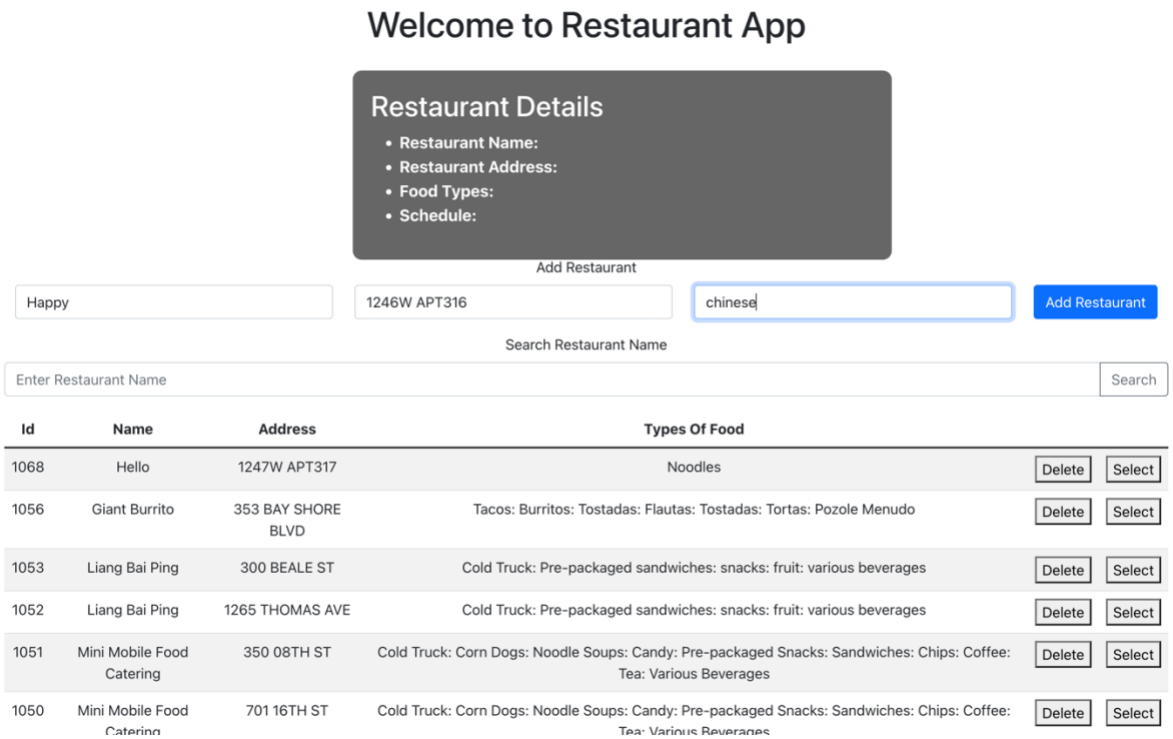
1. Overview of the application:

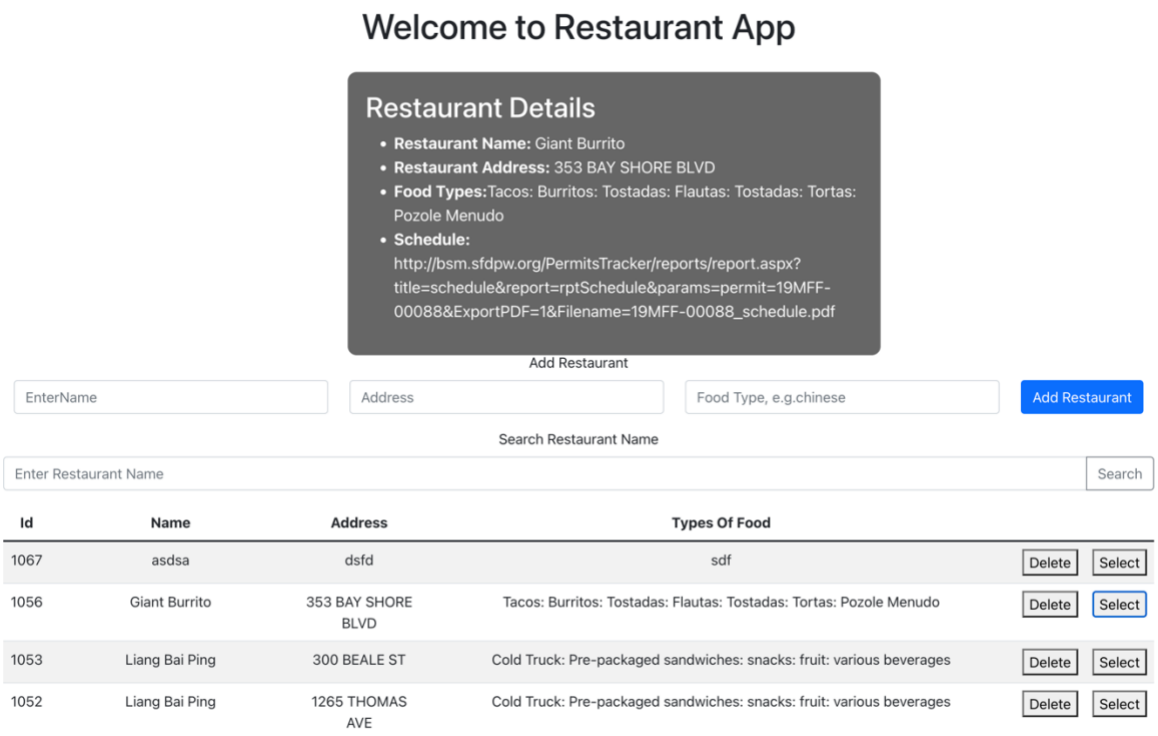
Food truck/restaurant application that shows the list of restaurants and food trucks in San Francisco. Used a CSV file that includes all the information of the Food truck/restaurant to initial data for the site. The site allows users to:

1) Add restaurants to the current restaurant list:

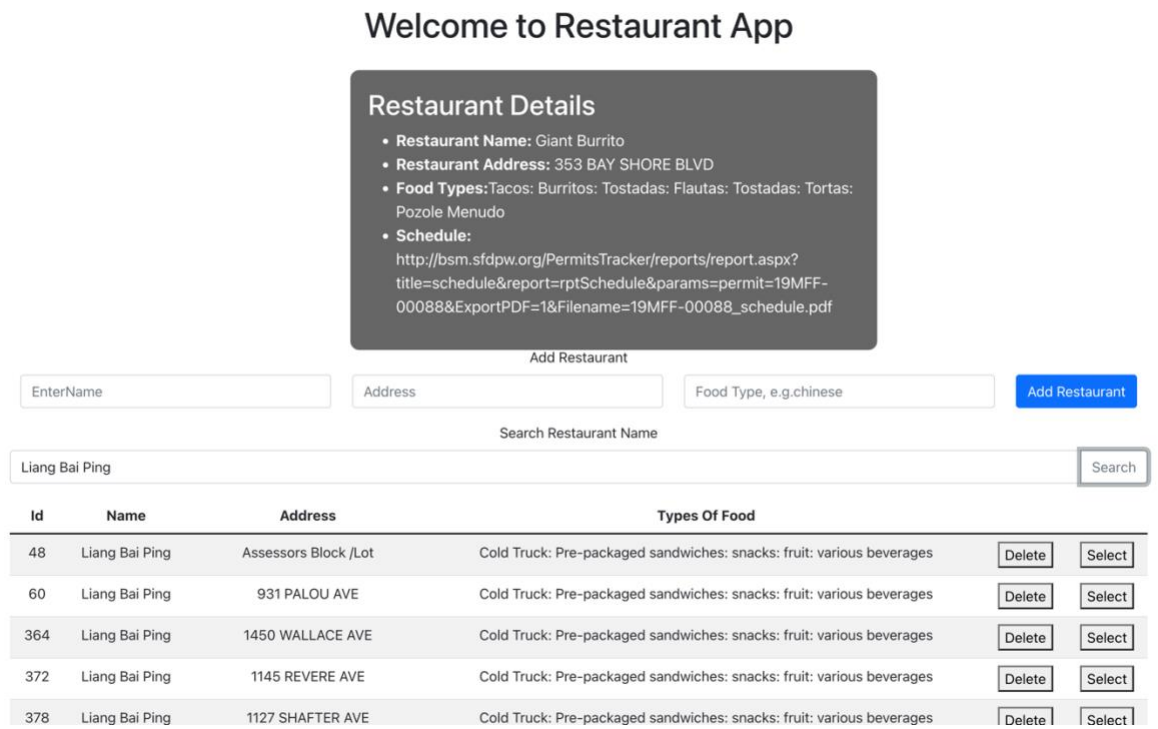


2)Click on a restaurant to view the restaurant details:

- Name
- Address
- Food types
- Schedule



2) Users can sort the list of restaurants by name:

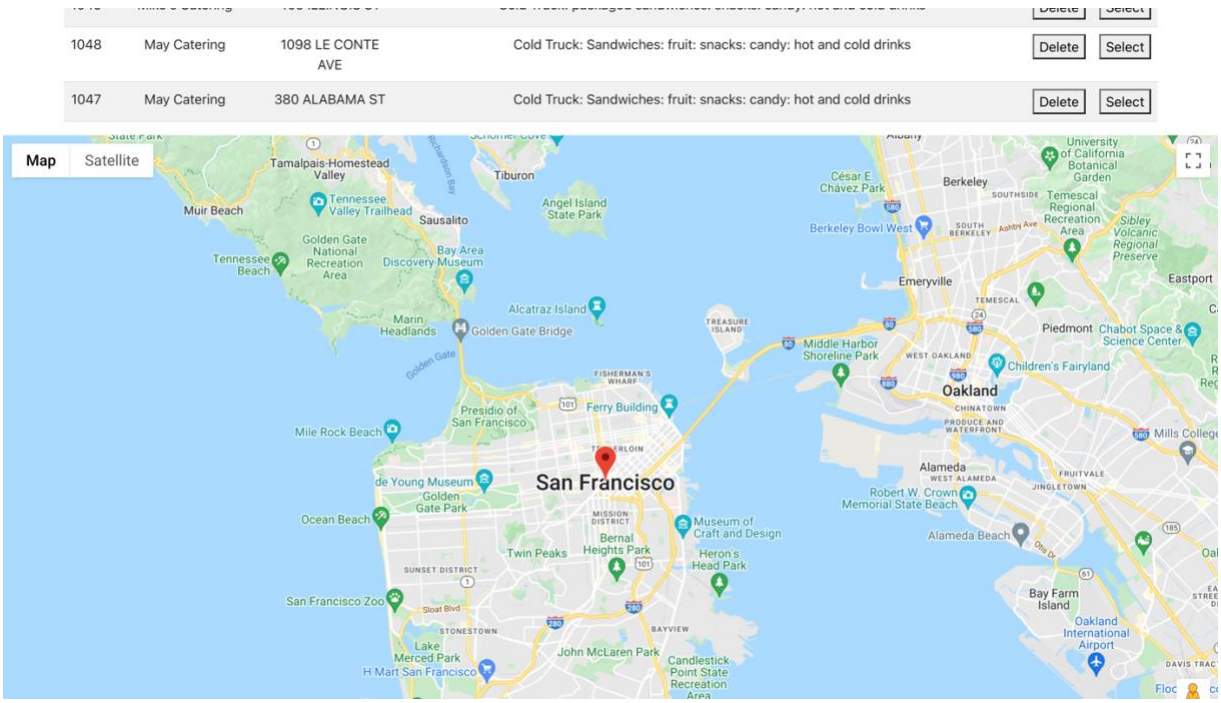


3) Integrated with Google Map API to show the city map of San Francisco where the food trucks/restaurants are located (Integration with a 3rd party RESTful API)

Usage of Object-Oriented principles inheritance:

GoogleMap class extends from Component, inheritance functions in Component class.

Screenshots of the app where distinct design decisions:



2. instructions to install and start the app (For mac users):

Backend:

1) Install Brew first
2) Then Install MySQL, in the terminal type these commands below:
   brew install mysql
   mysql_secure_installation
3) Start MySQL:
   brew services start mysql
4) Log into MySQL:
   mysql -u root -p
   enter password
5) Open restaturantBackend java folder run spring-boot in Terminal:
   ./mvnw spring-boot:run
6) Use http://localhost:8080/demo or Postman to check data to ensure the backend works properly

Frontend:

1) Open restaturantFrontend React app run npm in Terminal:
   npm start

2) Use http://localhost:3000/ to view the web application and do interactions

3. App overall architecture pattern:

Use MVC (Model, View, Controller)

View - Frontend React app loads restaurant list and interacts with users, fetches different APIs to convey user requests to the controller, and gets model updates from the controller.

Controller - The Backend java code mediates between the view and the model. Spring Boot RESTful API can create, retrieve, update, and delete (CRUD) client data with the user request from the view and data in the Model.

Model - MySQL Store data file of the restaurant list for the project, also get commands from the controller to first store then update data. Postman helps to load the original CSV file to the MySQL database and helps to test the RESTful API.

4. Expected payload and response for at least one route of the REST API (create/add):

In Spring Boot backend java project:

1) Maven Dependencies to pom.xml file:
web, JPA persistence starters, as well as the MySQL dependency…
2) Created the Model DemoObject Class with id, applicant, address, photo, foodItems, sechdule, location…
3) Created Repository the interface DemoObjectRepo extend CrudRepository<DemoObject,Long>, JpaSpecificationExecutor(){} (Used Object-Oriented principle inheritance)
4) Created the DemoObjectController which exposes a REST API by interacting with the Repository

Example Spring Boot RESTful API create:

In the DemoObjectController class

```
@PostMapping("/demo/add")
   public DemoObject add(@RequestBody DemoObject rest) {
      repo.save(rest);
      return rest;
   }
```

The method above allows the application to create a restaurant element and add it to the database storage.

In the Frontend React app:

AddRestaurant.js function AddRestaurant()

```
function handleSumbit(){
        //console.log(state);
        fetch('demo/add', {
            method:  'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(state),
         }).then(e => {listUpdated([state,...restaurants])});


    }
```

The function above fetches the API to call the backend spring-boot to add the new restaurant element with the input from the user to the database. After the database created the id for the restaurant the browser reloads everything.