081

In a new terminal, navigate to the directory titled "src" and run **npm install**

- This only needs to be done the first time the project is run.

In that new terminal, run **npm start** to run the app in your browser.

Backend:

NodeJS files to read input data from 4 CSVs to a MySQL database, respond to API calls from the frontend by returning or updating data in the database, and write the changes back to the studentAssignments CSV.

CSV Reading:

NodeJS file: **builddb.js**

Requirements: **fs** and **csv-parse**

4 Input CSVs:

1. **studentAssignments.csv** - Student data, project data, and student comments.
2. **studentsFinal.csv** - Student preference data (survey responders only)
3. **Students Without Prefs.csv** - student data for non-survey responders.
4. **projectsFinal.csv** - project data.

Because the CSVs are organized in a way to be useful for the Learning Factory and the PDF generator, the 4 CSVs are not simply read into the 4 database tables. Their data is spliced around to make it more queryable (see database table).

Functions:

- readCSVFile - takes a file path and a CSV parser object, and parses the CSV.
- readCSVs - calls readCSVFile for all 4 CSV paths. Creates the parser objects on the spot, which includes the variable the parsed data should be saved to.
    - The CSV paths and variables are declared at the top of builddb.js.

CSV Writing:

NodeJS files: **csvWrite.js**
Requirements: **fs**, **sql-template-strings**,
require("csv-writer").createObjectCsvWriter

The CSV writing function runs 4 queries to gather the data needed to create a new CSV called “tempStudentAssignments.csv”.

Functions:

- writeAssignmentsCSV - Runs the below queries and passes the data to the CSV writer. Change the output file path or column structure at the top of this function.
- getAssignments - runs a select query to get every student ID/project ID pair.
- getStudentDataForWrite - first_name, last_name, major, NDA, IP, on_campus
- getPreferenceDataForWrite - comment, time_a, time_b, time_c

- getNoSurveyTimeForWrite - time_a
    - Time_a is the time of the course the student signed up for, so it is available even for non-survey students. Time_b and time_c rely on preference data from the survey.

MySQL Database:

NodeJS files: **builddb.js** and **db.js**
Requirements: **mysql2** and **sql-template-strings**

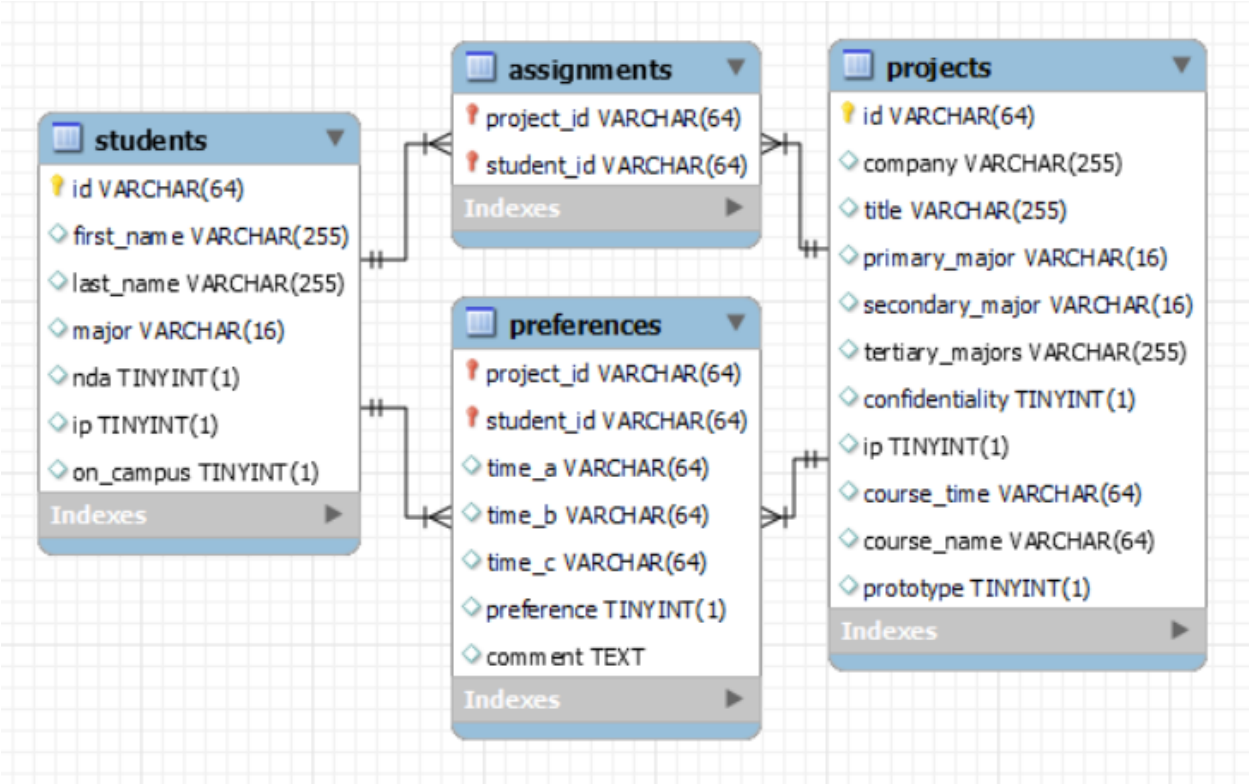
Database Creation:

The database is created programmatically by running “node builddb.js”. This should only be done when the database does not already exist. If the database needs to be refreshed from the CSV data for some reason, use the MySQL Workbench app to drop the database and then run builddb.js.

The main code in builddb.js is at the bottom of the file. The functions do pretty much what their names say, so tracing the function calls starting from the ones called in the main code gives a good idea of the entire process:

- databaseSetup - connects to MySQL (make sure your credentials are correct. You might also have to set your MySQL to the older authentication system) and runs queries to create the database “capstone” and the tables shown below.
- readCSVs - discussed above.
- loadTables - calls each table’s load function, which calls a function in db.js to run an insert query with the data taken from the CSVs.

Database Layout:



Queries:

Most of the queries are stored in db.js, but some are also in builddb.js. Once again, their names are mostly self-explanatory. Call the function and pass in the values you want to use with your query. The main queries are to add entries to the tables and retrieve data to either display or export to CSV.

Front end documentation for developers

How to use:

1. Clone the repository
2. Ensure you have node.js installed
3. Use terminal/cmd to find directory, then once within file use ‘npm install’
4. Ensure CSVs are loaded into directory to be used
5. **If using mac, download workbench**
   https://dev.mysql.com/downloads/workbench/

Main application is run through App.jsx in /src code

App.jsx takes 4 components created in /src/pages consisting of

- Dashboard.jsx
- Exports.jsx
- Table.jsx
- Students.jsx

Packages used:

Package.json contains all the dependencies and used packages

Front end components:

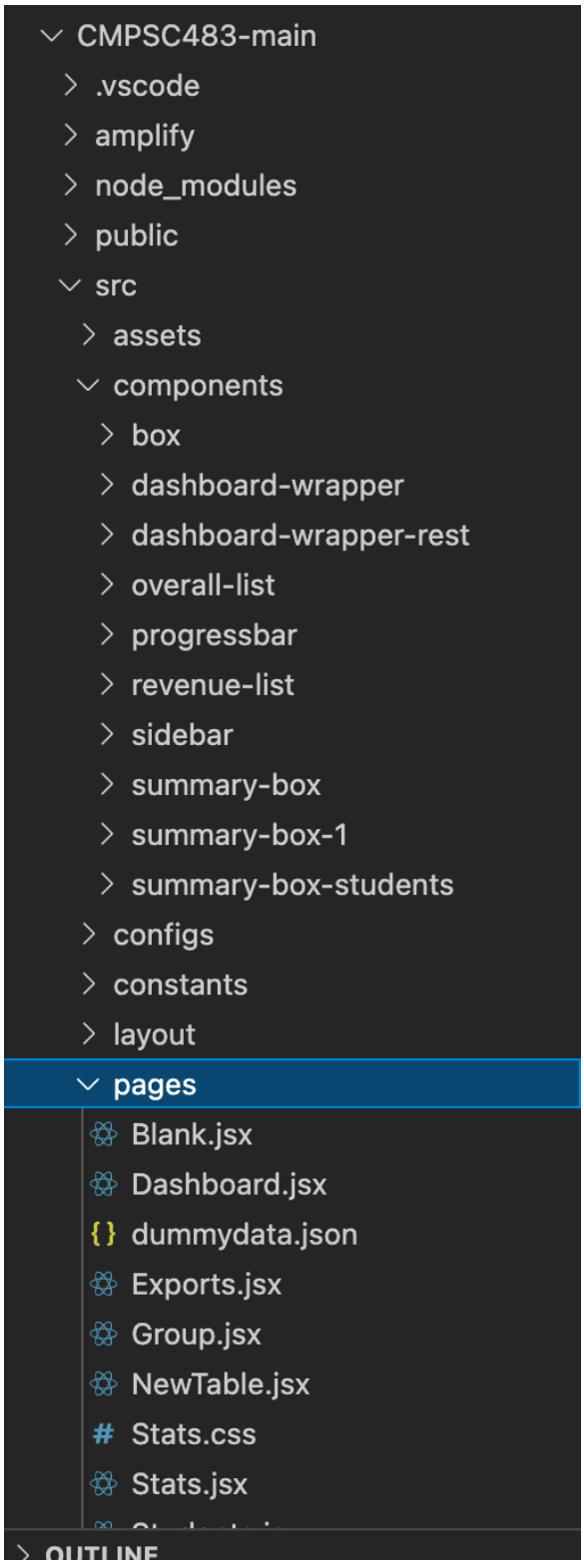


Table.jsx

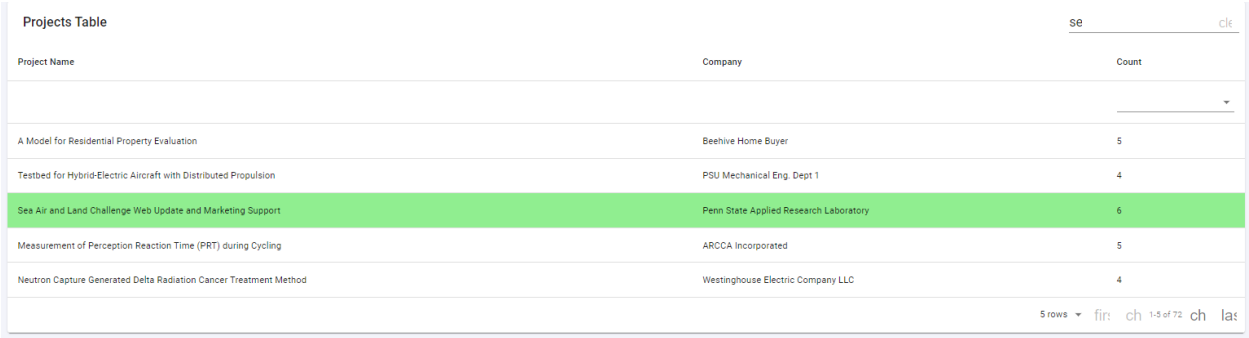
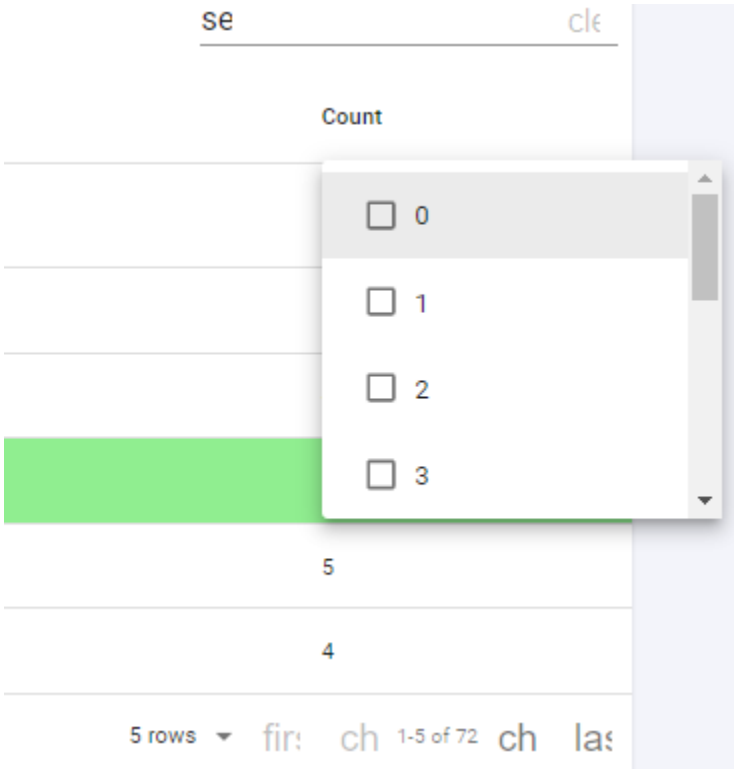


Table.jsx contains the tables with all capstone group projects and sponsors
When a team is selected as above, directly below displays all the group members with their major

Additionally, teams can be filtered by group member count, with multiple selections ranging from 0 to 10 group members



Export.jsx:

Export, has a single button that when pressed exports the updated database to a CSV file and stores in the **backend** directory

Students.jsx:

Students.jsx contains the students who are not assigned to groups, and helps with assigning them to a desired capstone team. After a student who is not assigned to a team is selected, or one without a preference is selected, then assigned sequentially where needed.

Dashboard.jsx:

Dashboard.jsx contains the ‘home page’ and shows general statistics by groups that are completed, students who are assigned, and the major breakdown of what students are currently enrolled in capstone. There is a bar graph for the major breakdown, and a circular graph that changes colors conditionally on how close students are to being fully assigned.

Within dashboard summary boxes there is a circular bar that changes based on color as well, if <50, color is red, if >50 && <100, purple, and if 100 the color is green

CSS:

CSS is imported and can be changed as desired, there are several CSS files, but the main styling is in style.css which is found in main directory. Additional CSS files can be found in ./src throughout the different components.

In addition to CSS styling, there are .js files that contain things such as coloring and some other general aesthetic features contained in ./src

Additional components:

Within each main component there are some additional components that are called that all are within the ./src files, primary example is from dashboard.jsx there is a component called dashboardrapper, which is called from ./src/summary-box1.jsx that contains this component that can easily be found from searching the definition within the dashboard.jsx file.

There are several of these components which can be found, but all the main code structure is found within ./src/pages where it is exported then called in app.jsx