CSCI 4140 A3

Dhairy Raval B00845519

Assignment uploaded to Github:
add link here

Table of Contents

→ A3-519.pdf:

1. Software & Libraries used:

Software:

- MySQL Workbench - database (local instance)
- VS Code Editor - creating the UI files and backend server files
- React - frontend
- Node + Express - backend
- GitHub - version control and sharing the project w/ TA & Prof
    - URL - add link here

Libraries:

- body-parser - used to parse the data from frontend
- cors - used w/ Axios to connect to DB
- express - hosting backend
- mysql - SQL library used to connect to DB
- nodemon - used to make development faster for backend

2. Software Organization (REST description):

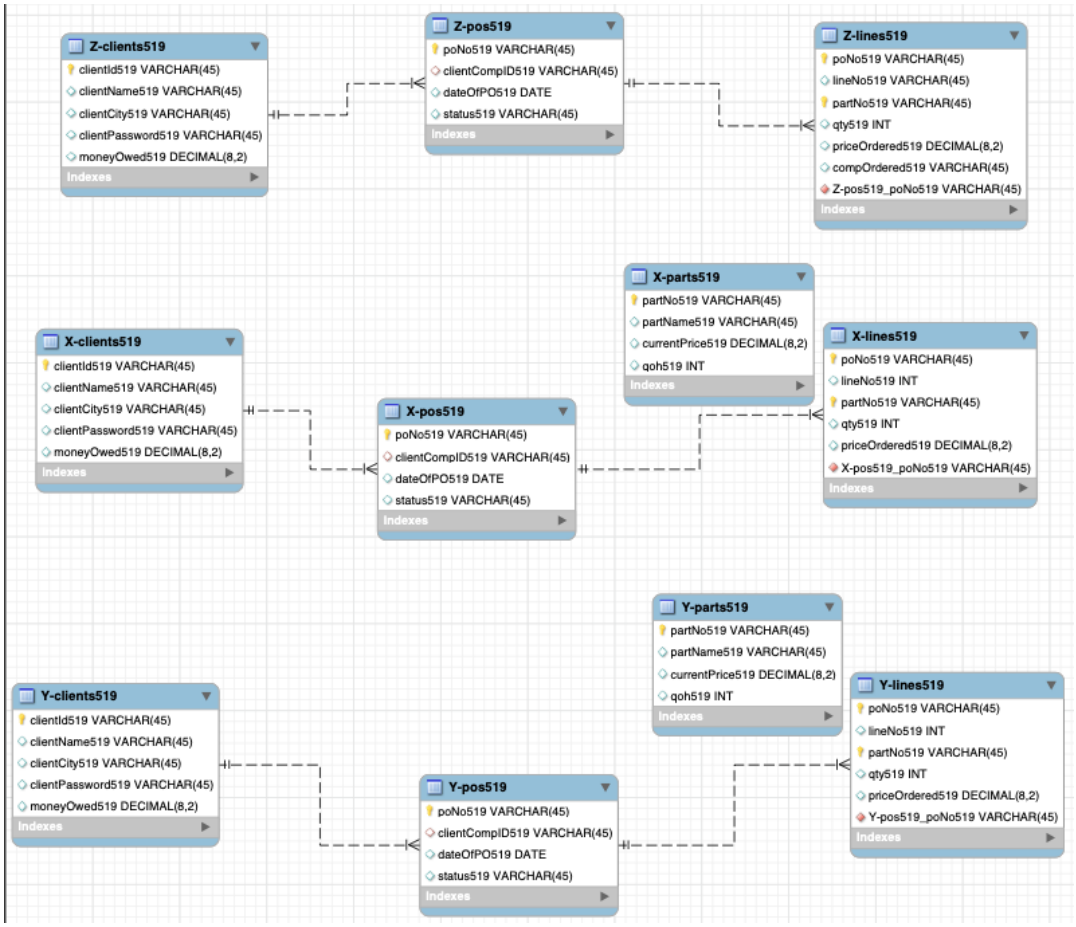
The assignments in completed on a local instance of mySQL Workbench using a single DB and following the naming convention when creating multiple tables to simulate 3 separate DBs. (Namely: X, Y, Z).

There are mainly 2 REST calls in the program.

1. **To list all available parts:**
   Using Node + Express we connect with the instance of MySQL and send 2 GET requests one to *X-parts519* and one to *y-parts519*. After we successfully receive a response, we add the rows into an array while making sure that the duplicate *partNames* are ignored. **URL:** `/api/listParts`

2. **To submit a PO:**
   Similar to the List all parts GET request, to submit a PO, we connect to MySQL and send 1 POST request to the DB. Here, we first have to get the all the details of parts of X & Y. Then we decide we the requested PO can be accomplished using the current Quantity on Hand of the requested parts.
   If the PO can be accomplished, we create the POs and Lines as required.
   URL: `/api/insertPo`

3. ER Model:



4. **SQL scripts & folder description:**

This section describes the folder structure of the 4140 A3 submission. The folder is uploaded as a zip file along with the index.pdf file to Brightspace and Github.

* NOTE: Github contains a single folder: 4140 A3 submission with all the following info.

4140 A3 Submission

- **4140 A3 Code.zip**
    - This is the code to run the React + Node/Express app to connect with the DB
    - To start the front-end run - **`npm start`** inside the *client* folder
    - To start the back-end run - **`npm run devStart`** inside the *server* folder

- **table-creation.sql**

script file to create all the required tables

- **insert.sql**

script file to populate all the tables with sample data

- **index.pdf**

The index file (this file)

5. **Results:**

When running the program, the client side (React) runs on localhost:3000 and the server side on localhost:3001.

- When you check “List Parts for sale” and submit the form the page will look like the following: The program will display parts from X & Y (removing the duplicates)

fig 1. List all the part from X or Y

The List of parts appears at the bottom of the page.

- When you enter valid form data as such:

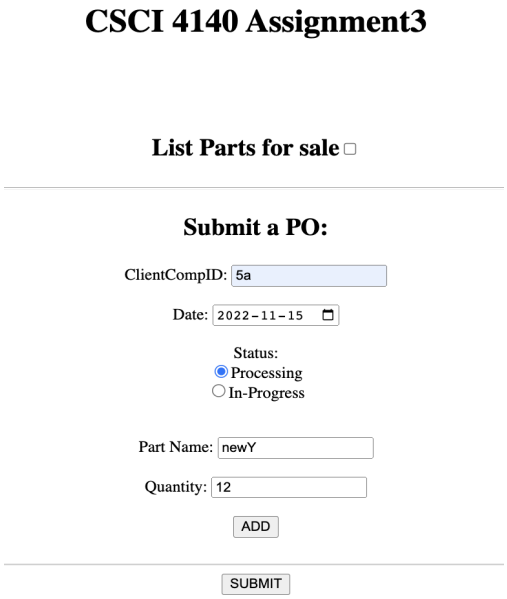
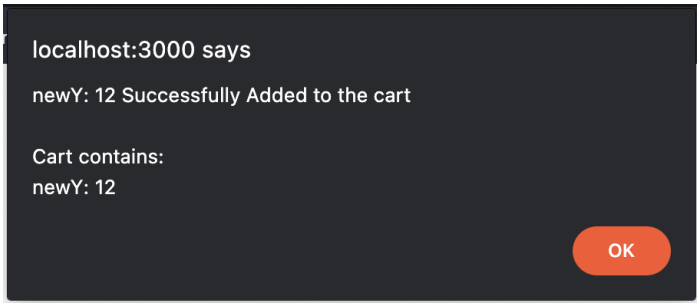


fig 2. Form with data

Once we enter all the fields of the form we can click on the “ADD” button to add the part and quantity desired to the shopping cart. We get the following message when we click on “ADD”:



We can continue changing the Part Name and Quantity till have the desired items in the quantity we need.

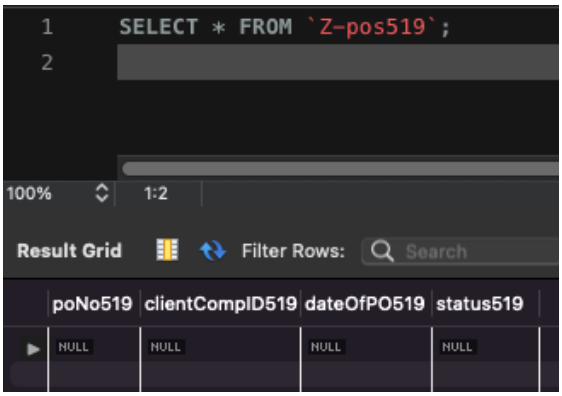
When the user clicks on the “SUBMIT” button after adding items into their cart and providing valid details for the PO. A new PO will be created in the table *Z-pos519* with a unique PO number.
After that the program attempt to find the parts from either company X or company Y and create their POs respectively.

To demonstrate this, we will be submitting the form with the details in fig 2 (Form with data).

***Note:** The part *‘newY’* is only present in table *Y-parts*, this will help demo the functionality of only creating POs for companies whose parts are required in the original PO.

Before submitting the form our tables in the DB look as follows:

Z-pos519:



Z-lines519:

```
1  SELECT * FROM `Z-lines519`;
2
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

poNo519	lineNo519	partNo519	qty519	priceOrdered5...	compOrdered5...
NULL	NULL	NULL	NULL	NULL	NULL

X-pos519:

```
1  select * from `X-pos519`
```

100% 25:1

Result Grid | Filter Rows: Search

poNo519	clientCompID519	dateOfPO519	status519
po1	1a	2022-10-02	processing
po2	1a	2022-09-30	in-progress
po3	2a	2022-09-30	in-progress
po4	3a	2022-10-02	processing
po5	4a	2022-09-29	completed
po6	5a	2022-09-29	completed
NULL	NULL	NULL	NULL

Y-pos519:

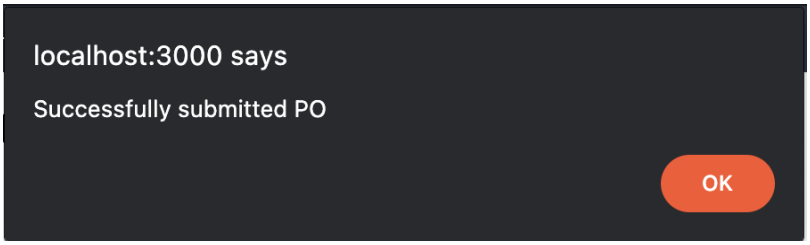
```
1  select * from `Y-pos519`
```

100% 25:1

Result Grid | Filter Rows: Search

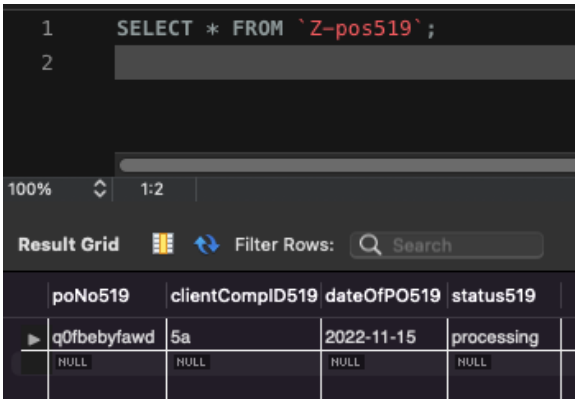
poNo519	clientCompID519	dateOfPO519	status519
po1	1a	2022-10-02	processing
po2	1a	2022-09-30	in-progress
po3	2a	2022-09-30	in-progress
po4	3a	2022-10-02	processing
po5	4a	2022-09-29	completed
NULL	NULL	NULL	NULL

When we click on submit we get the following message:

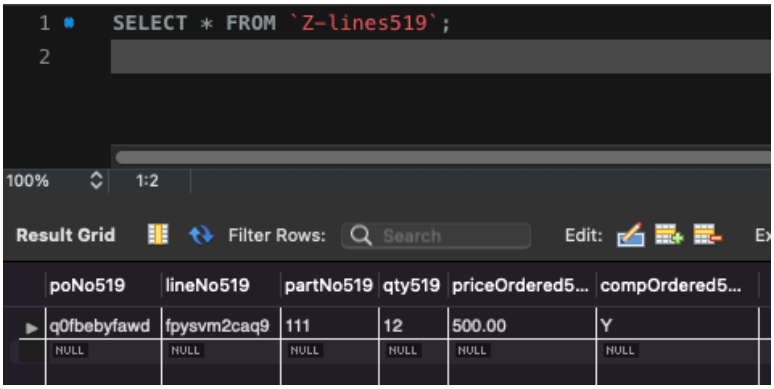


After submitting the form our tables in the DB look as follows:

Z-pos519:



Z-lines519: (**compOrdered** tells us which company the part was ordered from)



X-pos519: No change observed in this table as no part ordered from X

```
1 •  select * from `X-pos519`
```

100% 25:1

Result Grid Filter Rows: Search

poNo519	clientCompID519	dateOfPO519	status519
po1	1a	2022-10-02	processing
po2	1a	2022-09-30	in-progress
po3	2a	2022-09-30	in-progress
po4	3a	2022-10-02	processing
po5	4a	2022-09-29	completed
po6	5a	2022-09-29	completed
NULL	NULL	NULL	NULL

Y-pos519: (New field created)

```
1 •  select * from `Y-pos519`
```

100% 25:1

Result Grid Filter Rows: Search

poNo519	clientCompID519	dateOfPO519	status519
po1	1a	2022-10-02	processing
po2	1a	2022-09-30	in-progress
po3	2a	2022-09-30	in-progress
po4	3a	2022-10-02	processing
po5	4a	2022-09-29	completed
q0fbebyfawd	5a	2022-11-15	processing
NULL	NULL	NULL	NULL

6. **LIMITATIONS:**

- The program would work as expected if the user enters valid data however if the user were to enter invalid data. The request to the SQL server won't work and the PO wouldn't be submitted. The program will show an error message but it won't be precise.

- We cannot place a PO where some quantity of an item will be fulfilled by company X and the rest by company Y. For example, if we want 10 units of some part 'a', we cannot get 5 units of 'a' from X and 5 from Y. All 10 need to be from one company.

- The method used to generate random alphanumeric strings has a small probability of producing duplicate strings (1 in ~ 70M), which will result in the PO not being saved in the DB.

7. **REFERENCES:**

- A simple method to generate random alphanumeric strings (used for PoNo. and LineNo.)
  → Date: Nov 10, 2022
  → URL: https://stackoverflow.com/questions/10726909/random-alpha-numeric-string-in-javascript