FULL-STACK WEB APPLICATION

PROJECT OVERVIEW:

The application we developed utilizes the latest versions of Angular for the frontend and Spring Boot for the backend. Its primary functionality is to enable prospective students to share feedback about their campus visit via a survey form. Additionally, users will have access to view all survey records submitted thus far.

The survey form will encompass various input fields, allowing students to express their preferences regarding the campus and elucidate their interest in the university. This project serves as an opportunity to delve into full-stack web application development.

Technologically, the application leverages Angular, Node.js, Spring Boot, RESTful API web services and MySQL database. We additionally incorporated CRUD operations to enhance user interaction.

On the homepage, users are presented with two distinct buttons leading to separate pages: one for accessing the survey form and the other for viewing all submitted surveys. Users can effortlessly submit their feedback via the survey page. Furthermore, clicking on the "list all surveys" button directs users to a page showcasing a comprehensive list of submitted surveys. Each survey entry is accompanied by options for updating or deleting the respective record, providing users with control over their submissions.

Technologies:

- Angular
- Node.js
- Spring Boot
- Restful API

IDE:

- Visual Studio Code

Database:

- MySQL server and work bench

FRONT-END(Angular, Node.js):

Angular and node.js Setup:

- Installed Node.js and npm according to my OS specification.

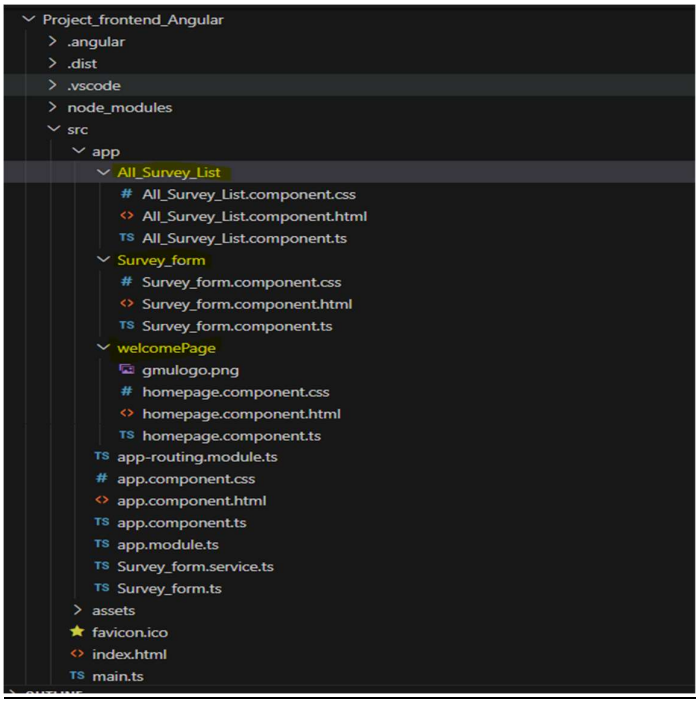
- Installed Angular CLI for executing Angular commands: **npm install -g @angular/cli**.
- Created a new Angular project using the command: ng new your-project-name. i.e. **ng new Project_frontend_Angular.**

Component Generation:

- Generated components required for the application using the ng generate component command.
- Components created:

  Welcome Homepage: '**ng generate component welcomePage'**.
  Survey Form**: 'ng generate component Survey_form'**.
  Survey List: '**ng generate component All_Survey_List'**.

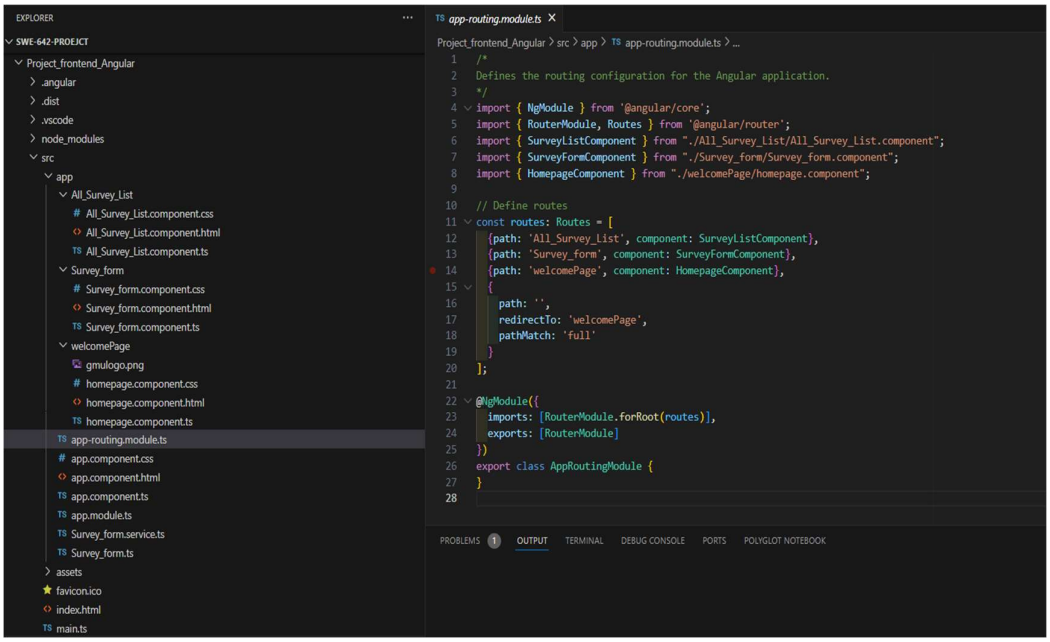
- Each component has its own .html, .css, and .ts file.



Angular Router Setup:

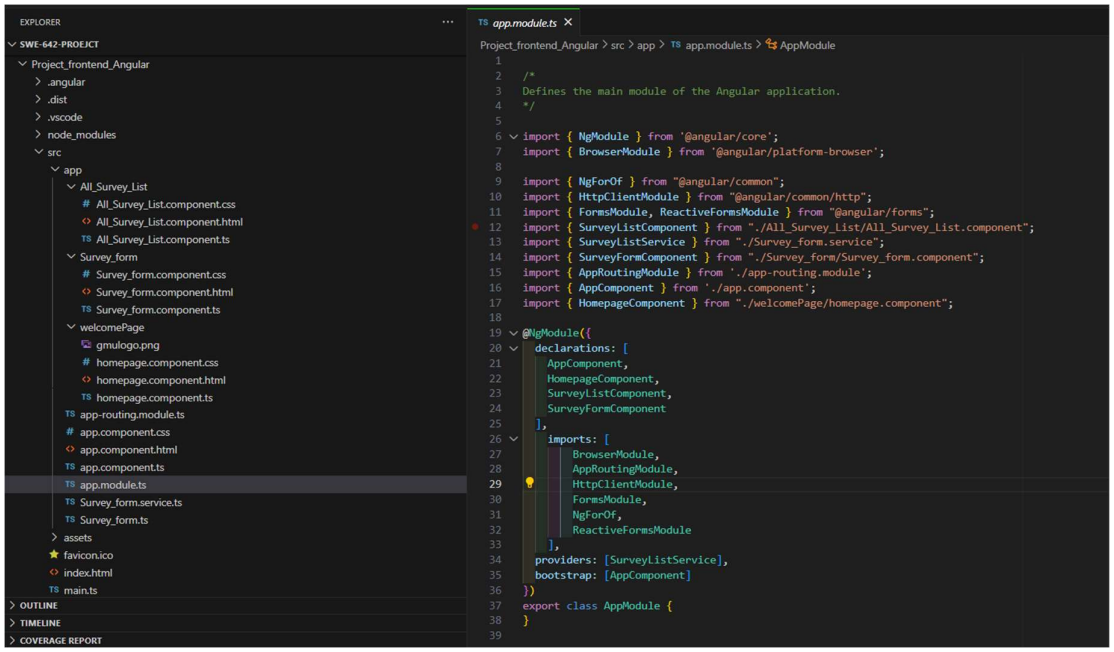
- Implemented seamless navigation between multiple views using Angular Router.
- Generated the necessary routing module file using the command: **ng generate module app-routing --flat --module=app**.

- This created ***app-routing.module.ts***, defining routes specifying which component to load for each path.



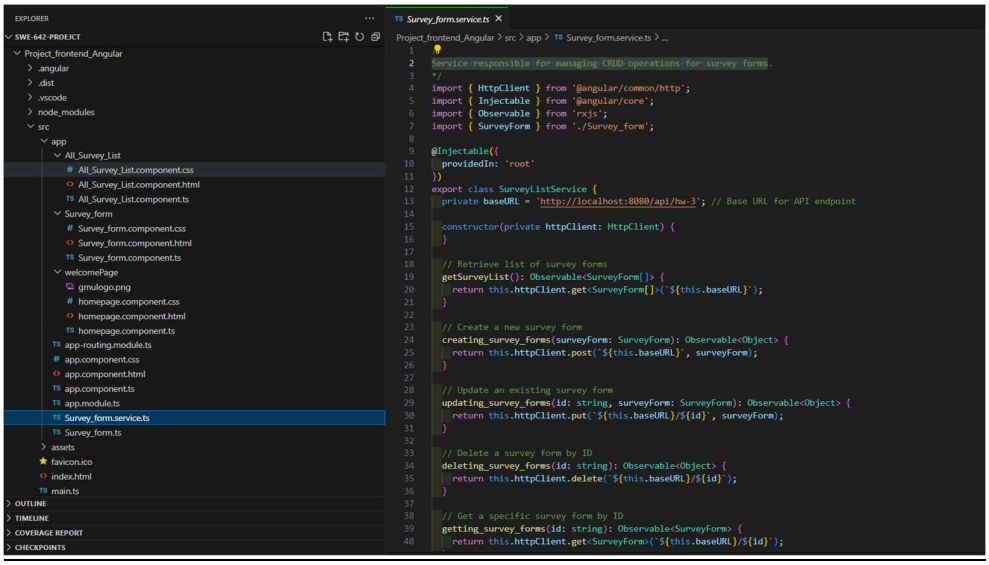
App Module Configuration:

- Defined the main module (**AppModule**) of the Angular application in ***app.module.ts***.
- Imported necessary modules like **BrowserModule**, **HttpClientModule**, **FormsModule**, **ReactiveFormsModule,** and **AppRoutingModule**.
- Declared all components used in the application (**AppComponent, HomepageComponent, SurveyListComponent, SurveyFormComponent**).
- Provided *SurveyListService* as a provider for managing survey data.
- Specified **AppComponent** as the root component to bootstrap the application.



Service Generation and Implementation:

- Generated a service named "Survey_form" using the "**ng generate service Survey_form**" command.
- This created a new service file named ***Survey_form.service.ts*** in the src/app directory and updated ***app.module.ts*** to include the new service in the providers array.
- Implemented services responsible for managing CRUD operations for survey forms.
- API call methods were defined in the ***survey.service.ts*** file.



Component Implementation:

- Developed HTML, CSS, and TypeScript code in the corresponding components.
- Implemented API call methods in the ***survey.service.ts*** file for data interaction.

Running the Application:

- Executed the angular application using the **ng serve** command.

```
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\panth\SWE-642-PROEJCT\Project_frontend_Angular>ng serve
Initial chunk files | Names         |  Raw size
polyfills.js        | polyfills     |  83.60 kB |
main.js             | main          |  68.63 kB |
styles.css          | styles        |  95 bytes |

                    | Initial total | 152.33 kB

Application bundle generation complete. [3.314 seconds]

Watch mode enabled. Watching for file changes...
  → Local:   http://localhost:4200/
  → press h + enter to show help
```

By following these steps, the front-end of the Angular application was developed, allowing seamless navigation between views and efficient management of survey data.

BACK-END (Spring Boot, RESTful Web Services, JPA):

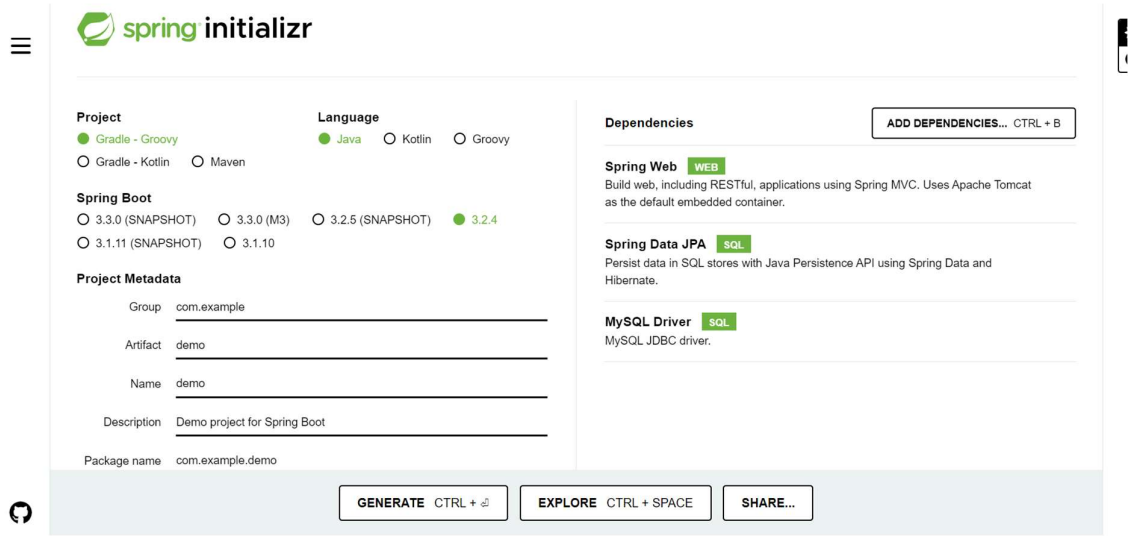
Tools and Technologies Used:

- *Maven*
- *Spring Boot*
- *RESTful Web Services*
- *Spring Data JPA*
- *Database (MySQL Workbench)*

Setup and Installation:

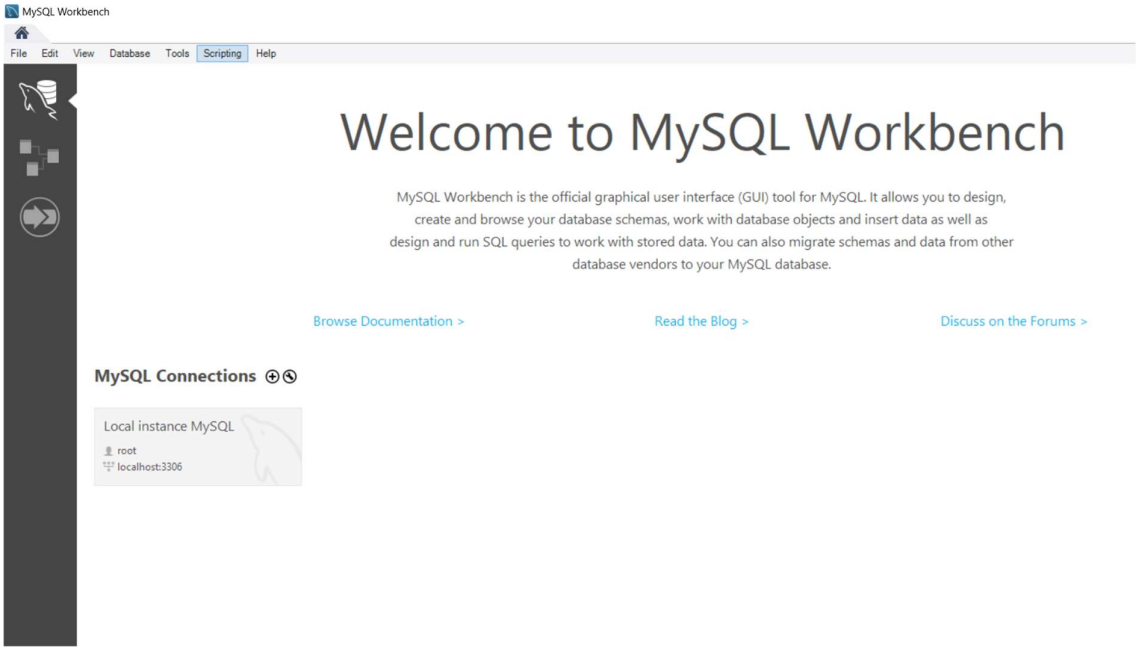
Navigated and Configured Spring Boot:

We went to https://start.spring.io/ and selected Maven. We added **Spring Web, Spring Data JPA, and MySQL Driver** as a dependencies to the project. we explored the generated Spring Java project to determine further dependencies.



MySQL Setup:

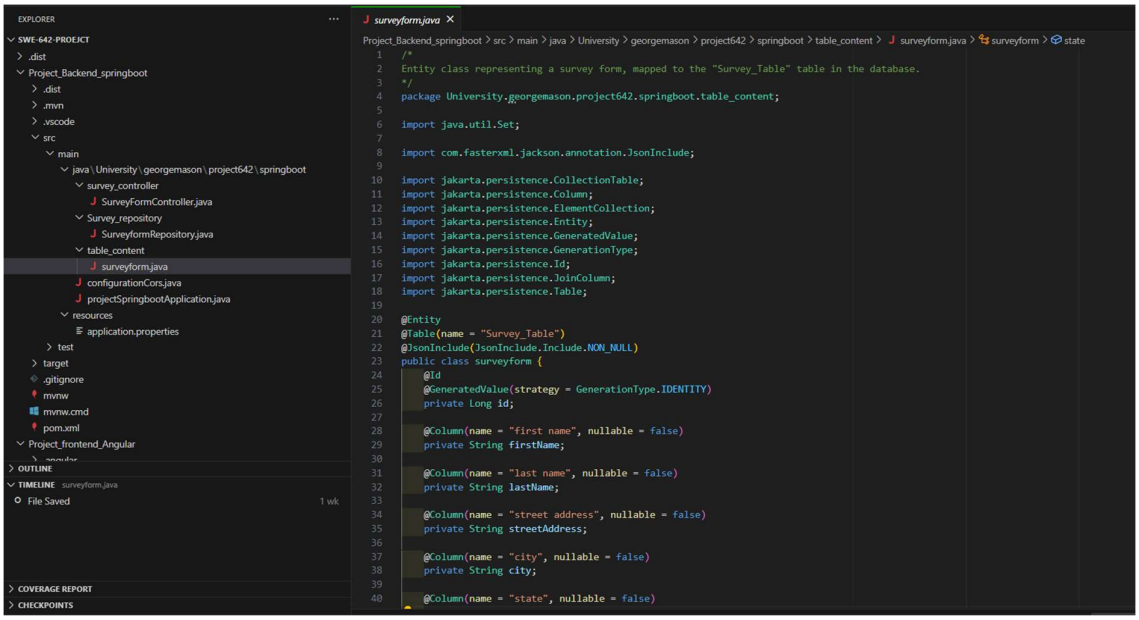
We downloaded and installed MySQL Server and Workbench from the official MySQL website. We configured Workbench to connect to my Local instance MySQL server by providing the hostname, port, and credentials.



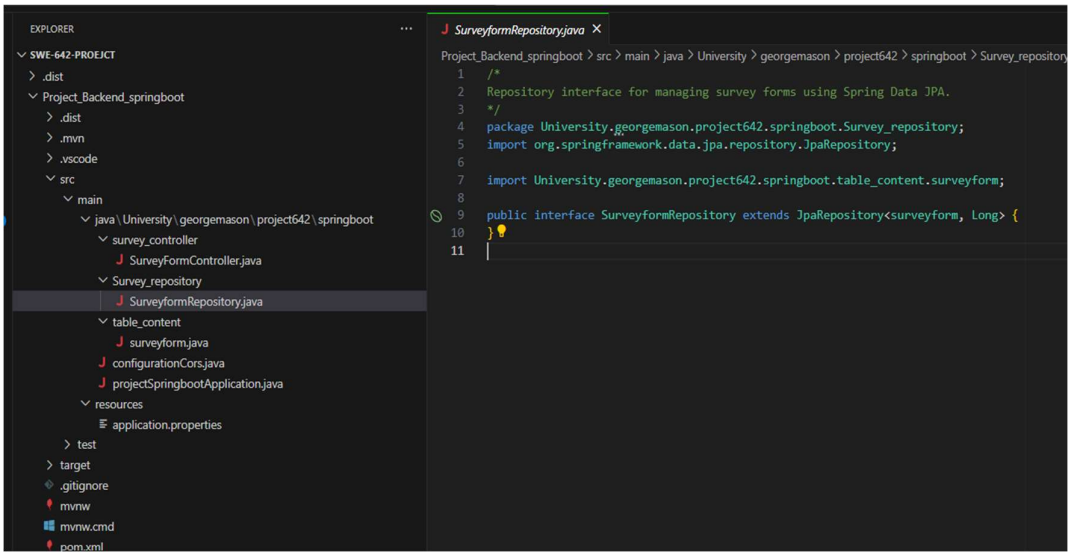
RESTful Endpoints:

Survey Entity, Repository, and Controller:

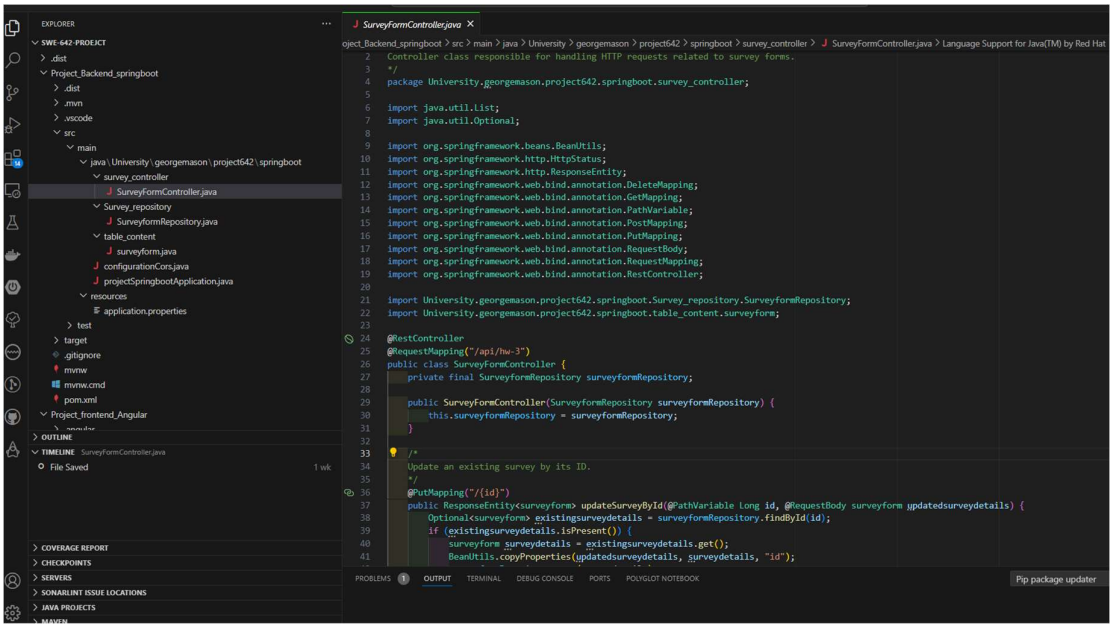
- We created the table_contents entity class(surveyform.java) with necessary fields and annotations.



- We developed the *SurveyformRepository* interface extending JpaRepository<Survey, Long> to handle database operations.



- We crafted the *SurveyformController* class annotated with **@RestController** and mapping to **"/api/hw-3".**



- We implemented CRUD operations in the SurveyController for creating, reading, updating, and deleting surveys.

Endpoints:

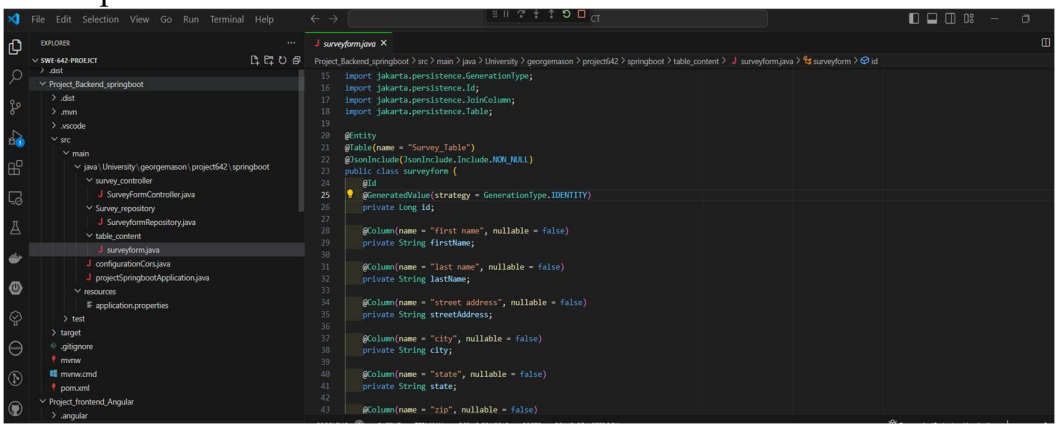
- POST **/api/hw-3**: creating a new survey.
- GET **/api/hw-3**: retrieving all surveys.
- GET **/api/hw-3/{id}**: retrieving a specific survey by ID.
- DELETE **/api/hw-3/{id}**: deleting a survey by ID.

- PUT **/api/hw-3/{id}**: updating a survey by ID.

Database Interaction:

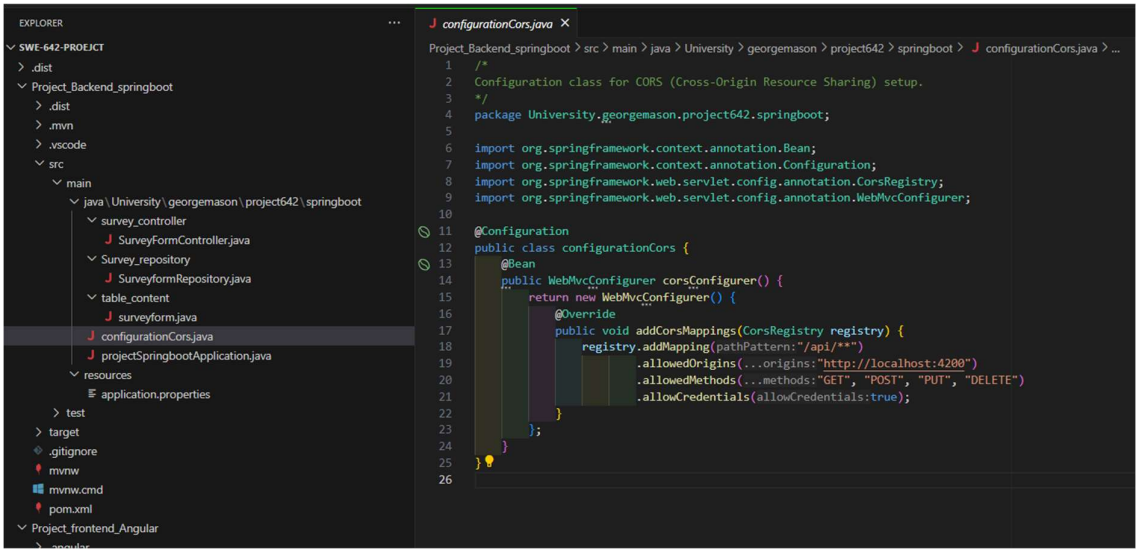
Schema and Tables Creation:

we created 'h3project' schema if not exist, and ' Survey_Table' table with necessary fields. Also, we created a table 'liked_most_options' for handling liked options.



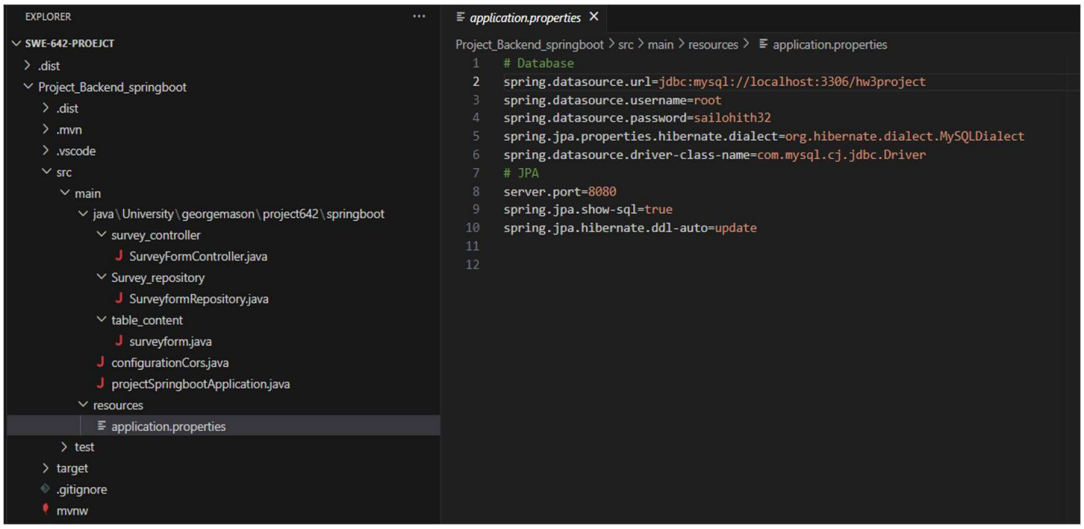
CORS configuration:

This configuration class sets up CORS (Cross-Origin Resource Sharing) for the Spring Boot application. It allows requests from the specified origin (http://localhost:4200) and permits specified HTTP methods (GET, POST, PUT, DELETE) for requests to paths starting with /api/. Additionally, it allows credentials to be included in CORS requests.



application.properties Setup:

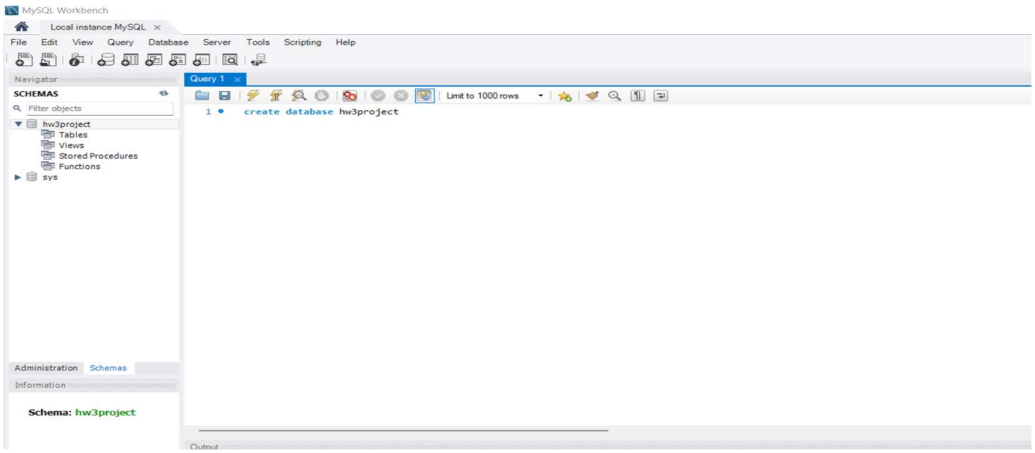
- We updated application.properties in our project's resources directory.
- We configured data source properties to connect to my MySQL instance.
- we configured JPA properties such as dialect and hibernate properties if needed.
- we used spring.jpa.hibernate.ddl-auto=update to generate or update the database schema based on my entity classes during development.



Usage:

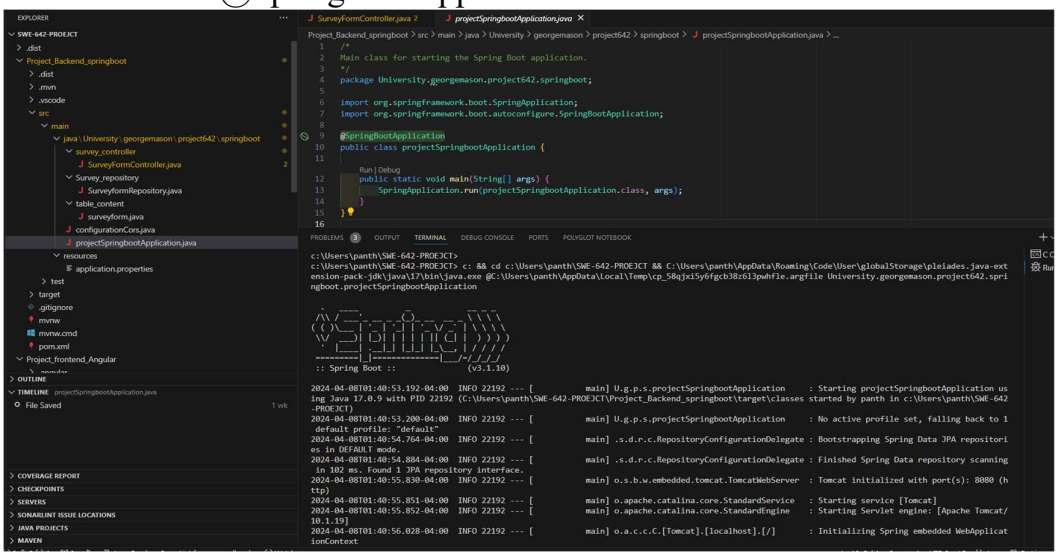
We ensured that the Survey entity class(surveyform.java) is annotated with @Entity and SurveyformRepository interface with @Repository. We updated the surveyform @Entity class to match the database schema.

Create database in MySQL:

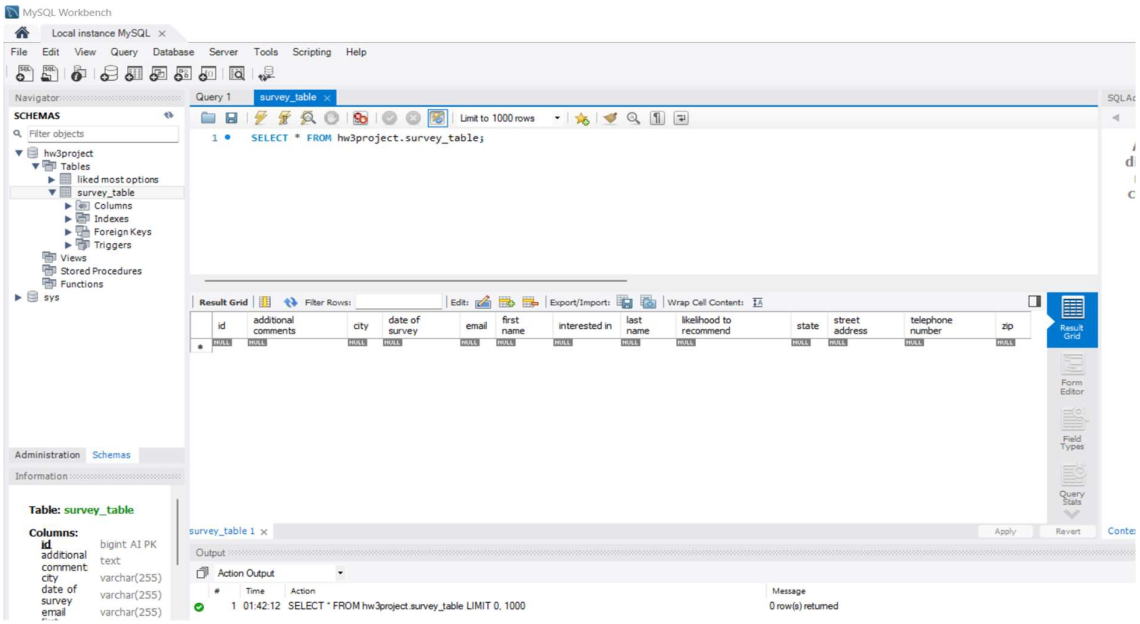


Running the spring boot application:

We should run @SpringBootApplication.



After successful running of springboot application, we can see the table content is displayed in the database.



After starting the spring boot application you can access this using this url: http://localhost:8080

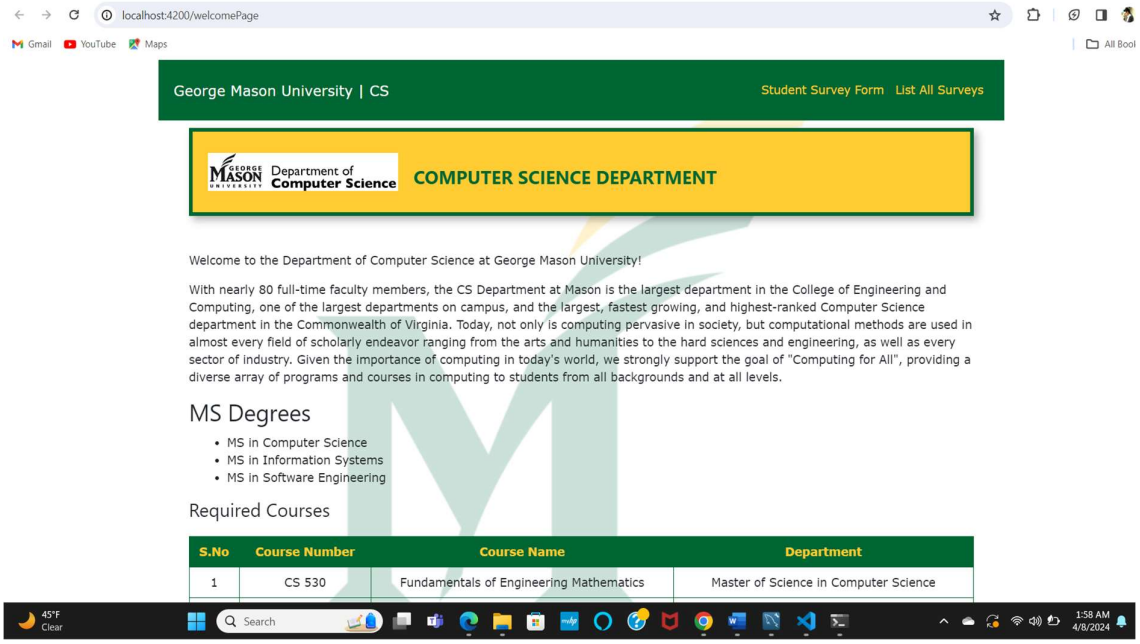
By following these steps, we seamlessly set up our Spring Boot application with MySQL database integration and RESTful endpoints for managing survey data.

Application Demonstration:

- Firstly run the springboot application(backend) and then run the angular application using “ng serve” command to get the Full stack application of the student survey for our project.
- we should click on http://localhost:4200/ link or navigate in the browser with this link, after successfully running spring boot application. We get our application webpages displayed.

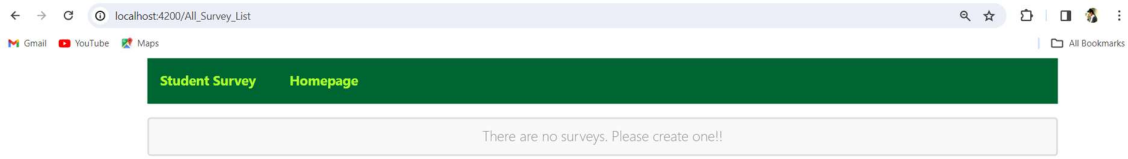
```
ng serve
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\panth\SWE-642-PROEJCT\Project_frontend_Angular>ng serve
Initial chunk files | Names         |  Raw size
polyfills.js        | polyfills     |  83.60 kB |
main.js             | main          |  68.63 kB |
styles.css          | styles        |  95 bytes |

                    | Initial total |  152.33 kB

Application bundle generation complete. [3.267 seconds]

Watch mode enabled. Watching for file changes...
  → Local:   http://localhost:4200/
  → press h + enter to show help
```

Welcome Homepage:

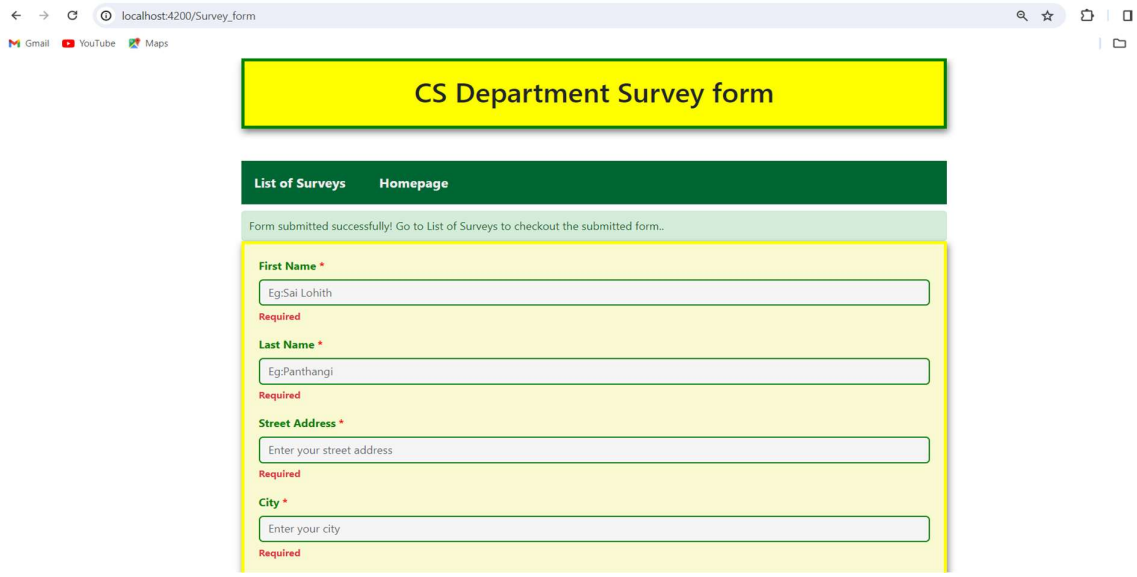


Survey Form:

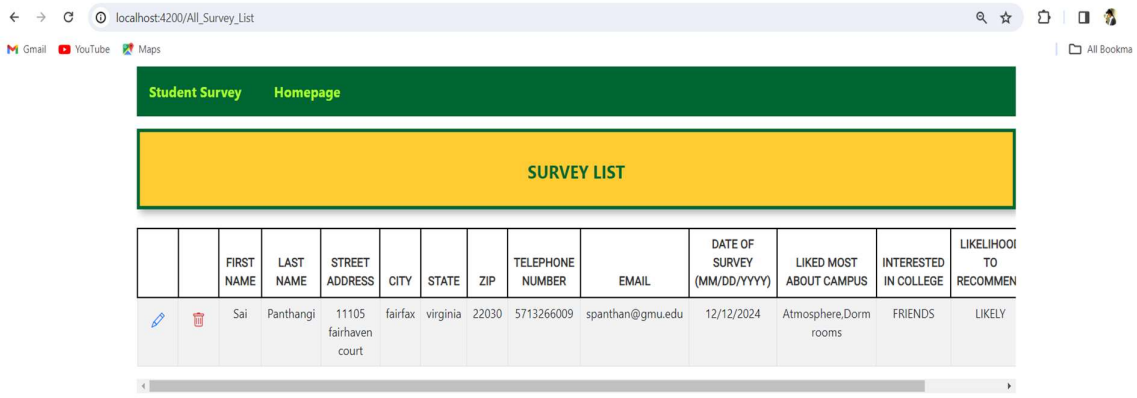
List Of All Survey:



Let's fill out and submit the form and see, whether it's updated in database and reflected in the frontend application as well. Fill the form and click on submit, we then get a success message for form submission if given valid inputs for required fields.

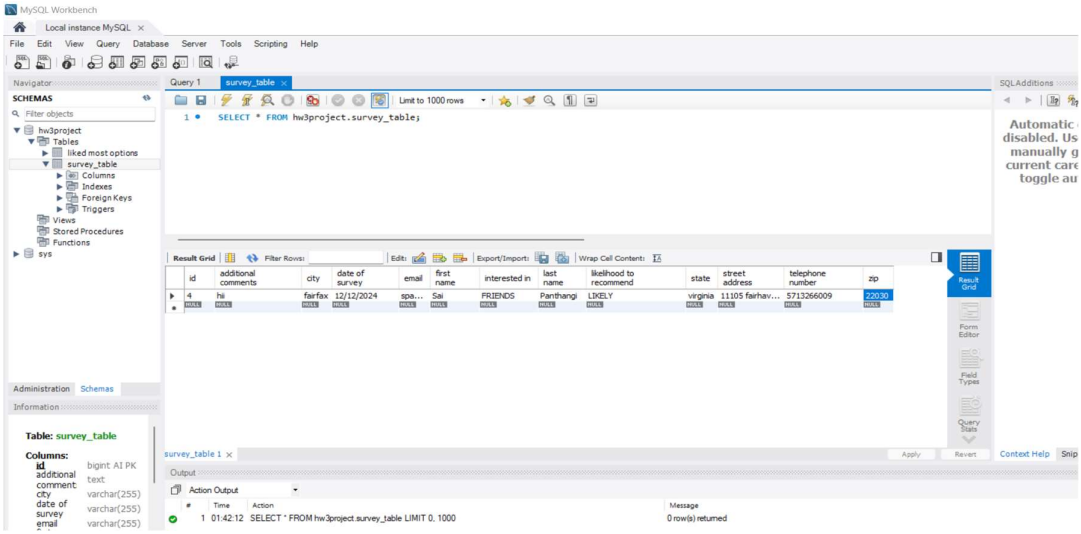


After submitting form, List of All surveys webpage displayed the submitted form details.



		FIRST NAME	LAST NAME	STREET ADDRESS	CITY	STATE	ZIP	TELEPHONE NUMBER	EMAIL	DATE OF SURVEY (MM/DD/YYYY)	LIKED MOST ABOUT CAMPUS	INTERESTED IN COLLEGE	LIKELIHOOI TO RECOMMEN
		Sai	Panthangi	11105 fairhaven court	fairfax	virginia	22030	5713266009	spanthan@gmu.edu	12/12/2024	Atmosphere,Dorm rooms	FRIENDS	LIKELY

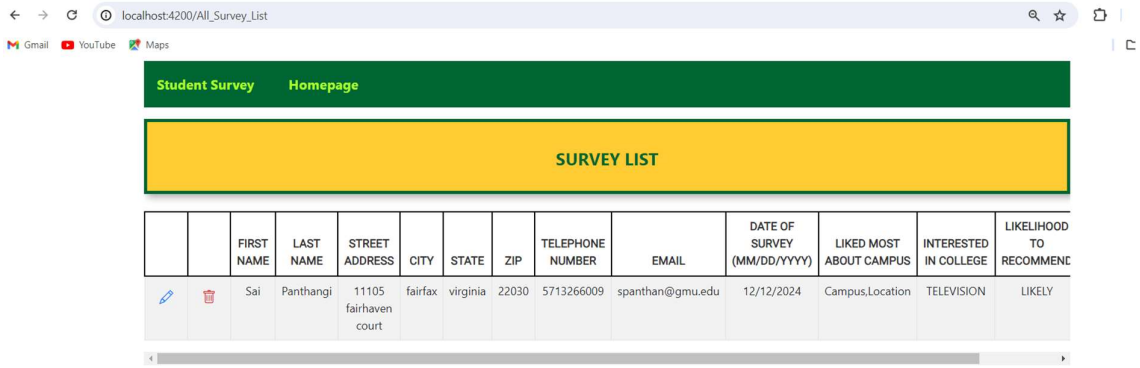
We can see in the MySQL database also, the submitted student form details.



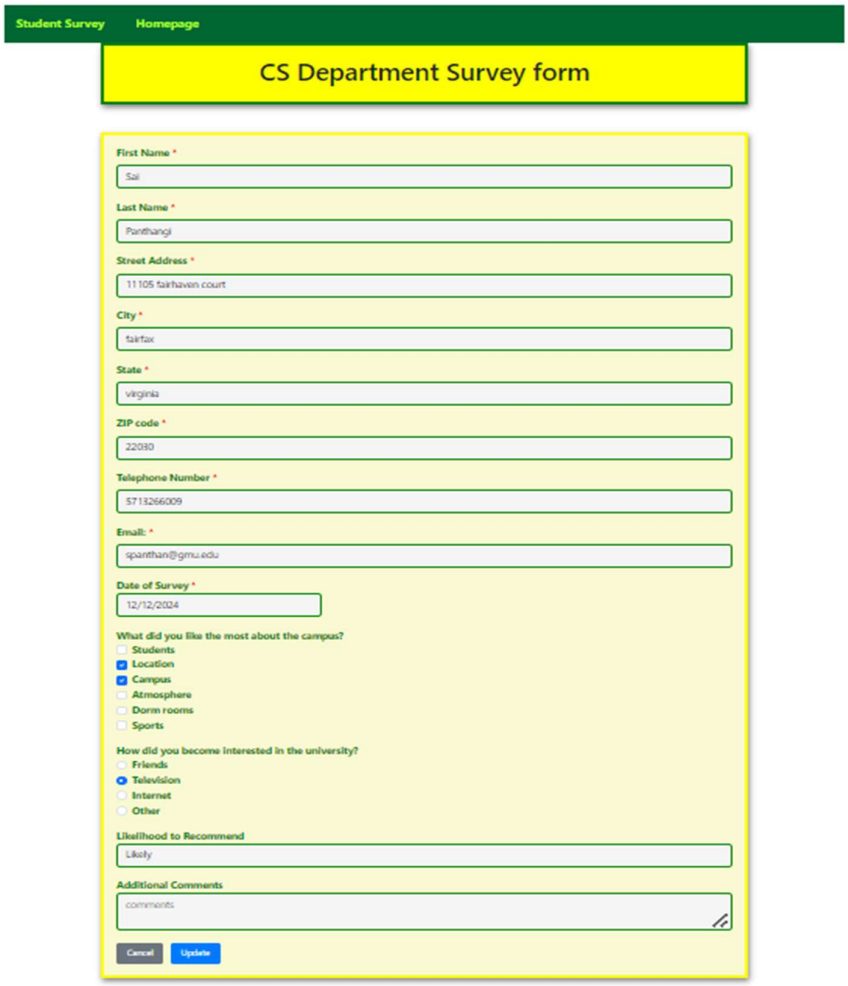
/ Additional features added to the project for user interaction and usage of services in the spring boot application using JPA and restful web services. **/.**

Lets now show the CRUD operations functioning. Firstly, let's check the update feature. I will change the first name content of submitted survey from "sai" to "SWE-PROEJCT".

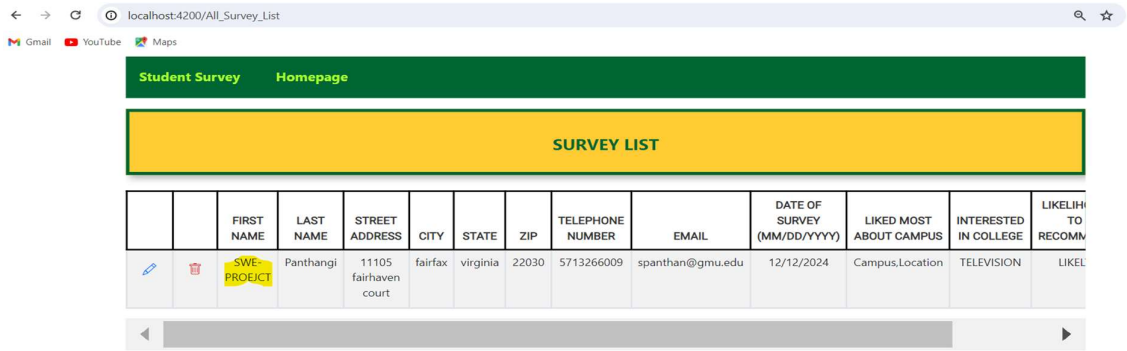
Before Update:



Clicking on update button will retrieve the form:

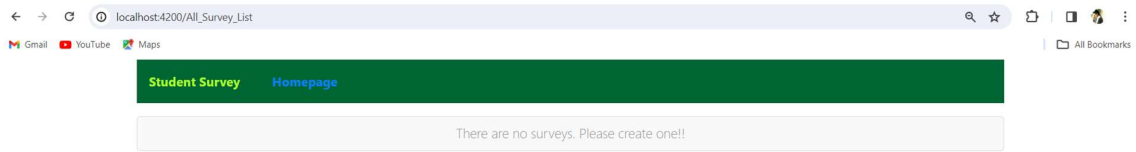


After changing the input field and clicking on Update button will display Updated form:

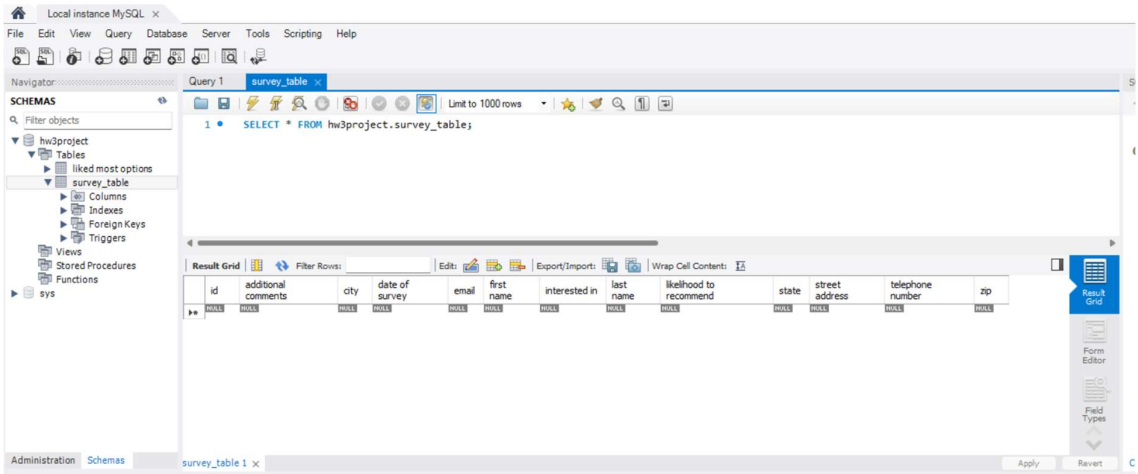


Delete form:

Lets delete the recent submitted form and check this functionality in our full stack application. You can see after clicking on delete button, the survey form is cleared from the list of survey table and as well as database.



Deleted that student survey form from the database as well.



CONCLUSION:

In conclusion, this Project workflow seamlessly integrates Angular for the frontend and Spring Boot for the backend to facilitate user interaction and data management in the student survey application. Users can easily submit their feedback, which is securely stored and accessible for later reference. The application leverages RESTful APIs for efficient communication between the frontend and backend, ensuring a smooth and responsive user experience.