SWE Assignment 3
Survey Form Web Application using Angular 15 and Springboot

Angular CLI :FrontEnd

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

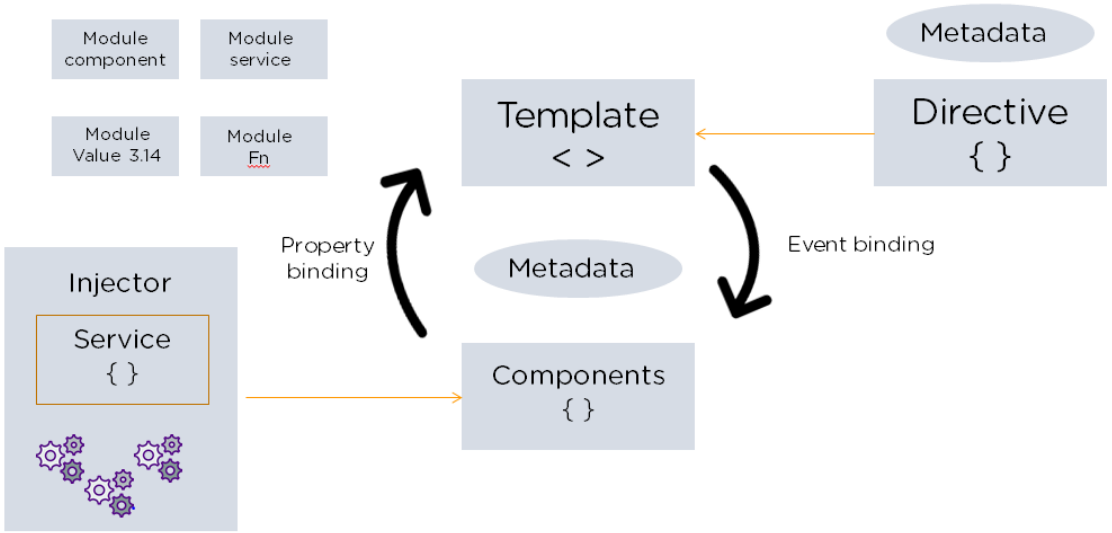
- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

Commands used:

-To install node.js : Visit the Node.js official site https://nodejs.org/en/download/ ,download and install the latest version.
-To install Angular CLI :- npm install -g @angular/cli
-To create a new angular project:- ng new project_name
cd project_name
ng serve
-To install Bootstrap:- npm install bootstrap –save
-To create a new component:- ng generate component component_name OR
ng g c component_name
-To create a new service:- ng generate service service_name OR
ng g s service_name

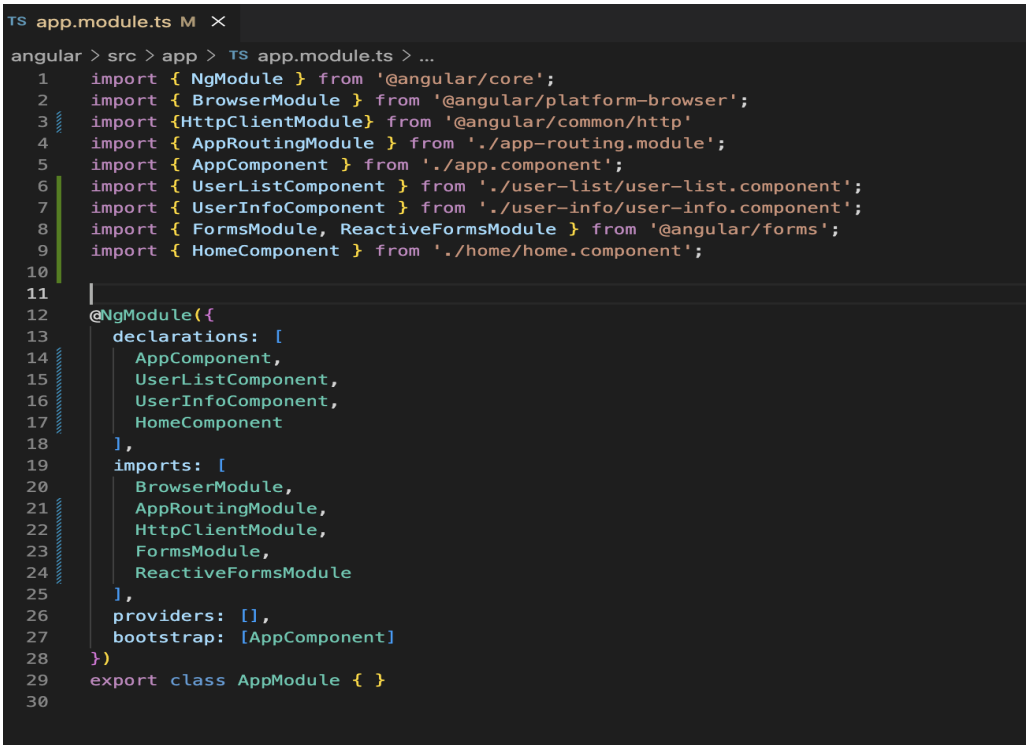
Angular Architecture

Angular is a full-fledged model-view-controller (MVC) framework. It provides clear guidance on how the application should be structured and offers bi-directional data flow while providing real DOM.



1. Modules

An Angular app has a root module, named AppModule, which provides the bootstrap mechanism to launch the application.



2. Components

Each component in the application defines a class that holds the application logic and data. A component generally defines a part of the user interface (UI).

Our application has 4 components:
-AppComponent ,
-UserListComponent:

- ◆ It shows the survey page where the user survey form is displayed. The user enters the details namely: name, address, email, etc.
- ◆ Once the user enters the form details and presses the submit button onSubmit() method is called. The method called HTTP post is made to the backend with user data.
- ◆ A registered service is created to make an HTTP client call.

-UserInfoComponent:

- ◆ The show user list all the users that have registered through the survey page. It makes an HTTP get a call to the backend to get all the user details and show them in show user Html.
- ◆ The user class is created to map the data to the class and display it in UI.

-HomeComponent:

- ◆.Used for the Home page.

3. Templates

The Angular template combines the Angular markup with HTML to modify HTML elements before they are displayed. There are two types of data binding:

Event binding: Lets your app respond to user input in the target environment by updating your application data.
Property binding: Enables users to interpolate values that are computed from your application data into the HTML.

4. Metadata
Metadata tells Angular how to process a class. It is used to decorate the class so that it can configure the expected behavior of a class.

5. Services
When you have data or logic that isn't associated with the view but has to be shared across components, a service class is created. The class is always associated with the @Injectible decorator.

6. Dependency Injection
This feature lets you keep your component classes crisp and efficient. It does not fetch data from a server, validate the user input, or log directly to the console. Instead, it delegates such tasks to the services.

- Our project's index page has an app-root component. The HTML for the app component has two links: Survey-Form and Users. For page navigation angular router link is used, this takes you to the page. When a new component is instantiated, the tag router-outlet> broadcasts an activate event, and when a component is destroyed, it emits a deactivate event.

index.html M

angular > src > index.html > ...

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>SWE645</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

app.component.html M

angular > src > app > app.component.html > h1.text-center

```
<nav class="navbar navbar-custom navbar-expand-sm navbar-dark">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a routerLink="home" routerLinkActive="active" class="navbar-brand">Home</a>

    </li>

    <li class="nav-item">
      <a routerLink="Survey-Form" routerLinkActive="active" class="navbar-brand">Survey Form</a>

    </li>

    <li class="nav-item">
      <a routerLink="users" routerLinkActive="active" class="navbar-brand">Users List</a>

    </li>
  </ul>

</nav>

<h1 class="text-center">{{title}}</h1>
<div class="container">
  <router-outlet></router-outlet>
</div>
```

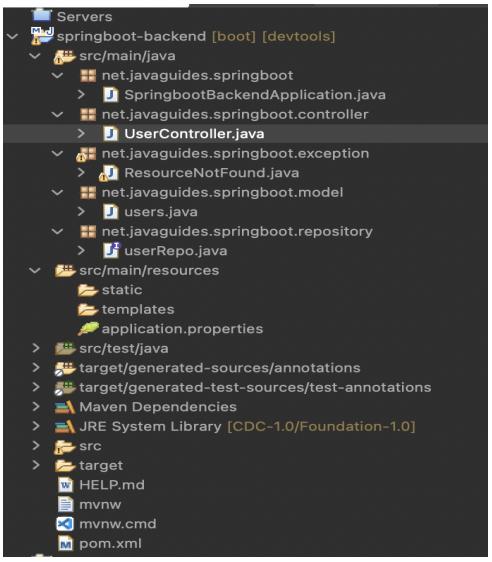
SpringBoot-BackEnd

Our backend application is implemented using Java SpringBoot. Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications.

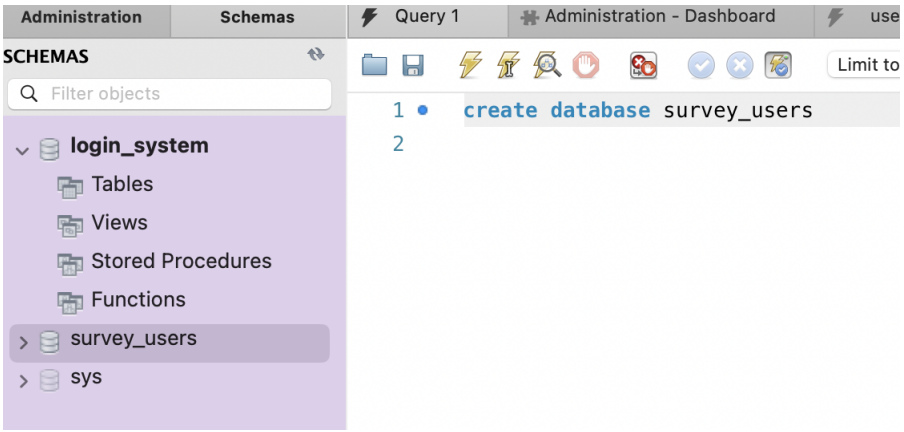
-To create a springBoot Application on eclipse, we need to download Spring Tool Suite (STS) on eclipse. STS requires Latest version of Java to be downloaded.
- Once the application is installed, we create a new springboot project and install all the dependencies. Our project uses 4 dependencies:

1. **Springdata jpa dependency :** to develop RESTful web services**.**
2. **MySQL Driver**
3. **SpringBoot Devtools dependency**

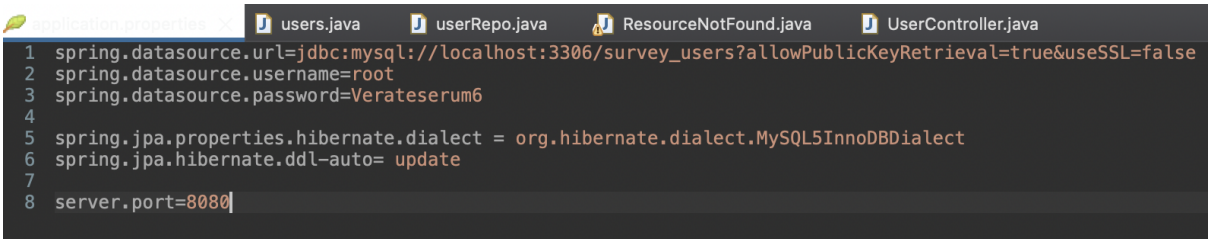
-Springboot application file structure:



- After configuring all the java files we create our database survey_users in MySQL Workbench.



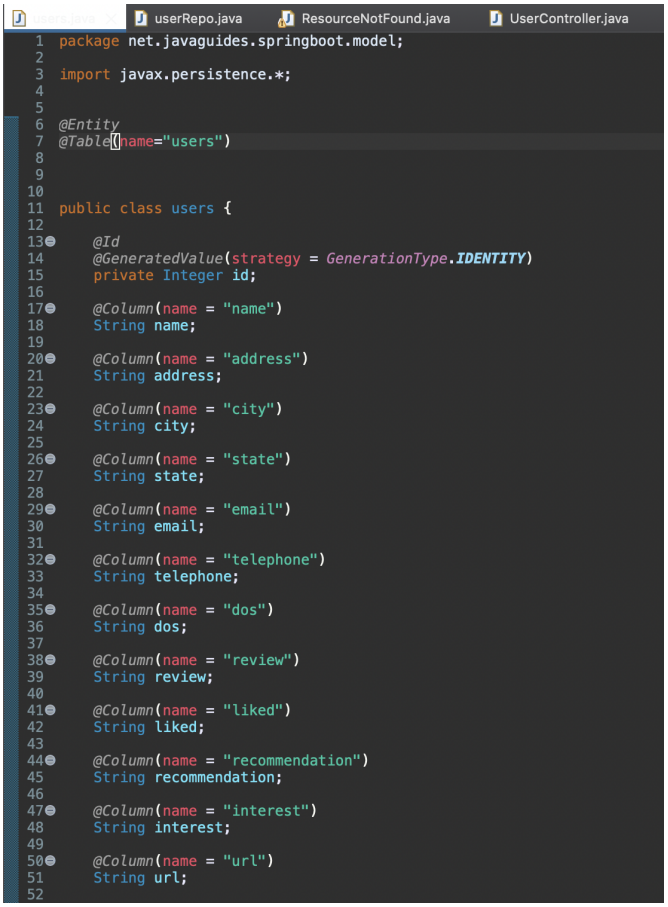
-To configure database files, we update application properties under src/main/resources:



We also configure hibernate properties . spring.jpa.hibernate.ddl-auto=update automatically creates tables in our database.

-To create a JPA Entity & Repository:

- To create a JPA entity, we create a users.java that defines the instance variables. To map this model to relational database table we use JPA annotations: @Entity and @Table(name=”users”)
- To define primary key for table we use @Id , @GeneratedValue(strategy=GenerationType.Identity) and @Column annotations.



- Our Database gets updated with these column names:

- To create a Repository interface, we created a UserRepo.java class and annotated with @Repository

-To Create RESTful Web services:

- Creating an exception class ResourceNotFond.java and annotating it with @ResponseStatus(value=”httpStatus.NOT_FOUND”).This api returns Not Found status to the client.
- Creating controller class UserController.java. The controller is where we create the REST api’s. Thus we annotate this class with @RestController and @RequestMapping(“/api/v1/”). We then go ahead and inject our Repository UserRepo.java here using @Autowired.
- We develop our get users REST api that returns a list of all users and annotate it with @getMapping(“/users”).

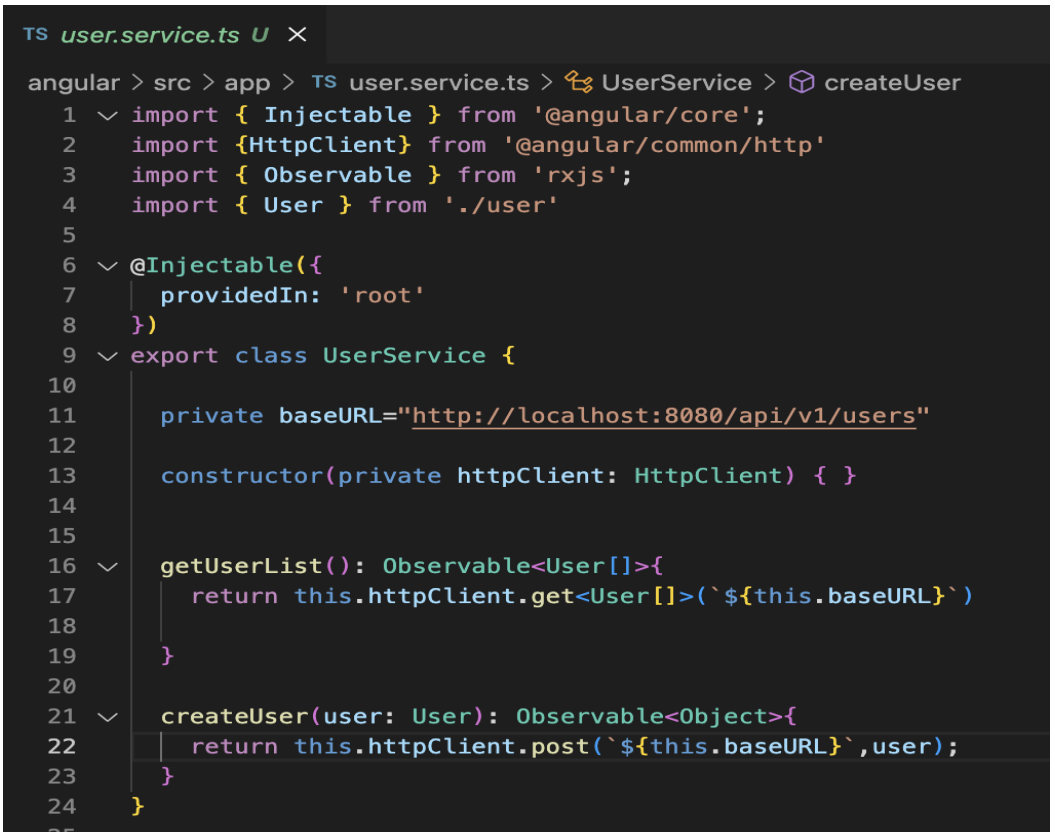
```
package net.javaguides.springboot.controller;

import java.util.*;


@RestController
@RequestMapping("/api/v1/")
@CrossOrigin(origins = "http://localhost:4200")
public class UserController {

    @Autowired
    private userRepo userRepository;


    //get all users

    @GetMapping("/users")
    public List<users> getALlUsers(){

        return userRepository.findAll();
    }
```

- Now if he type the url: localhost:8080/api/v1/users then this api gets called.It returns a list of users to the clients

- We then create a CreteUser RESTapi:.

```
    //create user rest api
    @PostMapping("/users")
    public users createUser(@RequestBody users user) {
        return userRepository.save(user);
    }
}
```

-We create a service in angular called User.service that creates getUserList and createUser which are of the type Observable.



-Our User-list component subscribes to the getUsers observable.

```
import { Component } from '@angular/core';
import {User } from '../user'
import { UserService } from '../user.service';

@Component({
  selector: 'app-user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent {
  users: User[];
  constructor (private userService: UserService) {}

  ngOnInit(): void{
    this.getUsers();


  }

  private getUsers(){
    this.userService.getUserList().subscribe(data =>{this.users=data;});
  }

}
```

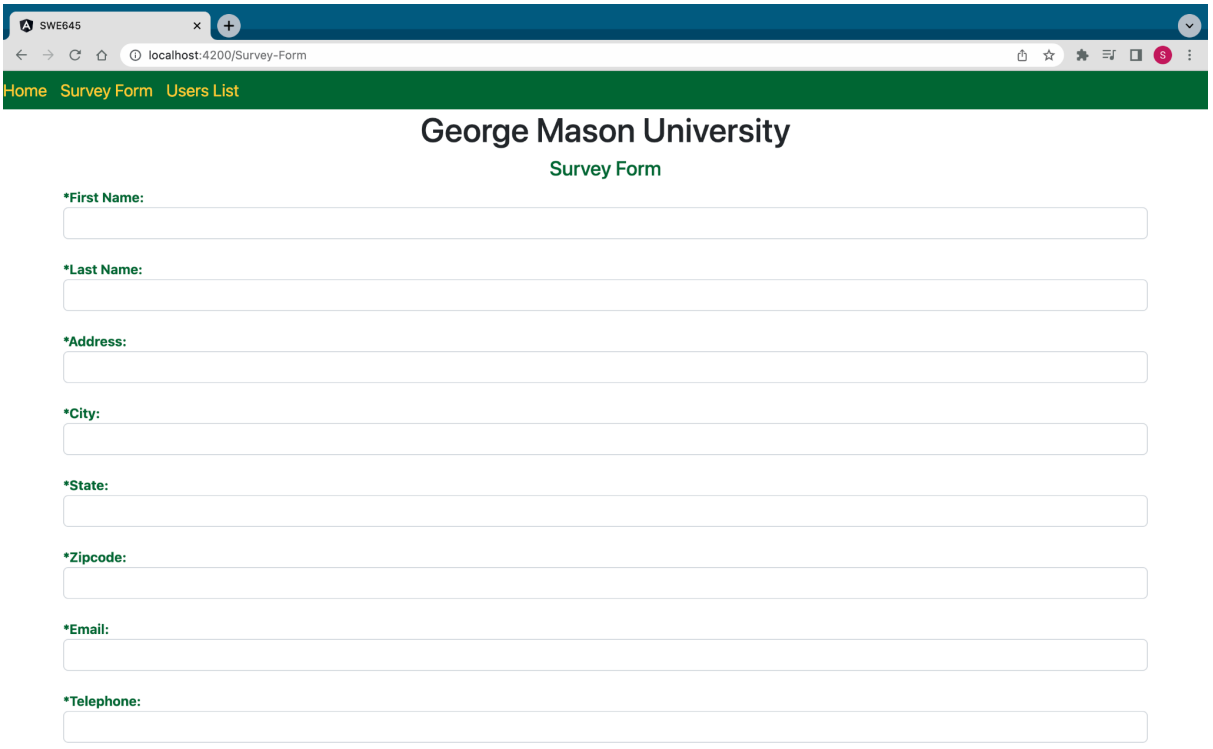
-And userinfo Component subscribes to the create user observable

```
  saveUser(){
    this.userService.createUser(this.user).subscribe(data=>{
      console.log(data);

      this.goToUserList();
    },
    error=>console.log(error));
  }

  goToUserList(){
    this.router.navigate(['/users']);

  }
```

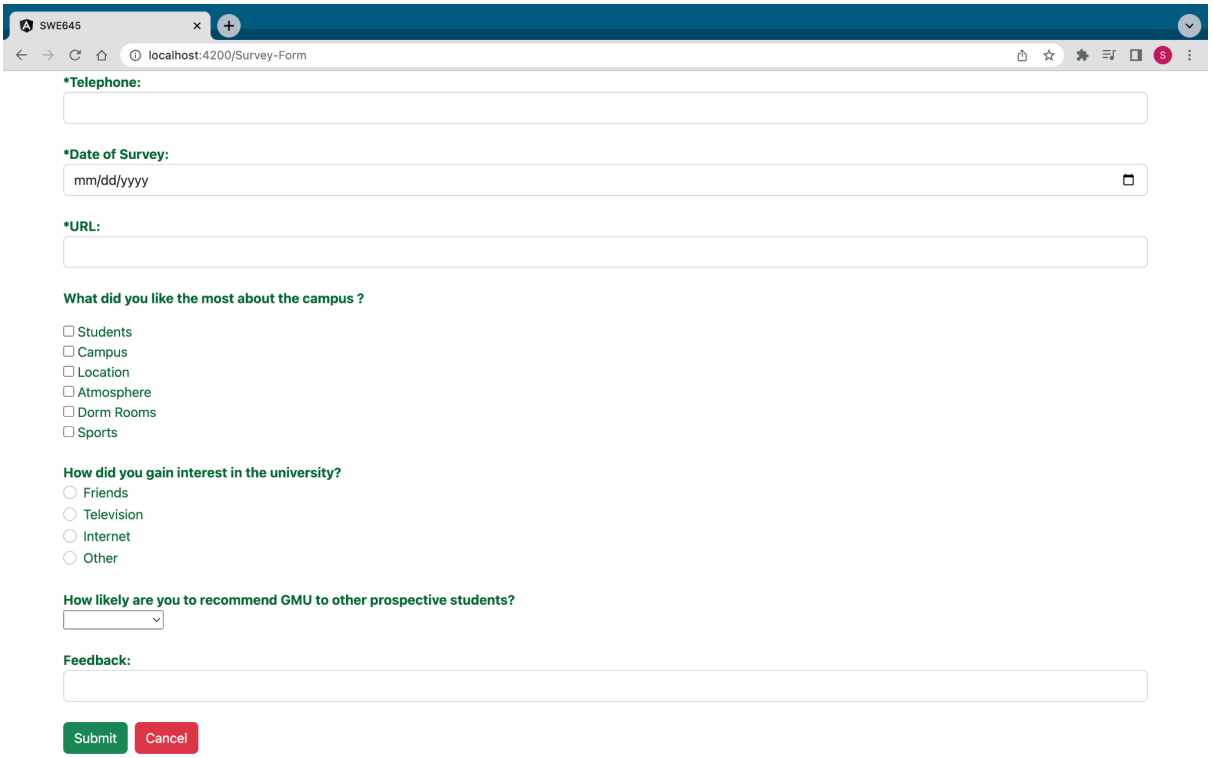
RESULTS

HOME PAGE:

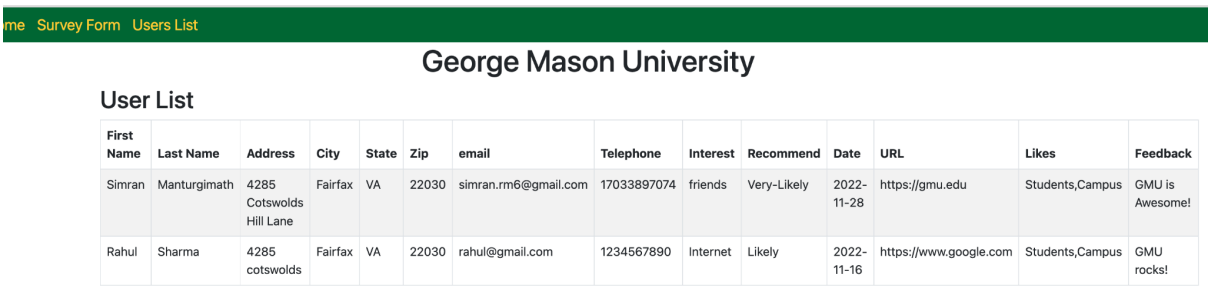
SURVEY FORM:



SURVEY FORM:



USER LIST:



First Name	Last Name	Address	City	State	Zip	email	Telephone	Interest	Recommend	Date	URL	Likes	Feedback
Simran	Manturgimath	4285 Cotswolds Hill Lane	Fairfax	VA	22030	simran.rm6@gmail.com	17033897074	friends	Very-Likely	2022-11-28	https://gmu.edu	Students,Campus	GMU is Awesome!
Rahul	Sharma	4285 cotswolds	Fairfax	VA	22030	rahul@gmail.com	1234567890	Internet	Likely	2022-11-16	https://www.google.com	Students,Campus	GMU rocks!

TEAM MEMBERS:

Team Members	G-Number	Work-Distribution
Simran Manturgimath	G01251959	Angular + Video
Rahul Sharma	G01329939	Angular + SpringBoot
Bharath Shivaprasanna	G01337117	SpringBoot + Report