

SWE642-ASSIGNMENT 3

FULL STACK APPLICATION DEVELOPMENT

1. Team Members

- Meet Rajesh Popat - G01506756
- Tanuja Konda Reddy - G01507320
- Ritika Prashanth - G01487838
- Manasa Mummaadi - G01515437

2. Introduction

This project implements a full-stack web application to collect and manage student survey data. The frontend is built with Angular, and the backend is a Spring Boot REST API using JPA/Hibernate with MySQL hosted on Amazon RDS. Users can submit a survey and view, update, or delete existing surveys.

3. System Architecture

- Frontend: Angular SPA using standalone components and Reactive Forms.
- Backend: Spring Boot REST controllers exposing CRUD endpoints.
- Database: MySQL on Amazon RDS accessed via JPA/Hibernate.
- Communication: JSON over HTTP between Angular and Spring Boot.

4. Technology Stack

Layer	Technology	Notes
Frontend	Angular 19 (standalone APIs)	Standalone components, Angular Router, HttpClient, Reactive Forms
Backend	Spring Boot (REST)	Controllers, Services, Repositories
ORM	JPA / Hibernate	Entity mapping to MySQL
Database	MySQL (Amazon RDS)	Cloud-hosted, secured via SG
Tools	Node.js, npm, Maven, Spring, JDK, JRE	Build & test

5. Database Design

Primary Entity: StudentSurvey

Fields :

- id - number (optional)
- first_name - string (required)
- last_name - string (required)
- street - string (required)
- city - string (required)
- state - string (required)
- zip - string (required)
- telephone - string (required)
- email - string (required, email format)
- survey_date - string (required, date)
- liked - string[] (checkboxes: students, location, campus, atmosphere, dorm rooms, sports)
- interest_source - enum: friends | television | internet | other
- recommend - enum: Very Likely | Likely | Unlikely
- comments - string (optional)

```
SpringAPI > student-survey-api > src > main > resources > db > migration > V1_init.sql
1  -- //Group Project: Student Survey
2  -- //Members: Meet, Tanuja, Ritika, Manasa
3  CREATE TABLE IF NOT EXISTS surveys (
4      id BIGINT PRIMARY KEY AUTO_INCREMENT,
5      first_name  VARCHAR(100) NOT NULL,
6      last_name   VARCHAR(100) NOT NULL,
7      street      VARCHAR(200) NOT NULL,
8      city        VARCHAR(100) NOT NULL,
9      state       VARCHAR(100) NOT NULL,
10     zip         VARCHAR(20) NOT NULL,
11     telephone   VARCHAR(50) NOT NULL,
12     email       VARCHAR(255) NOT NULL,
13     survey_date DATE        NOT NULL,
14     liked        JSON        NOT NULL,
15     interest_source ENUM('friends','television','internet','other') NOT NULL,
16     recommend    ENUM('Very Likely','Likely','Unlikely')          NOT NULL,
17     comments     TEXT,
18     created_at   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
19     updated_at   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
20     UNIQUE KEY ux_surveys_email_date (email, survey_date)
21 );
22
```

6. Backend Implementation (Spring Boot)

This section will include controller endpoints, service layer, repository interfaces, entity annotations, application.properties (RDS connection), and CURL examples.

6.1 Tech & Requirements

- Java 17 or newer
- Maven 3.8+

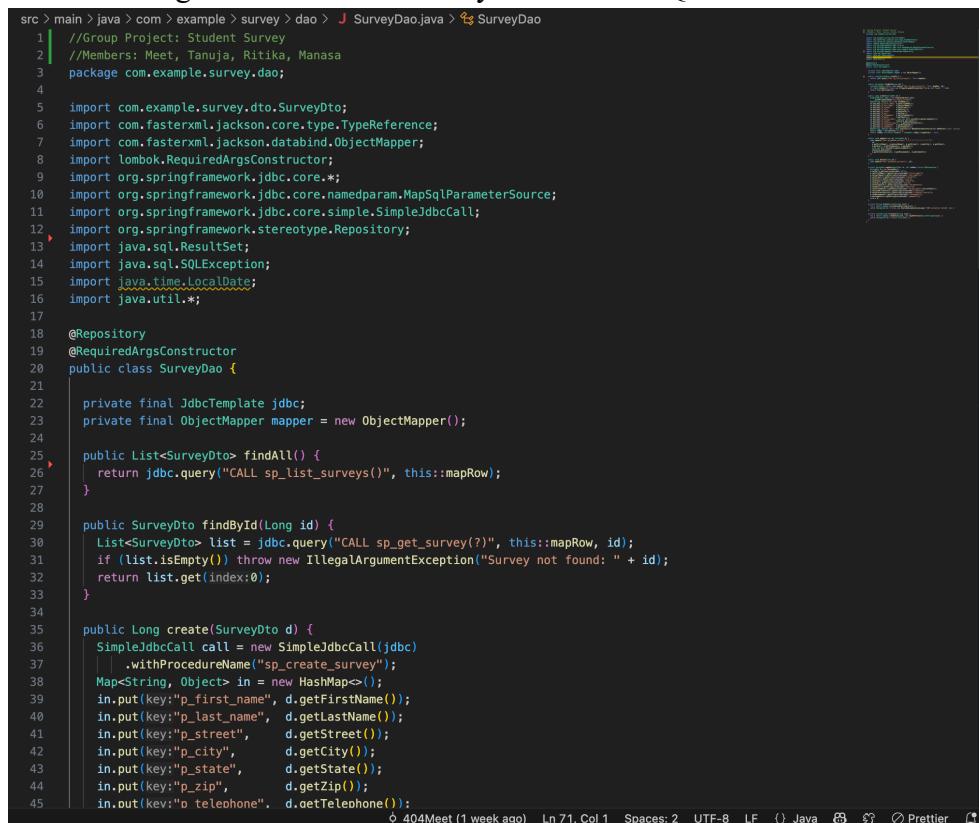
- Spring Boot (Web, Data JPA, Validation)
- Database: MySQL in production; H2 in-memory for local dev/testing

6.2 Project Layout

- The repository uses standard Maven layout.
- Data Access object – SurveyDao.java
- Data Transfer Object – SurveyDto.java
- Service – SurveyService.java
- Controller – SurveyController.java
- Start Point – StudentSurveyApiApplication.java
- DB Migrations- V1__Init.sql, V2__Procedures.sql
- Properties- application.properties
- Dependencies: pom.xml

6.3 Data Access Object

- This class directly interacts with the database layer.
- Typically wraps JPA repository calls.
- Handles CRUD operations like findAll(), create(), delete(), etc.
- In many modern Spring apps, DAOs are often replaced by JpaRepository interfaces, but having a DAO adds flexibility for custom SQL.



```

src > main > java > com > example > survey > dao > SurveyDao.java > SurveyDao
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritiika, Manasa
3 package com.example.survey.dao;
4
5 import com.example.survey.dto.SurveyDto;
6 import com.fasterxml.jackson.core.type.TypeReference;
7 import com.fasterxml.jackson.databind.ObjectMapper;
8 import lombok.RequiredArgsConstructor;
9 import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
10 import org.springframework.jdbc.core.simple.SimpleJdbcCall;
11 import org.springframework.stereotype.Repository;
12 import java.sql.ResultSet;
13 import java.sql.SQLException;
14 import java.time.LocalDate;
15 import java.util.*;
16
17
18 @Repository
19 @RequiredArgsConstructor
20 public class SurveyDao {
21
22     private final JdbcTemplate jdbc;
23     private final ObjectMapper mapper = new ObjectMapper();
24
25     public List<SurveyDto> findAll() {
26         return jdbc.query("CALL sp_list_surveys()", this::mapRow);
27     }
28
29     public SurveyDto findById(Long id) {
30         List<SurveyDto> list = jdbc.query("CALL sp_get_survey(?)", this::mapRow, id);
31         if (list.isEmpty()) throw new IllegalArgumentException("Survey not found: " + id);
32         return list.get(index:0);
33     }
34
35     public Long create(SurveyDto d) {
36         SimpleJdbcCall call = new SimpleJdbcCall(jdbc)
37             .withProcedureName("sp_create_survey");
38         Map<String, Object> in = new HashMap<>();
39         in.put(key:"p_first_name", d.getFirstName());
40         in.put(key:"p_last_name", d.getLastName());
41         in.put(key:"p_street", d.getStreet());
42         in.put(key:"p_city", d.getCity());
43         in.put(key:"p_state", d.getState());
44         in.put(key:"p_zip", d.getZip());
45         in.out(key:"o_telephone", d.setTelephone());

```

404Meet (1 week ago) Ln 71, Col 1 Spaces: 2 UTF-8 LF () Java ⚡ ⚡ ⚡ Prettier ⚡

6.4 Data Transfer Object

- A DTO (Data Transfer Object) is used to send and receive data between layers (e.g. frontend - backend) without exposing the full entity model.
- Helps enforce validation and prevent exposing internal fields like IDs or timestamps.
- Useful for mapping user input to entity objects.

```
src > main > java > com > example > survey > dto > SurveyDto.java > ...
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritika, Manasa
3 package com.example.survey.dto;
4
5 import jakarta.validation.constraints.*;
6 import lombok.*;
7 import java.time.LocalDate;
8 import java.util.List;
9
10 @Getter @Setter @NoArgsConstructor @AllArgsConstructor @Builder
11 public class SurveyDto {
12     private Long id;
13
14     @NotBlank private String firstName;
15     @NotBlank private String lastName;
16     @NotBlank private String street;
17     @NotBlank private String city;
18     @NotBlank private String state;
19     @NotBlank private String zip;
20     @NotBlank private String telephone;
21     @Email @NotBlank private String email;
22
23     @NotNull private LocalDate surveyDate;
24
25     @NotNull private List<String> liked; // sent/received as JSON array
26
27     @NotBlank private String interestSource; // friends | television | internet | other
28     @NotBlank private String recommend; // Very Likely | Likely | Unlikely
29
30     private String comments;
31 }
32
```

6.5 Service

- The business logic layer of your app.
- Acts as a bridge between Controller (web layer) and DAO/Repository (data layer).
- Handles validations, transformations, and transactional operations.

```
src > main > java > com > example > survey > service > SurveyService.java > ...
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritika, Manasa
3 package com.example.survey.service;
4
5 import com.example.survey.dao.SurveyDao;
6 import com.example.survey.dto.SurveyDto;
7 import lombok.RequiredArgsConstructor;
8 import org.springframework.stereotype.Service;
9
10 import java.util.List;
11
12 @Service
13 @RequiredArgsConstructor
14 public class SurveyService {
15     private final SurveyDao dao;
16
17     public List<SurveyDto> findAll() { return dao.findAll(); }
18     public SurveyDto findById(Long id) { return dao.findById(id); }
19     public SurveyDto create(SurveyDto dto) { Long id = dao.create(dto); return dao.findById(id); }
20     public SurveyDto update(Long id, SurveyDto dto) { dao.update(id, dto); return dao.findById(id); }
21     public void delete(Long id) { dao.delete(id); }
22 }
23
```

6.6 Controller

- The entry point for HTTP requests (REST endpoints).
- Handles client requests (GET, POST, PUT, DELETE) and returns JSON responses.
- Uses `@RestController`, `@RequestMapping`, and `@RequiredArgsConstructor`.

```
src > main > java > com > example > survey > web > SurveyController.java > ...
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritika, Manasa
3 package com.example.survey.web;
4
5 import com.example.survey.dto.SurveyDto;
6 import com.example.survey.service.SurveyService;
7 import jakarta.validation.Valid;
8 import lombok.RequiredArgsConstructor;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.net.URI;
13 import java.util.List;
14
15 @RestController
16 @RequestMapping("/api/surveys")
17 @RequiredArgsConstructor
18 public class SurveyController {
19
20     private final SurveyService service;
21
22     @GetMapping
23     public List<SurveyDto> all() { return service.findAll(); }
24
25     @GetMapping("/{id}")
26     public SurveyDto one(@PathVariable Long id) { return service.findById(id); }
27
28     @PostMapping
29     public ResponseEntity<SurveyDto> create(@Valid @RequestBody SurveyDto dto) {
30         SurveyDto saved = service.create(dto);
31         return ResponseEntity.created(URI.create("/api/surveys/" + saved.getId())).body(saved);
32     }
33
34     @PutMapping("/{id}")
35     public SurveyDto update(@PathVariable Long id, @Valid @RequestBody SurveyDto dto) {
36         return service.update(id, dto);
37     }
38
39     @DeleteMapping("/{id}")
40     public ResponseEntity<Void> delete(@PathVariable Long id) {
41         service.delete(id);
42         return ResponseEntity.noContent().build();
43     }
44 }
```

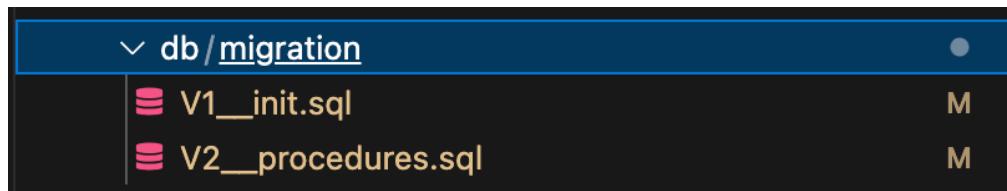
6.7 Start Point (StudentSurveyApiApplication.java)

- The main class that boots the entire Spring application.
- Contains the main() method with SpringApplication.run() which starts the embedded Tomcat server.

```
src > main > java > com > example > survey > J StudentSurveyApiApplication.java > ...
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritika, Manasa
3 package com.example.survey;
4
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 @SpringBootApplication
9 public class StudentSurveyApiApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(StudentSurveyApiApplication.class, args);
13     }
14
15 }
16
```

6.8 Database Migration

- These are Flyway migration scripts used to version-control your database schema.
- Automatically executed when the app starts (if Flyway is enabled).
- Example:
 - V1__Init.sql: creates tables and constraints
 - V2__Procedures.sql: adds stored procedures or triggers



6.9 Properties (application.properties)

- Central configuration file for Spring Boot.
- Defines port, DB connection, logging, and JPA behavior.

```
src > main > resources > application.properties
1 # //Group Project: Student Survey
2 # //Members: Meet, Tanuja, Ritika, Manasa
3 spring.application.name=student-survey-api
4 # --- Datasource (local MySQL) ---
5 spring.datasource.url=jdbc:mysql://student-survey-mysql.cghk2266oph7.us-east-1.rds.amazonaws.com:3306/student_sur
6 spring.datasource.username=*****
7 spring.datasource.password=*****
8
9 # --- JPA/Hibernate ---
10 spring.jpa.hibernate.ddl-auto=validate
11 spring.jpa.properties.hibernate.format_sql=true
12
13 # --- Flyway (DB migrations) ---
14 spring.flyway.enabled=true
15 spring.flyway.locations=classpath:db/migration
16
17 # --- Jackson ---
18 spring.jackson.serialization.write-dates-as-timestamps=false
19
20 # --- Server ---
21 server.port=8080
22
23 # --- CORS (used by WebConfig @Value) ---
24 app.cors.allowed-origins=http://swe642-hw3-meet.s3-website-us-east-1.amazonaws.com/
```

6.10 Dependencies (pom.xml)

- Maven's Project Object Model file.
- Lists all required libraries, build plugins, and metadata.

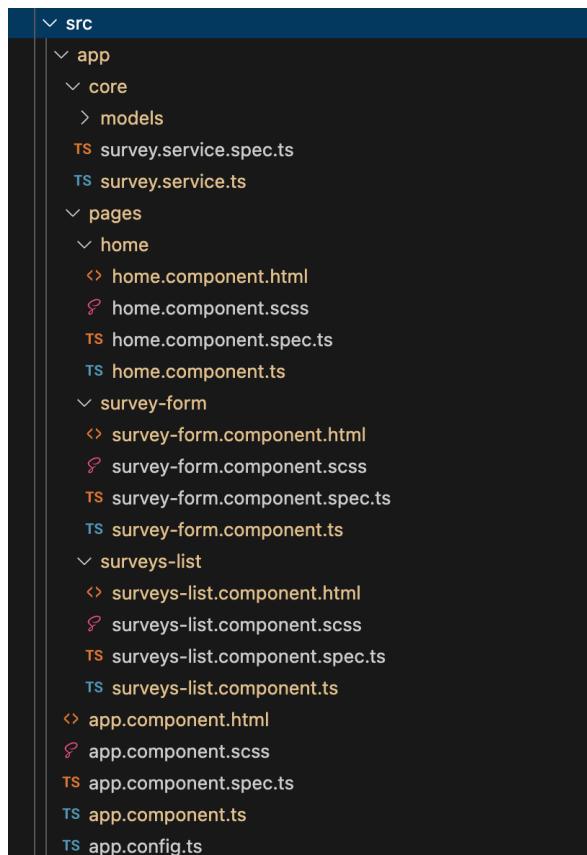
```
pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   <dependencies>
32
37     <dependency>
38       <groupId>org.springframework.boot</groupId>
39       <artifactId>spring-boot-starter-data-jpa</artifactId>
40     </dependency>
41     <dependency>
42       <groupId>org.springframework.boot</groupId>
43       <artifactId>spring-boot-starter-validation</artifactId>
44     </dependency>
45     <dependency>
46       <groupId>org.springframework.boot</groupId>
47       <artifactId>spring-boot-starter-web</artifactId>
48     </dependency>
49     <dependency>
50       <groupId>org.flywaydb</groupId>
51       <artifactId>flyway-core</artifactId>
52     </dependency>
53     <dependency>
54       <groupId>org.flywaydb</groupId>
55       <artifactId>flyway-mysql</artifactId>
56     </dependency>
57
58     <dependency>
59       <groupId>com.mysql</groupId>
60       <artifactId>mysql-connector-j</artifactId>
61       <scope>runtime</scope>
62     </dependency>
63     <dependency>
64       <groupId>org.projectlombok</groupId>
65       <artifactId>lombok</artifactId>
66       <optional>true</optional>
67     </dependency>
68     <dependency>
69       <groupId>org.springframework.boot</groupId>
70       <artifactId>spring-boot-starter-test</artifactId>
71       <scope>test</scope>
72     </dependency>
73     <dependency>
74       <groupId>com.vladmihalcea</groupId>
75       <artifactId>hibernate-types-60</artifactId>
76       <version>2.21.1</version>
77     </dependency>
78   </dependencies>
79 
```

7. Frontend Implementation (Angular)

7.1 Project Structure

Root: src/app/ with the following notable areas:

- core/survey.service.specs.ts — Testing file for service.
- core/survey.service.ts — HttpClient-based CRUD methods
- core/models/survey.ts — Interface (Properties for service)
- pages/home — HomeComponent for the welcome page
- pages/survey-form — SurveyFormComponent for create/update form
- pages/surveys-list — SurveysListComponent for listing and actions



7.2 Routing (Standalone)

Routes configured in app.routes.ts:

- path: "" → HomeComponent
- path: 'survey' → SurveyFormComponent
- path: 'surveys' → SurveysListComponent
- path: 'surveys/:id/edit' → SurveyFormComponent

```

Angular > student-survey > src > app > ts app.routes.ts > ...
1 import { Routes } from '@angular/router';
2 import { HomeComponent } from './pages/home/home.component';
3 import { SurveyFormComponent } from './pages/survey-form/survey-form.component';
4 import { SurveysListComponent } from './pages/surveys-list/surveys-list.component';
5
6 export const routes: Routes = [
7   { path: '', component: HomeComponent },
8   { path: 'survey', component: SurveyFormComponent },
9   { path: 'surveys', component: SurveysListComponent },
10  { path: 'surveys/:id/edit', component: SurveyFormComponent },
11  { path: '**', redirectTo: '' }
12];
13

```

7.3 Components

- HomeComponent — Welcome page with links to Student Survey and List All Surveys.

```

Angular > student-survey > src > app > pages > home > ts home.component.ts > ...
1 //Group Project: Student Survey
2 //Members: Meet, Tanuja, Ritika, Manasa
3
4 import { Component } from '@angular/core';
5 import { RouterModule } from '@angular/router';
6 import { CommonModule } from '@angular/common';
7
8 @Component({
9   selector: 'app-home',
10  (property) Component.styleUrls?: string[] | undefined
11  Relative paths or absolute URLs for files containing CSS styleheets to use in this component.
12  styleUrls: ['./home.component.scss']
13})
14 export class HomeComponent {}
15
16

```

- SurveyFormComponent — Reactive form to create/update a survey (required fields, checkboxes, radios, dropdown).

```

import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormBuilder, FormGroup, FormArray, FormControl, ReactiveFormsModule, Validators } from '@angular/forms';
import { Router, RouterModule, ActivatedRoute } from '@angular/router';
import { SurveyService } from '../../../../../core/survey.service';
import { Survey } from '../../../../../core/models/survey';

@Component({
  selector: 'app-survey-form',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule, RouterModule],
  templateUrl: './survey-form.component.html',
  styleUrls: ['./survey-form.component.scss']
})
export class SurveyFormComponent implements OnInit {
  form!: FormGroup;
  editId?: number;

  constructor(
    private fb: FormBuilder,
    private svc: SurveyService,
    private route: ActivatedRoute,
    private router: Router
  ) {}

  ngOnInit(): void {
    this.form = this.fb.group({
      firstName: ['', Validators.required],
      lastName: ['', Validators.required],
      street: ['', Validators.required],
      city: ['', Validators.required],
      state: ['', Validators.required],
      zip: ['', Validators.required],
      telephone: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      surveyDate: ['', Validators.required],
      recommend: ['', Validators.required],
    })
  }
}

```

```

    liked: new FormArray([
      new FormControl(false, { nonNullable: true }),
      new FormControl(false, { nonNullable: true })
    ]),
    interestSource: ['', Validators.required],
    comments: ['']
  });

  const id = this.route.snapshot.paramMap.get('id');
  if (id) {
    this.editId = +id;
    this.svc.getById(this.editId).subscribe(s => this.patchFromModel(s));
  }
}

get likedArray(): FormArray<FormControl<boolean>> {
  return this.form.get('liked') as FormArray<FormControl<boolean>>;
}

private patchFromModel(s: Survey) {
  this.form.patchValue({
    firstName: s.firstName,
    lastName: s.lastName,
    street: s.street,
    city: s.city,
    state: s.state,
    zip: s.zip,
    telephone: s.telephone,
    email: s.email,
    surveyDate: s.surveyDate,
    recommend: s.recommend,
    interestSource: s.interestSource,
    comments: s.comments
  });

  const likedKeys = ['campus', 'dorms', 'students', 'location', 'atmosphere', 'sports'];
  likedKeys.forEach((key, i) => {
    this.likedArray.at(i).setValue(s.liked?.includes(key) ?? false);
  });
}

submit() {
  if (this.form.invalid) {
    alert('Please fill all required fields');
    return;
  }

  const likedKeys = ['campus', 'dorms', 'students', 'location', 'atmosphere', 'sports'];
  const likedPicked = this.likedArray.value
    .map(v, i) => (v ? likedKeys[i] : null)
    .filter(Boolean) as string[];

  const payload: Survey = { ...this.form.getRawValue(), liked: likedPicked };

  const req$ = this.editId
    ? this.svc.update(this.editId, payload)
    : this.svc.create(payload);

  req$.subscribe({
    next: () => {
      alert('Survey submitted successfully!');
      this.router.navigate(['/surveys']);
    },
    error: (err) => {
      console.error('Save failed:', err);
      alert(`Save failed: ${err?.status || ''} ${err?.statusText || err?.message || ''}`);
    }
  });
}

cancel() {
  this.router.navigate(['/']);
}

```

- SurveysListComponent — Tabular list of all surveys with Update/Delete actions.

```

Angular > student-survey > src > app > pages > surveys-list > ts surveys-list.component.ts > ...
1  //Group Project: Student Survey
2  //Members: Meet, Tanuja, Ritika, Manasa
3
4  import { Component, OnInit } from '@angular/core';
5  import { SurveyService } from '../../core/survey.service';
6  import { Survey } from '../../core/models/survey';
7  import { RouterModule, Router } from '@angular/router';
8  import { CommonModule } from '@angular/common';
9  import { finalize } from 'rxjs/operators';
10
11 @Component({
12   selector: 'app-surveys-list',
13   standalone: true,
14   imports: [RouterModule, CommonModule],
15   templateUrl: './surveys-list.component.html',
16   styleUrls: ['./surveys-list.component.scss']
17 })
18 export class SurveysListComponent implements OnInit {
19   rows: Survey[] = [];
20   loading = true;
21   errorMsg = '';
22
23   constructor(private svc: SurveyService, private router: Router) {}
24
25   ngOnInit(): void {
26     this.load();
27   }
28
29   load() {
30     this.loading = true;
31     this.errorMsg = '';
32     this.svc.getAll()
33       .pipe(finalize(() => this.loading = false))
34       .subscribe({
35         next: (data) => this.rows = data ?? [],
36         error: (err) => {
37           console.error('Survey fetch failed', err);
38           this.rows = [];
39           this.errorMsg = typeof err?.message === 'string'
40             ? err.message
41             : 'Could not load surveys.';
42         }
43       });
44   }
45
46   edit(id?: number) {
47     if (id) this.router.navigate(['/surveys', id, 'edit']);
48   }
49
50   remove(id?: number) {
51     if (!id) return;
52     if (!confirm('Delete this survey?')) return;
53     this.svc.delete(id).subscribe(() => this.load());
54   }
55 }
56

```

7.4 Services (HttpClient)

Base API URL: <http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys>

Methods:

- getAll() - GET <http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys>
- getById(id) -GET <http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys/{id}> — Fetch a survey by ID (if implemented).

- `create(survey)` — POST `http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys` — Create a new survey.
- `update(id, survey)` — PUT `http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys/{id}` — Update an existing survey.
- `delete(id)` — DELETE `http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys/{id}` — Delete a survey by ID.

```
Angular > student-survey > src > app > core > survey.service.ts > ...
1  //Members: Meet, Tanuja, Ritika, Manasa
2
3
4  import { Injectable } from '@angular/core';
5  import { HttpClient } from '@angular/common/http';
6  import { Observable } from 'rxjs';
7  import { Survey } from './models/survey';
8
9  @Injectable({ providedIn: 'root' })
10 export class SurveyService {
11   private baseUrl = 'http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys';
12
13   constructor(private http: HttpClient) {}
14
15   getAll(): Observable<Survey[]> {
16     return this.http.get<Survey[]>(this.baseUrl);
17   }
18
19   getById(id: number): Observable<Survey> {
20     return this.http.get<Survey>(` ${this.baseUrl}/ ${id}`);
21   }
22
23   create(s: Survey): Observable<Survey> {
24     return this.http.post<Survey>(this.baseUrl, s);
25   }
26
27   update(id: number, s: Survey): Observable<Survey> {
28     return this.http.put<Survey>(` ${this.baseUrl}/ ${id}` , s);
29   }
30
31   delete(id: number): Observable<void> {
32     return this.http.delete<void>(` ${this.baseUrl}/ ${id}`);
33   }
34 }
```

7.5 Reactive Forms & Validation

`SurveyFormComponent` builds a `FormGroup` with required validators for `firstName`, `lastName`, `street`, `city`, `state`, `zip`, `telephone`, `email`, `surveyDate`, and `validators.email` for `email`. It binds checkboxes (`liked`), radio buttons (`interestSource`), and dropdown (`recommend`).

```
ngOnInit(): void {
  this.form = this.fb.group({
    firstName: ['', Validators.required],
    lastName: ['', Validators.required],
    street: ['', Validators.required],
    city: ['', Validators.required],
    state: ['', Validators.required],
    zip: ['', Validators.required],
    telephone: ['', Validators.required],
    email: ['', [Validators.required, Validators.email]],
    surveyDate: ['', Validators.required],
    recommend: ['', Validators.required],
    liked: new FormArray([
      new FormControl(false, { nonNullable: true }),
      new FormControl(false, { nonNullable: true })
    ]),
    interestSource: ['', Validators.required],
    comments: ['']
  });
}
```

7.6 List Page & Actions

SurveysListComponent displays all surveys in a table, handles loading/error states, and triggers update/delete actions via SurveyService. Includes navigation to edit route /surveys/:id/edit.

```
<table class="table table-bordered">
  *ngIf="!loading && (rows?.length ?? 0) > 0">
  <thead>
    <tr>
      <th>Name</th><th>Email</th><th>Date</th><th>Recommend</th><th></th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let s of rows">
      <td>{{ s.firstName }} {{ s.lastName }}</td>
      <td>{{ s.email }}</td>
      <td>{{ s.surveyDate | date:'mediumDate' }}</td>
      <td>{{ s.recommend }}</td>
      <td>
        <button class="btn btn-sm btn-outline-secondary" (click)="edit(s.id)">Edit</button>
        <button class="btn btn-sm btn-outline-danger" (click)="remove(s.id)">Delete</button>
      </td>
    </tr>
  </tbody>
</table>
```

8. Frontend–Backend Integration

Angular consumes the Spring Boot REST API using HttpClient. CORS must be enabled on the backend. End-to-end flow: User submits survey → Angular POST /api/surveys → Spring Boot persists via JPA → Record appears on List All Surveys.

GET

The screenshot shows the Chrome DevTools Network tab with a single request listed:

- Name:** surveys
- Request URL:** http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** 3.85.18.106:80
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
 - Access-Control-Allow-Credentials: true
 - Access-Control-Allow-Origin: http://swe642-hw3-meet.s3-website-us-east-1.amazonaws.com
 - Content-Length: 2767
 - Content-Type: application/json
 - Date: Fri, 24 Oct 2025 13:46:07 GMT
 - Keep-Alive: timeout=60
 - Vary: Accept-Encoding
 - X-Powered-By: ARR/3.0
- Request Headers:** Accept: application/json, text/plain, */*

POST

All Surveys

Name	Email	Date	Recommendation
Manasa Mummadi	manasammudi2@gmail.com	Oct 24, 2025	Likely
Wilson charles	wilson@gmail.com	Oct 23, 2025	Very Likely
b b	b@gmail.com	Oct 16, 2025	Very Likely
a a	a@gmail.com	Oct 16, 2025	Very Likely

9. Deployment Setup

Amazon RDS: MySQL instance with security group allowing the backend host. Backend is packaged as a JAR and ran on an EC2 instance. Angular can be served locally (ng serve) or deployed as static assets (ng build) as we did behind S3 bucket.

S3 Bucket: <http://swe642-hw3-meet.s3-website-us-east-1.amazonaws.com/>

Objects (8)

Name	Type	Last modified	Size	Storage class
favicon.ico	ico	October 16, 2025, 19:55:36 (UTC-04:00)	14.7 KB	Standard
index.html	html	October 16, 2025, 19:55:36 (UTC-04:00)	5.0 KB	Standard
main-HBSUL4TY.js	js	October 16, 2025, 19:47:44 (UTC-04:00)	302.2 KB	Standard
main-Z3TIVBEP.js	js	October 16, 2025, 19:55:36 (UTC-04:00)	305.4 KB	Standard
media/	Folder	-	-	-
polyfills-B6TNH2Q6.js	js	October 16, 2025, 19:55:36 (UTC-04:00)	33.8 KB	Standard
scripts-SQ7W6IC7.js	js	October 16, 2025, 19:55:36 (UTC-04:00)	78.5 KB	Standard
styles-VRDYZCWE.css	css	October 16, 2025, 19:55:36 (UTC-04:00)	225.6 KB	Standard

EC2 Instance: <http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys>

Instance summary for i-0d456e3341e897244 (swe642-Meet) Info

Updated less than a minute ago

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
Instance details Info AMI ID: ami-0feef160a1079475 AMI name: Windows_Server-2025-English-Full-Base-2025.08.13 Stop protection: Disabled Instance reboot migration: Default (On) Stop-hibernate behavior: Disabled State transition reason: -						
Monitoring: disabled Allowed image: - Launch time: Wed Oct 08 2025 19:10:16 GMT-0400 (Eastern Daylight Time) (16 days) Instance auto-recovery: Default AMI Launch index: 0 Credit specification: unlimited						
Platform details: Windows Termination protection: Disabled AMI location: amazon/Windows_Server-2025-English-Full-Base-2025.08.13 Lifecycle: normal Key pair assigned at launch: AWS.Windows.Key Kernel ID: -						

```

meet@Meets-MacBook-Air ~ % curl -i http://ec2-3-85-18-106.compute-1.amazonaws.com/api/surveys
HTTP/1.1 200 OK
Keep-Alive: timeout=60
Content-Type: application/json
Vary: Access-Control-Request-Headers
X-Powered-By: ARR/3.0
Date: Thu, 23 Oct 2025 20:53:50 GMT
Content-Length: 2140

[{"id":8,"firstName":"b","lastName":"b","street":"b 123 st","city":"b","state":"b","zip":"22030","telephone":"57126311281","email":"@gmail.com","surveyDate":"2025-10-16","liked":["campus","dorms","students","location","atmosphere","sports"],"interestSource":"internet","recommend":"Very Likely","comments":""}, {"id":7,"firstName":"a","lastName":"a","street":"a","city":"a","state":"a","zip":"22030","telephone":"5712631261","email":"a@gmail.com","surveyDate":"2025-10-16","liked":[],"interestSource":"friends","recommend":"Very Likely","comments":"N/A"}, {"id":6,"firstName":"John","lastName":"Doe","street":"123 Main St","city":"Fairfax","state":"VA","zip":"22030","telephone":"7035551212","email":"john_doe@example.com","surveyDate":"2025-10-16","liked":["students","campus","dorms"],"interestSource":"friends","recommend":"Very Likely","comments":"Testing curl insert"}, {"id":5,"firstName":"Tanuja","lastName":"Reddy","street":"9826 Fairfax Square","city":"Fairfax","state":"Virginia","zip":"22031","telephone":"5712633271","email":"tanu@gmail.com","surveyDate":"2025-10-09","liked":["location","campus","atmosphere"],"interestSource":"television","recommend":"Likely","comments":"Hii"}, {"id":4,"firstName":"Ritika","lastName":"Prashanth","street":"9826 Fairfax Square","city":"Fairfax","state":"Virginia","zip":"22030","telephone":"5712691291","email":"riri@gmail.com","surveyDate":"2025-10-09","liked":["location","campus"],"interestSource":"internet","recommend":"Very Likely","comments":"Hello"}, {"id":3,"firstName":"Manasa","lastName":"Anandana","street":"9826 Fairfax Square","city":"Fairfax","state":"Virginia","zip":"22030","telephone":"5712611211","email":"maanu@gmail.com","surveyDate":"2025-10-09","liked":["atmosphere","dorms","sports"],"interestSource":"friends","recommend":"Likely","comments":"Hey There!!!!"}, {"id":1,"firstName":"Meet","lastName":"Popat","street":"3915 FAIRFAX SQ","city":"Fairfax","state":"Virginia","zip":"22031","telephone":"5712631261","email":"meet@gmailmeet@Meets-MacBook-meet@Meets-MacBook-meet@Meets-MacBook-meet@Meets-MacBook-meet@Meets-MacBook-meet@Meets-MacBook-Air ~ %"]

```

Amazon RDS (MySQL):

The screenshot shows the AWS RDS MySQL console for the 'student-survey-mysql' database. The 'Summary' tab is selected, displaying basic information like DB identifier, status (Available), role (Instance), engine (MySQL Community), and region (us-east-1d). The 'Connectivity & security' tab is active, showing endpoint details (student-survey-mysql.cghk2266oph7.us-east-1.rds.amazonaws.com), port (3306), VPC (vpc-0da1d3dbe45420cb9), and subnet group (default-vpc-0da1d3dbe45420cb9). It also lists several subnets and their IP ranges. The 'Networking' section shows the availability zone (us-east-1d) and network type (IPv4). The 'Security' section includes VPC security groups (default, rds-mysql-sg), certificate authority (rds-ca-rsa2048-g1), and DB instance certificate expiration date (October 08, 2026, 15:44 UTC-04:00).

10. Testing & Validation

- Functional testing of CRUD endpoints via CURL and UI.
- Form validation: required fields, email format, date selection.
- Integration testing: Verify DB records after UI actions.

The screenshot shows MySQL Workbench with a query results grid titled 'surveys 1'. The query is 'select * from surveys;'. The results show 10 rows of survey data:

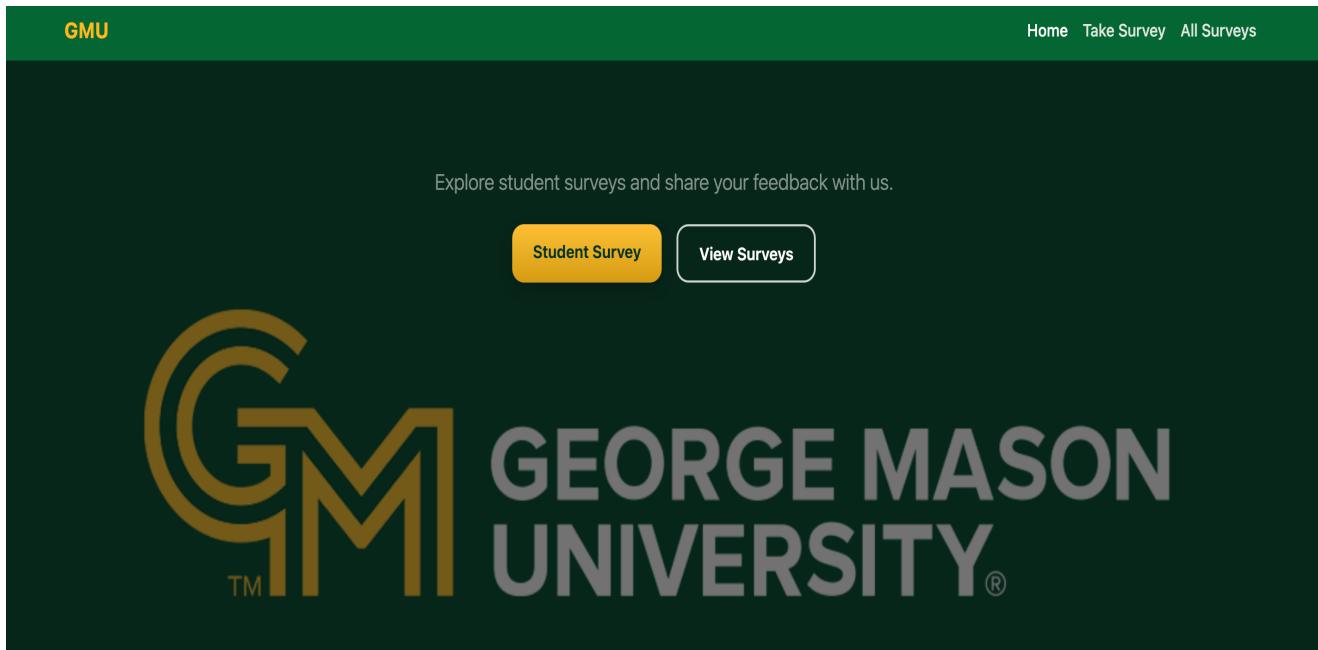
	id	first_name	last_name	street	city	state	zip	telephone	email	survey_date	liked
1	Meet	Popat		3915 FAIRFAX SQ	Fairfax	Virginia	22031	5712631261	meet@gmail.com	2025-10-08	[{"students": "location", "campus": "atmosphere", "dorms": "sports"}]
3	Manasa	Anandha		9826 Fairfax Square	Fairfax	Virginia	22030	5712611211	maanu@gmail.com	2025-10-09	[{"location": "campus", "atmosphere": "dorms"}]
4	Ritika	Prashanth		9826 Fairfax Square	Fairfax	Virginia	22031	5712691291	riti@gmail.com	2025-10-09	[{"location": "campus", "atmosphere": "dorms"}]
5	Tanuja	Reddy		9826 Fairfax Square	Fairfax	Virginia	22031	5712633271	tanu@gmail.com	2025-10-10	[{"location": "campus", "atmosphere": "dorms"}]
6	John	Doe		123 Main St	Fairfax	VA	22030	703551212	john.doe@gmail.com	2025-10-16	[{"students": "campus", "dorms": "atmosphere"}]
7	a	a	a	a	a	a	a	5712631261	a@gmail.com	2025-10-16	[]
8	b	b	b 123 st	b	b	Virginia	22030	57126311281	b@gmail.com	2025-10-16	[{"campus": "dorms", "students": "location", "atmosphere": "dorms"}]
10	Wilson	charles		2831 Ashburn	Ashburn	VA	22030	571263272	wilson@gmail.com	2025-10-23	[{"campus": "dorms", "students": "atmosphere"}]

11. Application Overview

11.1 Home Page:

We start on the Home Page, which is a clean entry point. Displays a welcoming message and two navigation buttons:

- “**Student Survey**” - opens the survey form page
- “**View Surveys**” - opens the table view of all existing survey entries



11.2 Student Survey Page:

Navigate to the Student Survey page by clicking on **Student Survey** Button

- **Required Fields:** All personal and contact details, like name, address, email, and phone number, are required.
- **Checkboxes:** We use checkboxes for what they liked most about the campus: students, location, campus, and more.
- **Radio Buttons:** They indicate how they heard about us using radio buttons for friends, television, or internet.
- **Dropdown:** And they select their likelihood of recommending the school from a simple dropdown list.

Student Survey

First Name *

Last Name *

Street *

City *

State *

ZIP *

Telephone *

Email *

Survey Date *

 mm/dd/yyyy

Likelihood to Recommend *

What did you like most?

- Campus
- Dorm Rooms
- Students
- Location
- Atmosphere
- Sports

How did you become interested? *

- Friends
- Television
- Internet
- Other

Comments

When the user hits Submit, the Angular app fires a RESTful call to our backend to store the data permanently in the database.

11.3 All Surveys Page:

This page shows all surveys recorded to date in a simple table format. This is our Read operation.

- **Update:** Every row has an **Edit** button. When clicked, it takes us back to the survey form, pre-filling all the data. We can make changes and resubmit. That handles our **Update** requirement.
- **Delete:** And of course, there's a **Delete** button right next to it. One click, and the record is gone.

All Surveys

[+ New Survey](#)

Name	Email	Date	Recommend	
Wilson charles	wilson@gmail.com	Oct 23, 2025	Very Likely	Edit Delete
b b	b@gmail.com	Oct 16, 2025	Very Likely	Edit Delete
a a	a@gmail.com	Oct 16, 2025	Very Likely	Edit Delete
John Doe	john_doe@example.com	Oct 16, 2025	Very Likely	Edit Delete
Tanuja Reddy	tanu@gmail.com	Oct 9, 2025	Likely	Edit Delete
Ritika Prashanth	riri@gmail.com	Oct 9, 2025	Very Likely	Edit Delete
Manasa Anandana	maanu@gmail.com	Oct 9, 2025	Likely	Edit Delete
Meet Popat	meet@gmail.com	Oct 8, 2025	Very Likely	Edit Delete

12. Conclusion

This project demonstrates a complete Angular + Spring Boot + MySQL (RDS) stack implementing a student survey with full CRUD functionality: **Create**, **Read**, **Update**, and **Delete**, all driven by the Angular frontend.

Website URL: <http://swe642-hw3-meet.s3-website-us-east-1.amazonaws.com/>