

# 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):

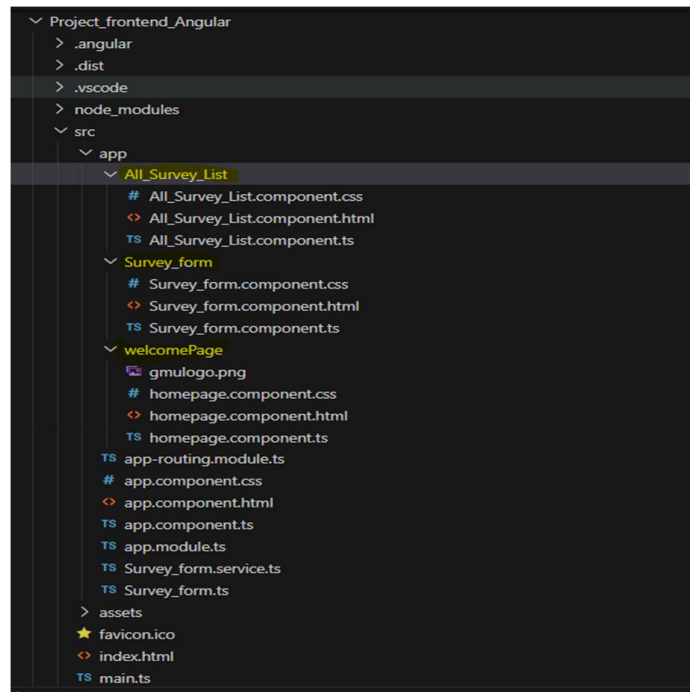
### Angular and node.js Setup:

- 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:

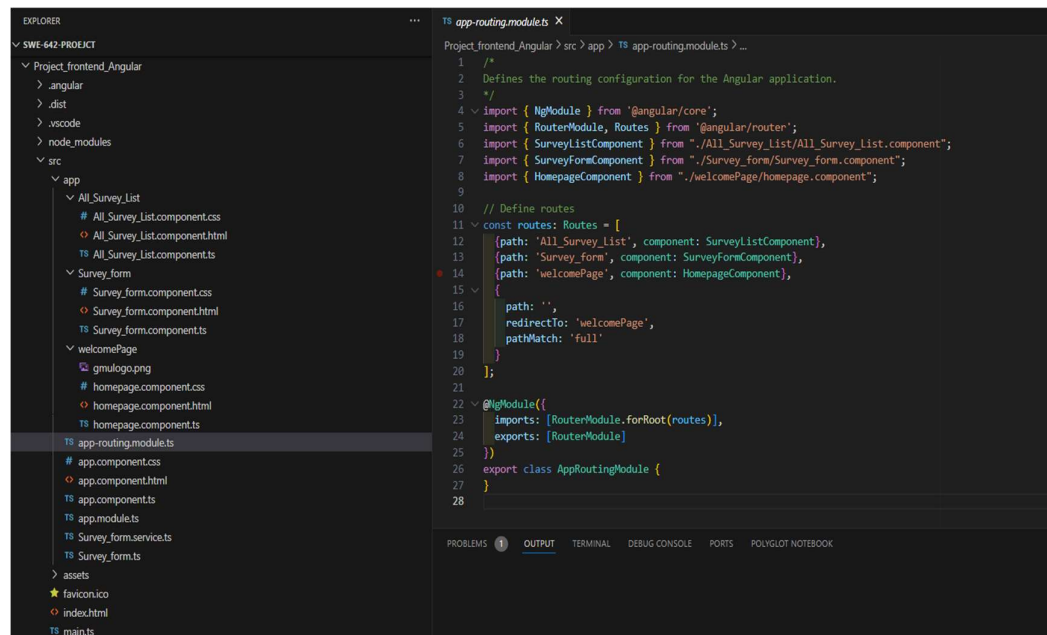
- Generated components required for the application using the **ng generate component** command.
- Components created:
  - Welcome Homepage: '**ng generate component welcomePage**'.
  - Survey Form: '**ng generate component Survey\_form**'.
  - Survey List: '**ng generate component All\_Survey\_List**'.
- 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**.

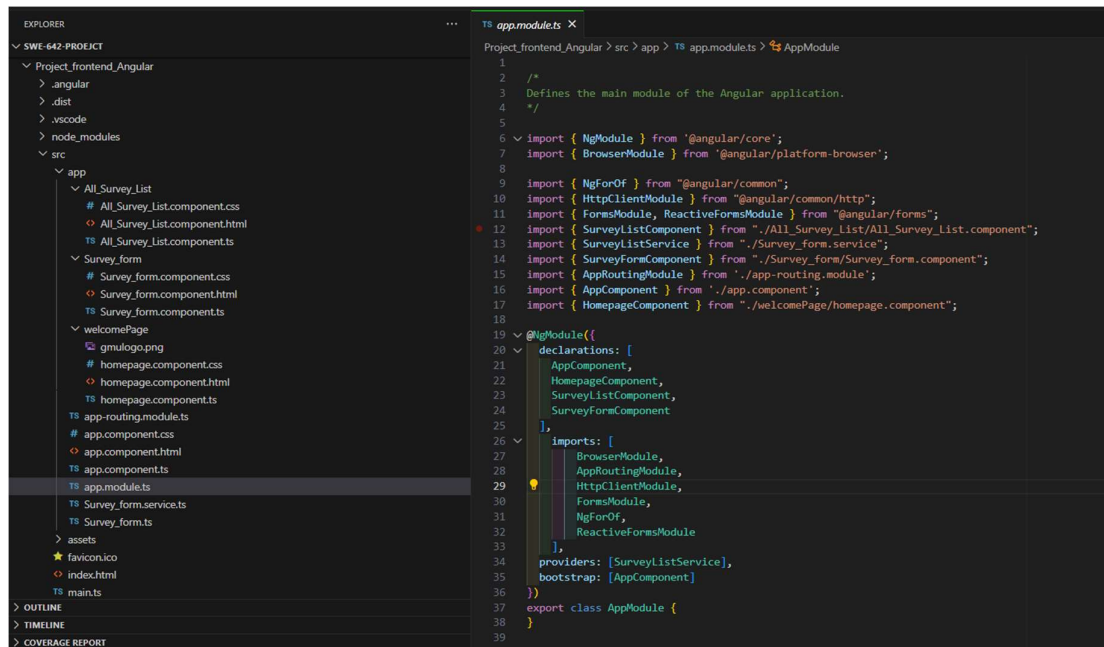
- This created ***app-routing.module.ts***, defining routes specifying which component to load for each path.



```
1 /*
2  * Defines the routing configuration for the Angular application.
3  */
4 import { NgModule } from '@angular/core';
5 import { RouterModule, Routes } from '@angular/router';
6 import { SurveyListComponent } from './All_Survey_List/All_Survey_List.component';
7 import { SurveyFormComponent } from './Survey_form/Survey_form.component';
8 import { HomeComponent } from './welcomePage/homepage.component';
9
10 // Define routes
11 const routes: Routes = [
12   {path: 'All_Survey_List', component: SurveyListComponent},
13   {path: 'Survey_form', component: SurveyFormComponent},
14   {path: 'welcomePage', component: HomeComponent},
15 ];
16
17 // Define routes
18 const routes: Routes = [
19   {
20     path: '',
21     redirectTo: 'welcomePage',
22     pathMatch: 'full'
23   },
24 ];
25
26 @NgModule({
27   imports: [RouterModule.forRoot(routes)],
28   exports: [RouterModule]
29 })
30 export class AppRoutingModule { }
31
32 export class AppRoutingModule { }
```

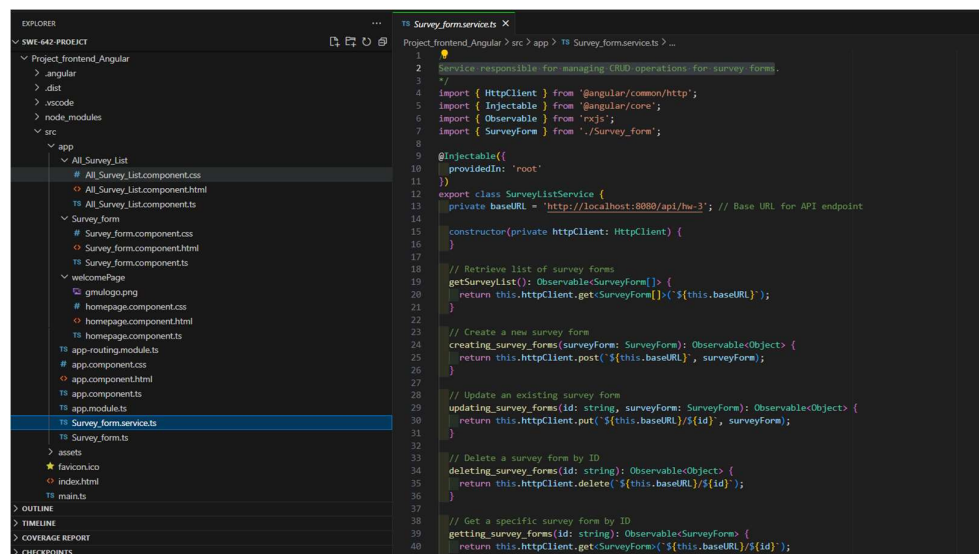
## App Module Configuration:

- 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-PROJECT\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.

The screenshot shows the Spring Initializr web application interface. It includes sections for Project, Language, Spring Boot, Project Metadata, Dependencies, and buttons for GENERATE, EXPLORE, and SHARE.

**Project**

- ☒ Gradle - Groovy
- ☐ Gradle - Kotlin
- ☐ Maven

**Language**

- ☒ Java
- ☐ Kotlin
- ☐ Groovy

**Spring Boot**

- ☐ 3.3.0 (SNAPSHOT)
- ☐ 3.3.0 (M3)
- ☐ 3.2.5 (SNAPSHOT)
- ☒ 3.2.4
- ☐ 3.1.11 (SNAPSHOT)
- ☐ 3.1.10

**Project Metadata**

Group:

Artifact:

Name:

Description:

Package name:

**Dependencies**

**Spring Web** WEB  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

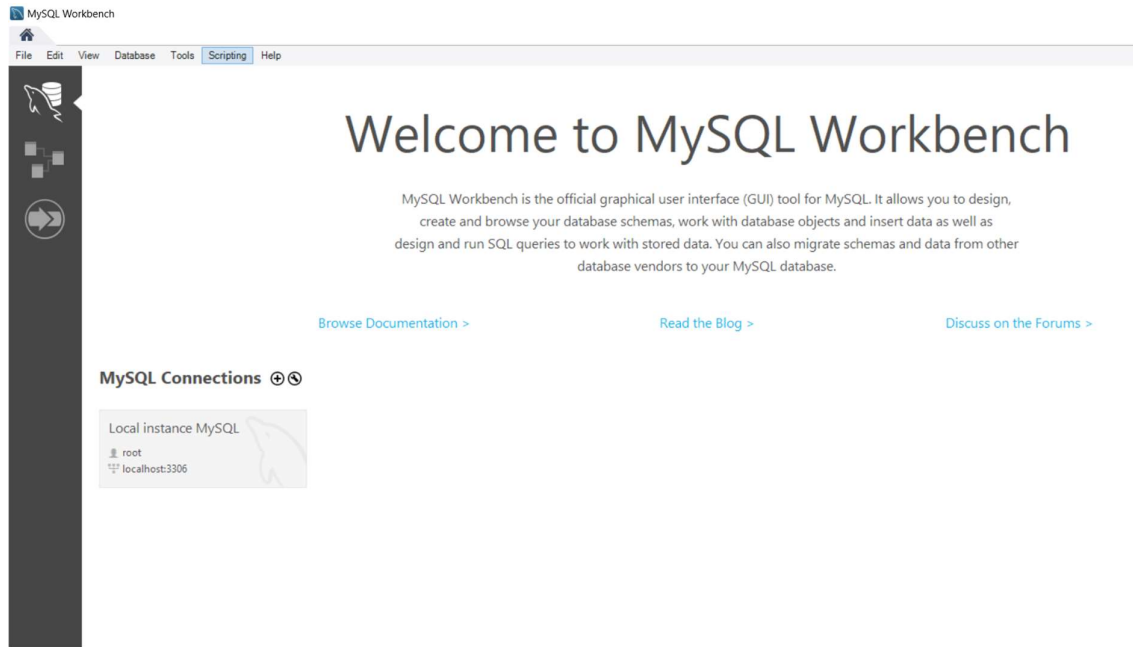
**Spring Data JPA** SQL  
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

**MySQL Driver** SQL  
MySQL JDBC driver.

**Buttons:** GENERATE (CTRL + G), EXPLORE (CTRL + SPACE), SHARE...

### **MySQL Setup:**

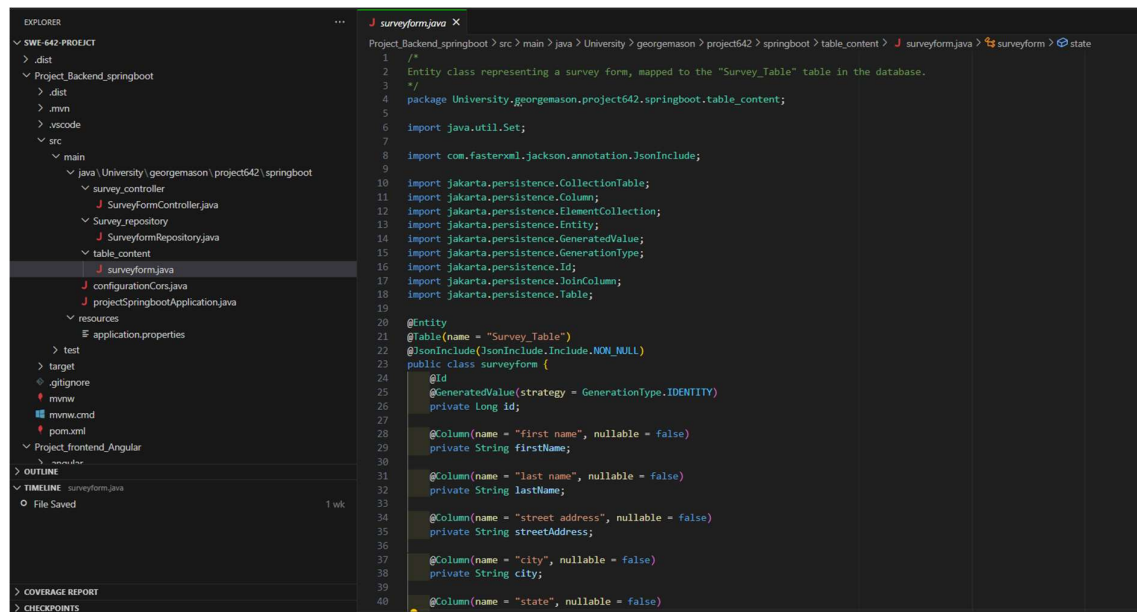
We downloaded and installed MySQL Server and Workbench from the official MySQL website. We configured Workbench to connect to my Local instance MySQL server by providing the hostname, port, and credentials.



## RESTful Endpoints:

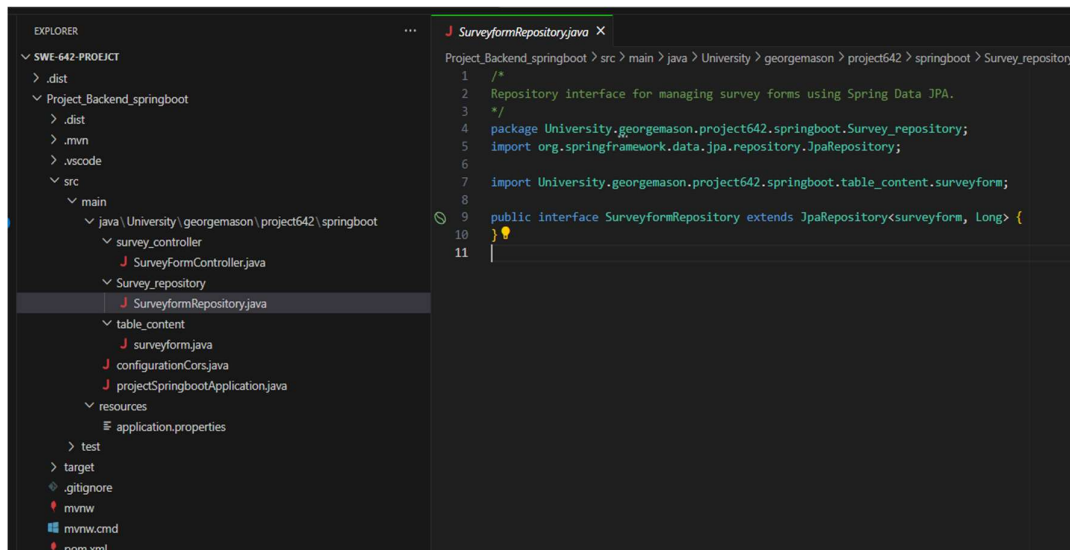
### Survey Entity, Repository, and Controller:

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

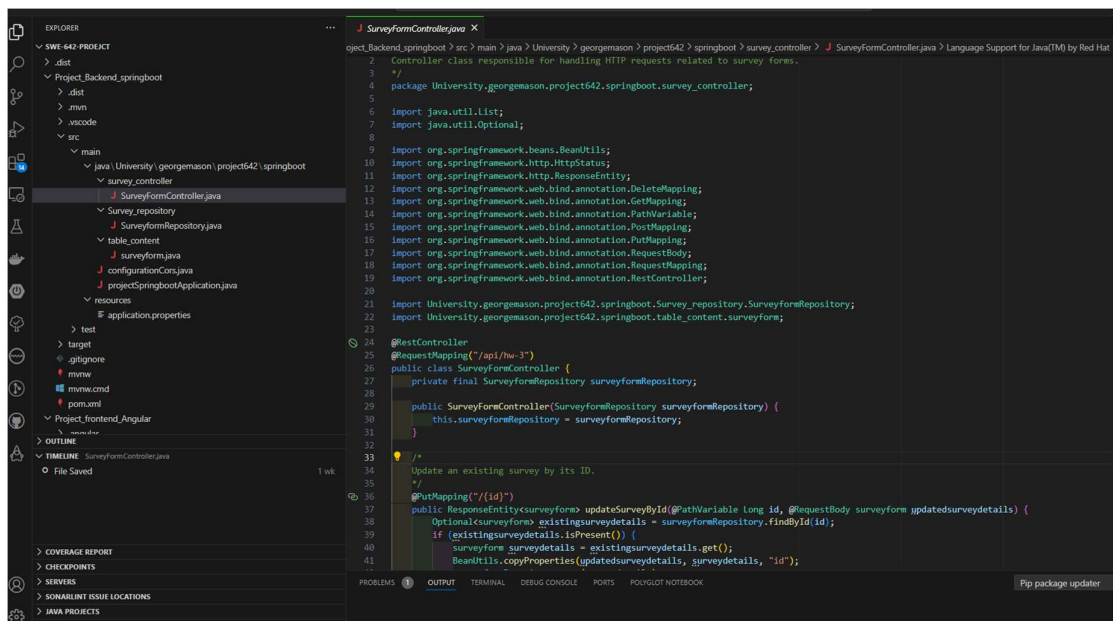


- We developed the `SurveyformRepository` interface extending `JpaRepository<Survey, Long>` to handle database operations.





- 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.

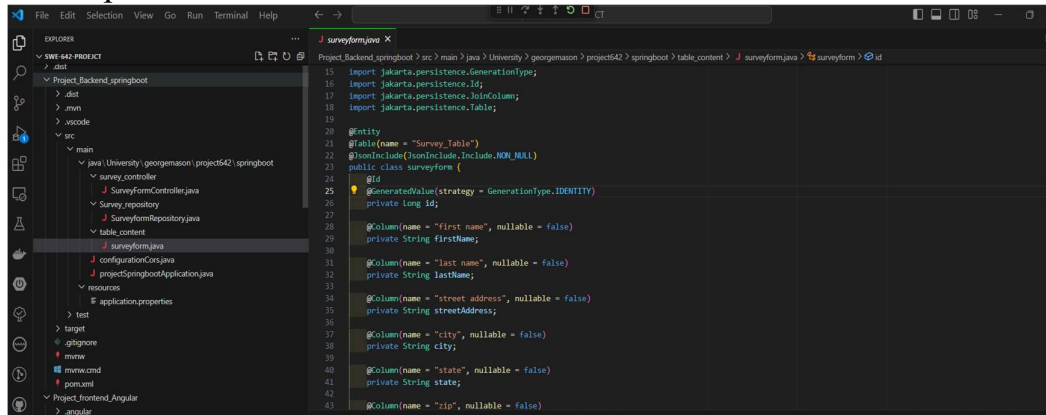


- PUT /api/hw-3/{id}: updating a survey by ID.

## Database Interaction:

### Schema and Tables Creation:

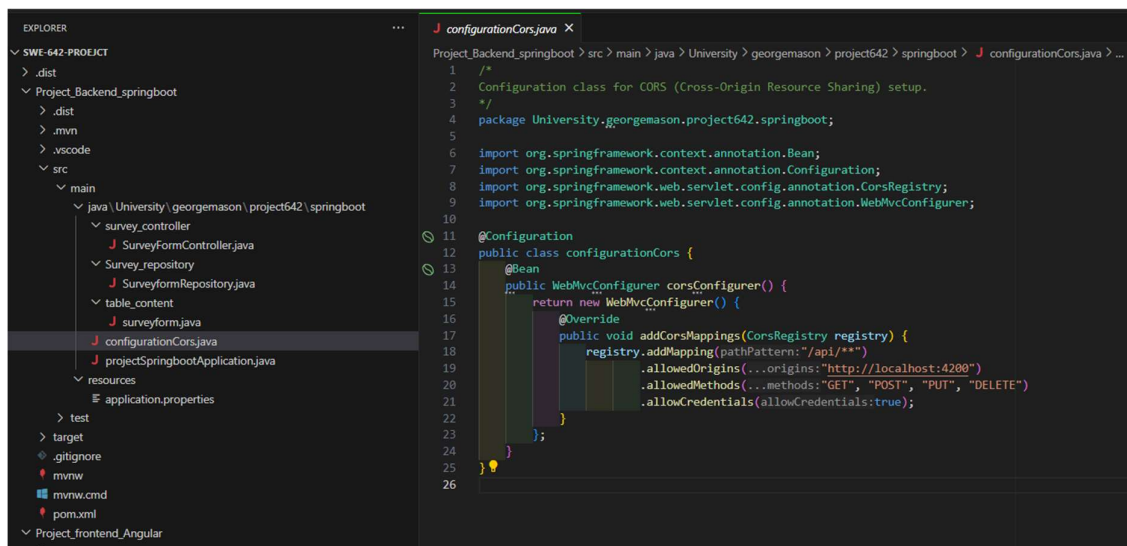
we created 'h3project' schema if not exist, and 'Survey\_Table' table with necessary fields. Also, we created a table 'liked\_most\_options' for handling liked options.



```
15 import jakarta.persistence.GenerationType;
16 import jakarta.persistence.Id;
17 import jakarta.persistence.Table;
18 import jakarta.persistence.Column;
19
20 @Entity
21 @Table(name = "Survey_Table")
22 @TableGenerator(name = "Survey_Table", table = "Survey_Table", type = GenerationType.IDENTITY)
23 public class SurveyForm {
24     @Id
25     @GeneratedValue(strategy = GenerationType.IDENTITY)
26     private long id;
27
28     @Column(name = "first name", nullable = false)
29     private String firstName;
30
31     @Column(name = "last name", nullable = false)
32     private String lastName;
33
34     @Column(name = "street address", nullable = false)
35     private String streetAddress;
36
37     @Column(name = "city", nullable = false)
38     private String city;
39
40     @Column(name = "state", nullable = false)
41     private String state;
42
43     @Column(name = "zip", nullable = false)
```

### CORS configuration:

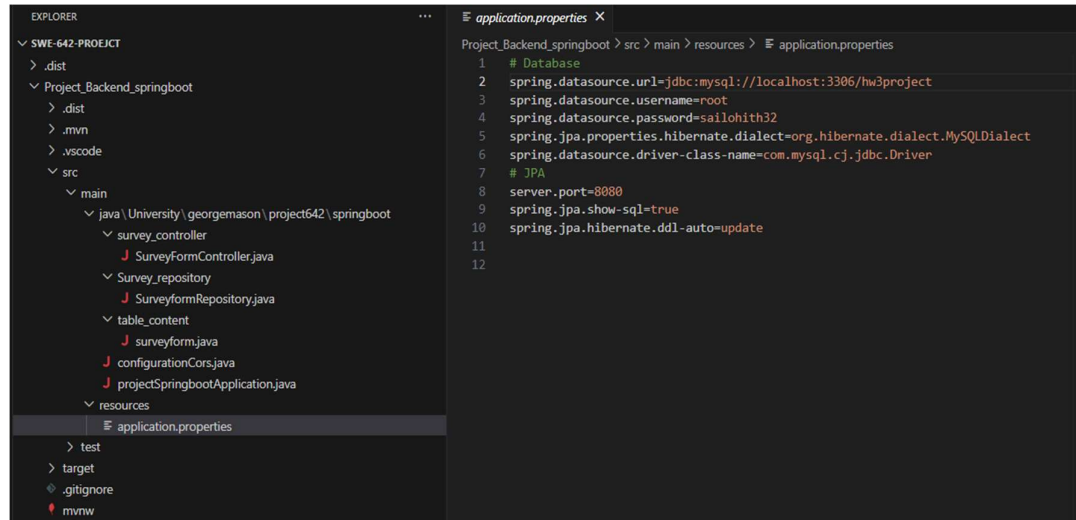
This configuration class sets up CORS (Cross-Origin Resource Sharing) for the Spring Boot application. It allows requests from the specified origin (http://localhost:4200) and permits specified HTTP methods (GET, POST, PUT, DELETE) for requests to paths starting with /api/. Additionally, it allows credentials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



```
1 /*
2  * Configuration class for CORS (Cross-Origin Resource Sharing) setup.
3  */
4 package University.georgemason.project642.springboot;
5
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.web.servlet.config.annotation.CorsRegistry;
9 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
10
11 @Configuration
12 public class ConfigurationCors {
13     @Bean
14     public WebMvcConfigurer corsConfigurer() {
15         return new WebMvcConfigurer() {
16             @Override
17             public void addCorsMappings(CorsRegistry registry) {
18                 registry.addMapping(pathPattern: "/api/**")
19                     .allowedOrigins(...origins: "http://localhost:4200")
20                     .allowedMethods(...methods: "GET", "POST", "PUT", "DELETE")
21                     .allowCredentials(allowCredentials: true);
22             }
23         };
24     }
25 }
26
```

## application.properties Setup:

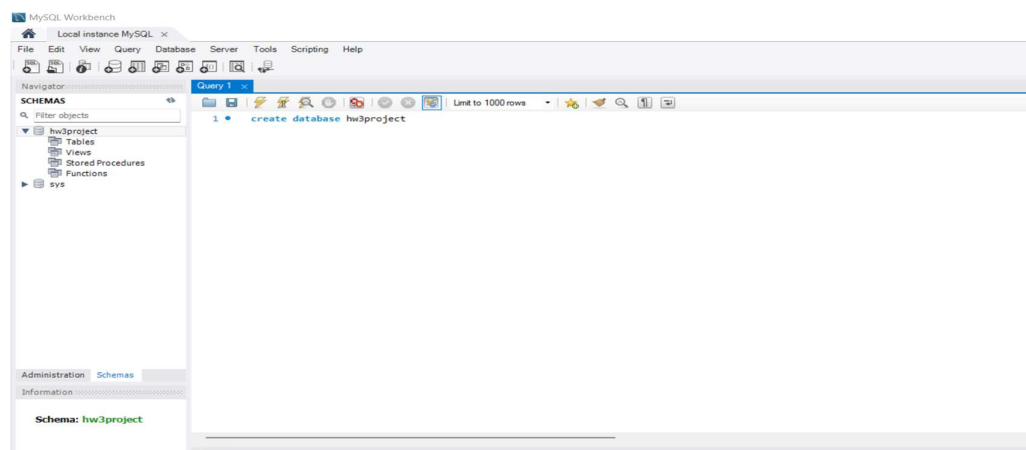
- We updated application.properties in our project's resources directory.
- We configured data source properties to connect to my MySQL instance.
- we configured JPA properties such as dialect and hibernate properties if needed.
- we used spring.jpa.hibernate.ddl-auto=update to generate or update the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



## Usage:

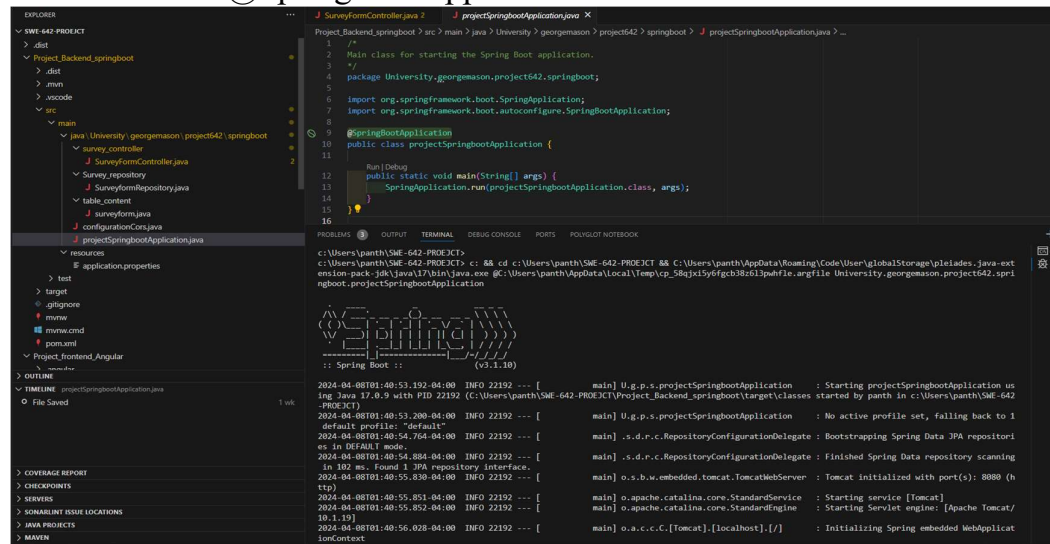
We ensured that the Survey entity class(surveyform.java) is annotated with `@Entity` and SurveyformRepository interface with `@Repository`. We updated the surveyform `@Entity` class to match the database schema.

## Create database in MySQL:

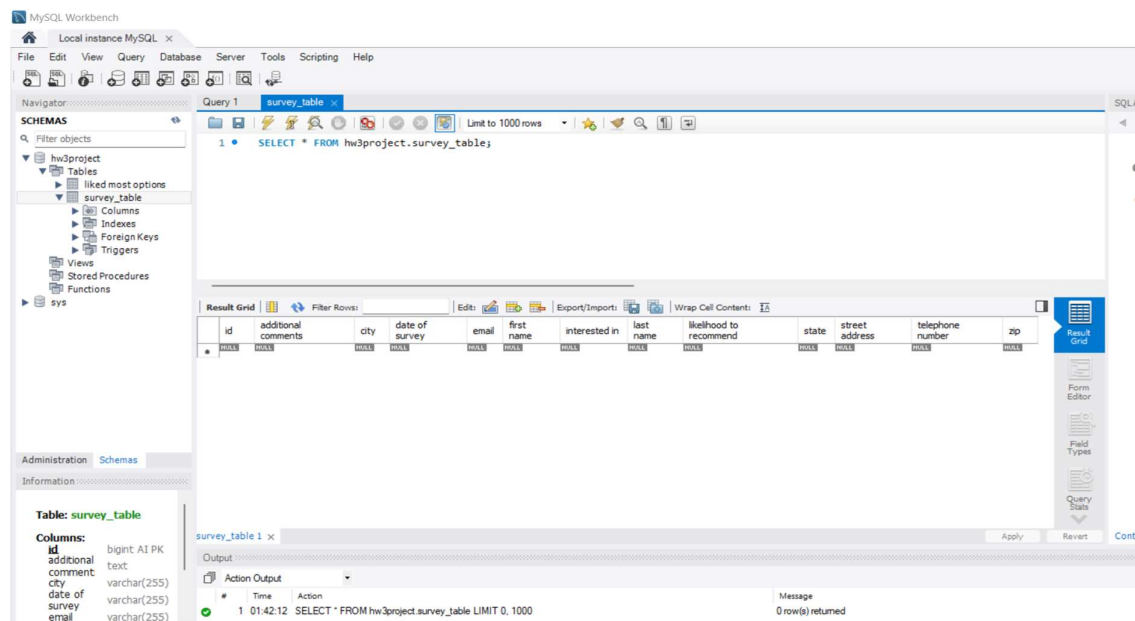


## Running the spring boot application:

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-PROJECT\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:

George Mason University | CS [Student Survey Form](#) [List All Surveys](#)

**COMPUTER SCIENCE DEPARTMENT**

Welcome to the Department of Computer Science at George Mason University!

With nearly 80 full-time faculty members, the CS Department at Mason is the largest department in the College of Engineering and Computing, one of the largest departments on campus, and the largest, fastest growing, and highest-ranked Computer Science department in the Commonwealth of Virginia. Today, not only is computing pervasive in society, but computational methods are used in almost every field of scholarly endeavor ranging from the arts and humanities to the hard sciences and engineering, as well as every sector of industry. Given the importance of computing in today's world, we strongly support the goal of "Computing for All", providing a diverse array of programs and courses in computing to students from all backgrounds and at all levels.

**MS Degrees**

- MS in Computer Science
- MS in Information Systems
- MS in Software Engineering

**Required Courses**

S.No	Course Number	Course Name	Department
1	CS 530	Fundamentals of Engineering Mathematics	Master of Science in Computer Science

Survey Form:

CS Department Survey form

List of SurveysHomepage

First Name \*

Eg Sai Lohith

Required

Last Name \*

Eg Parthangi

Required

Street Address \*

Enter your street address

Required

City \*

Enter your city

Required

State \*

Enter your state

Required

ZIP code \*

eg: 12345

Required

Telephone Number \*

Format: (xxx) xxx-xxxx

Required

Email: \*

gmu@gmail.com

Required

Date of Survey \*

MM/DD/YYYY

Required

What did you like the most about the campus?

☐ Students

☐ Location

☐ Campus

☐ Atmosphere

☐ Dorm rooms

☐ Sports

How did you become interested in the university?

☐ Friends

☐ Television

☐ Internet

☐ Other

Likelihood to Recommend

Select...

Additional Comments

comments

Cancel

Submit

List Of All Survey:

localhost:4200/All\_Survey\_List

GmailYouTubeMaps

Student SurveyHomepage

There are no surveys. Please create one!!

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.

localhost:4200/Survey\_form

Gmail YouTube Maps

### CS Department Survey form

List of Surveys Homepage

Form submitted successfully! Go to List of Surveys to checkout the submitted form..

**First Name \***  
Eg:Sai Lohith  
Required

**Last Name \***  
Eg:Panthangi  
Required

**Street Address \***  
Enter your street address  
Required

**City \***  
Enter your city  
Required



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

localhost:4200/All\_Survey\_List

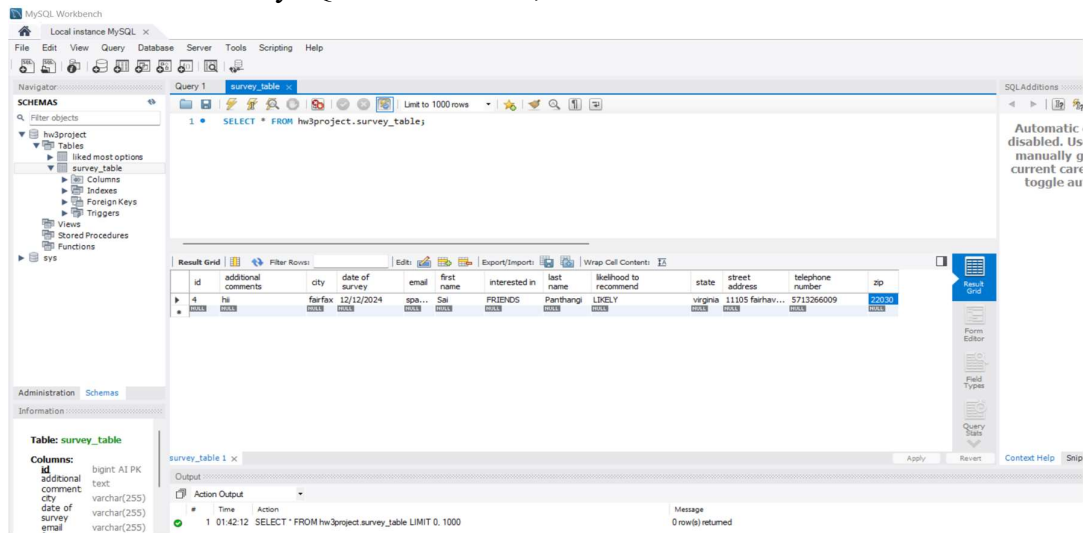
Gmail YouTube Maps All Bookma

Student Survey Homepage

### SURVEY LIST

		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	LIKELIHOOD TO RECOMMEN
		Sai	Panthangi	11105 fairhaven court	fairfax	virginia	22030	5713266009	spanthan@gmu.edu	12/12/2024	Atmosphere,Dorm rooms	FRIENDS	LIKELY

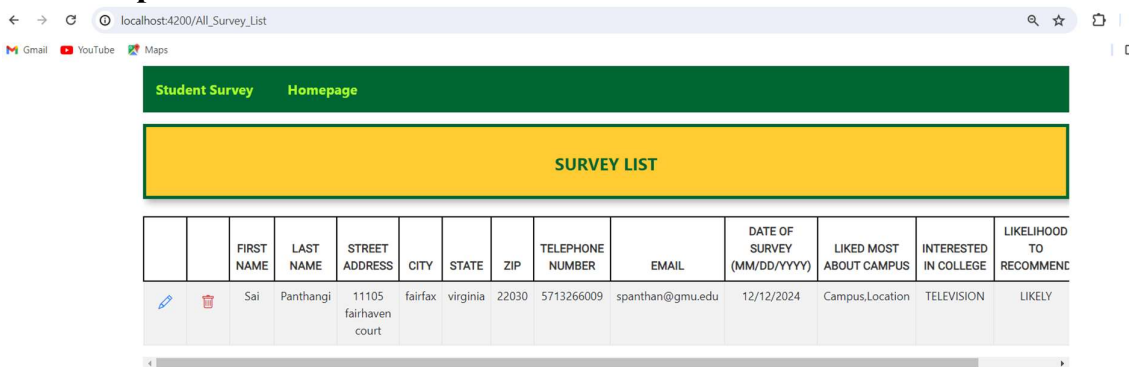
We can see in the MySQL database also, the submitted student form details.



**/\*\* 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:

Student Survey

Homepage

CS Department Survey form

First Name \*

Sai

Last Name \*

Panthangi

Street Address \*

11105 fairhaven court

City \*

fairfax

State \*

virginia

ZIP code \*

22030

Telephone Number \*

5713266009

Email: \*

spanthan@gmu.edu

Date of Survey \*

12/12/2024

What did you like the most about the campus?

☐ Students

☒ Location

☒ Campus

☐ Atmosphere

☐ Dorm rooms

☐ Sports

How did you become interested in the university?

☐ Friends

☒ Television

☐ Internet

☐ Other

Likelihood to Recommend

Likely

Additional Comments

comments

Cancel

Update

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

← → ↺

localhost:4200/All\_Survey\_List

🔍 ☆

Student Survey

Homepage

SURVEY LIST

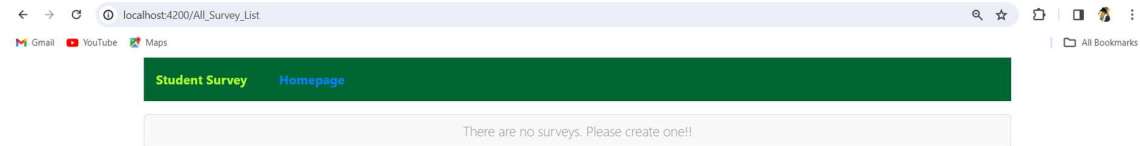
		FIRST NAME	LAST NAME	STREET ADDRESS	CITY	STATE	ZIP	TELEPHONE NUMBER	EMAIL	DATE OF SURVEY (MM/DD/YYYY)	LIKED MOST ABOUT CAMPUS	INTERESTED IN COLLEGE	LIKELIHOOD TO RECOMMEND
			Panthangi	11105 fairhaven court	fairfax	virginia	22030	5713266009	spanthan@gmu.edu	12/12/2024	Campus,Location	TELEVISION	LIKEL

◀

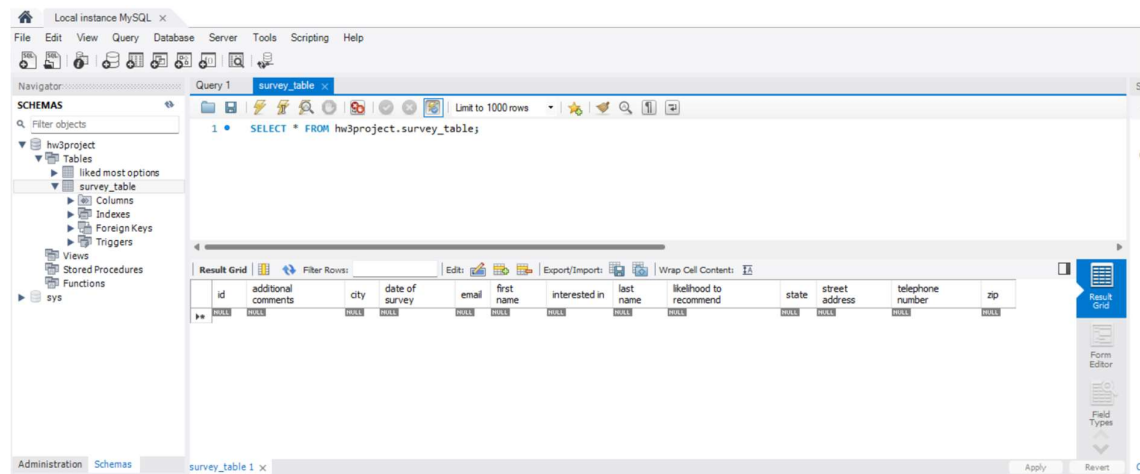
▶

## 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.