Authors :

Sai Krisha Chaitanya Srirangam - G01401635
Sundara Madhav Nemani - G01401630
Pranitha Kakumanu - G01379534

BACKEND DOCUMENTATION

Our backend application is built using Spring Boot, which enables us to implement REST endpoints for communication with the frontend Angular application. To store and retrieve data, we utilize Spring Data in conjunction with the H2 database. H2 is an open-source RDBMS written in Java, designed to be lightweight, fast, and seamlessly embeddable within Java applications. It supports SQL and provides a JDBC API for convenient database interactions. With Spring Data, integrating and communicating with H2 is straightforward and effortless.

REST Endpoints

POST /api/survey-form

Request Body: Survey form data
Response Body: "Form submitted successfully"
Description: Submits a survey form.

GET /api/survey-form

Response Body: List of survey forms
Description: Retrieves all survey forms.

```java
@CrossOrigin
@PostMapping("/survey-form")
public ResponseEntity<String> submitForm(@RequestBody SurveyForm form) {
    surveyRepository.save(form);
    return ResponseEntity.ok("Form submitted successfully");
}
@CrossOrigin
@GetMapping("/survey-form")
public ResponseEntity<List<SurveyForm>> getAllSurveyForms() {
    Iterable<SurveyForm> all = surveyRepository.findAll();
    List<SurveyForm> collect = StreamSupport.stream(all.spliterator(), parallel: false)
            .collect(Collectors.toList());
    return ResponseEntity.ok(collect);
}
```

Classes Description

SurveyController -

The SurveyController class is a Spring MVC controller that handles HTTP requests related to survey forms. It provides endpoints for submitting survey forms and retrieving all survey forms.

1. submitForm Method

This method handles the HTTP POST request for submitting a survey form. Saves the survey form to the repository using the surveyRepository.save(form) method. Returns a response with HTTP status code 200 (OK) and the message "Form submitted successfully".

2. getAllSurveyForms Method

This method handles the HTTP GET request for retrieving all survey forms. Retrieves all survey forms from the repository using the surveyRepository.findAll() method. Returns a response with HTTP status code 200 (OK) and the list of survey forms.

SurveyRepository Interface -

The SurveyRepository interface extends the CrudRepository interface provided by Spring Data, which already defines several methods for basic CRUD operations.

@Repository: Indicates that this interface is a repository component, allowing it to be automatically detected and instantiated by Spring Data.

```java
import com.student.survey.entities.SurveyForm;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface SurveyRepository extends CrudRepository<SurveyForm, Long> {
}
```

SurveyForm Class -

The SurveyForm class represents the entity that is persisted in the database. The SurveyForm class represents a survey form entity with various properties. It is annotated with @Entity, indicating that it is a persistent entity in the database. The @Id and @GeneratedValue annotations define the primary key identifier for the entity.

```java
@Data
@Entity
public class SurveyForm {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    private String streetAddress;
    private String city;
    private String state;
    private String zip;
    private String telephone;
    private String email;
    private String surveyDate;
    @ElementCollection
    private List<String> campusLikes;
    private String interestedIn;
    private String recommendation;
    private String comments;
}
```

SurveyApplication Class -

The SurveyApplication class is the entry point of the Spring Boot application for the Student Survey system. It serves as the starting point for the application's execution.

FRONTEND DOCUMENTATION

The Frontend part of the application is done using Angular. There are two main components **AppComponent**, **SurveyFormComponent** and **ListSurveyComponent.** The services used to call backend endpoint is **ApiServiceService.**

AppComponent

AppComponent has the basic Jumbotron and two buttons “Student Survey” and “List All Surveys”. Clicking on “Student Survey” will open the survey form below and “List All Survey” will open the all the surveys taken. This was achieved through the **<router-outlet></router-outlet>** and **RouterModule**

```
    <div class="pd-10 text-center">
        <button (click)="survey()" type="button" class="btn green btn-lg btn-block custom-font1">Student Survey</button>
        <button (click)="listSurvey()" type="button" class="btn green btn-lg btn-block custom-font1">List All Surveys</button>

    </div>
    <br>
  </div>
</div>
<router-outlet></router-outlet>
```

```
const routes: Routes = [
  { path: 'survey', component: SurveyFormComponent },
  { path: 'list-survey', component: ListSurveyComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

SurveyFormComponent

Here we use Reactive Forms to create the survey form. It is done by importing **FormBuilder** class and creating form groups for the form and checkboxes.

```
surveyForm: FormGroup = this.fb.group({
  firstName: ['', Validators.required],
  lastName: ['', Validators.required],
  streetAddress: ['', Validators.required],
  city: ['', Validators.required],
  state: ['', Validators.required],
  zip: ['', Validators.required],
  telephone: ['', Validators.required],
  email: ['', Validators.required],
  surveyDate: ['', Validators.required],
  interestedIn: [''],
  recommendation: [''],
  comments: ['']

});

checkboxes: FormGroup =
  this.fb.group({
    students: [false],
    location: [false],
    campus: [false],
    atmosphere: [false],
    dorm: [false],
    sports: [false]
  });
```

And on submit, we are creating the object to send to the backend and calling apiService to call the backend endpoint.

```
submit() {
  // console.log(this.surveyForm.getRawValue())
  if (this.surveyForm.valid) {
    let data = this.surveyForm.getRawValue();
    data.campusLikes = [];
    console.log(this.checkboxes.controls['students'].value)
    if (!!this.checkboxes.controls['students'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Students');
    }
    if (!!this.checkboxes.controls['location'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Location');
    }
    if (!!this.checkboxes.controls['campus'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Campus');
    }
    if (!!this.checkboxes.controls['atmosphere'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Atmosphere');
    }
    if (!!this.checkboxes.controls['dorm'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Dorm Rooms');
    }
    if (!!this.checkboxes.controls['sports'].value && !!this.checkboxes.controls['students'].value == true) {
      data.campusLikes.push('Sports');
    }
    console.log(data)
    this.service.apiCall(data);
    this.router.navigate(['/']);
  }
```

ListSurveyComponent

Here we just call the backend get endpoint to get all the survey data and display it in the table

```
export class ListSurveyComponent implements OnInit {
  surveyData = [];
  constructor(private service: ApiServiceService) { }

  ngOnInit(): void {
    this.service.getData().subscribe((data: any) => {
      console.log(data)
      this.surveyData = data;

    });
  }

  likes(data:any) {
    if(!!data) {
      return data.join(',');
    } else {
      return '';
    }
  }

}
```

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">First Name</th>
      <th scope="col">Last Name</th>
      <th scope="col">Street Address</th>
      <th scope="col">City</th>
      <th scope="col">State</th>
      <th scope="col">Zip</th>
      <th scope="col">Telephone</th>
      <th scope="col">Email</th>
      <th scope="col">Survey Date</th>
      <th scope="col">Campus likes</th>
      <th scope="col">Interested</th>
      <th scope="col">Recommendation</th>
      <th scope="col">Comments</th>



    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let item of surveyData">
      <th>{{item['firstName']}}</th>
      <td>{{item['lastName']}}</td>
      <td>{{item['streetAddress']}}</td>
      <td>{{item['city']}}</td>
      <td>{{item['state']}}</td>
      <td>{{item['zip']}}</td>
      <td>{{item['telephone']}}</td>
      <td>{{item['email']}}</td>
      <td>{{item['surveyDate']}}</td>
      <td>{{likes(item['campusLikes'])}}</td>
      <td>{{item['interestedIn']}}</td>
      <td>{{item['recommendation']}}</td>
      <td>{{item['comments']}}</td>

    </tr>
  </tbody>
</table>
```

ApiServiceService

Here are the backend get and post endpoints that call the backend from Angular. It is done using **HttpClient** from the the HttpClient Module.

```
export class ApiServiceService {

  constructor(private http: HttpClient) { }

  apiCall(data: any): any {
    const headers = new HttpHeaders({
      'Content-Type': 'application/json'
    });

    const requestOptions = { headers: headers };

    return this.http
      .post('http://localhost:8080/api/survey-form', data, requestOptions).subscribe(data => {
        console.log(data);
      });
  }

  getData(): any {
    return this.http
      .get('http://localhost:8080/api/survey-form');
  }
}
```