Project Documentation

SWE 642
Assignment-3

Team Members:
Ayush Noel Kattupalli, G01395882
Sai Venkata Dhanush Amirineni, G01377880
Sai Shishwan Gande, G01383809
Sai Hruthik Karumanchi, G01352466

Campus Survey - Web Application

Introduction:

This project aims to develop a full-stack web application that allows prospective students to fill out a survey form and provide feedback about their campus visit, as well as allow users to view all surveys recorded to date. The frontend portion of the application will be developed using Angular 2 or higher version, while the backend portion will be developed using RESTful Web Services, and JDBC or JPA. The backend implementation will use the Spring Boot platform.

The application will start with a welcome homepage and header with two links: "Survey Form" and "Surveys List". The "Survey Form" link will allow prospective students to fill out a survey form with various fields, including first name, last name, street address, city, state, zip, telephone number, e-mail, and date of survey, as well as checkboxes for indicating what they liked most about the campus, radio buttons for indicating how they became interested in the university, a dropdown list for selecting the likelihood of recommending the school to other prospective students, and a text area for additional comments. The Angular application will invoke a RESTful call when the user submits the survey, which will store the data in the database using JDBC or JPA.

The "Surveys List" link will allow users to view all surveys done to date. This project aims to provide a complete full-stack development experience, covering frontend and backend technologies, database management, and RESTful API design and implementation.

Technologies used:

1. Angular 12,
2. Spring Boot,
3. JDBC/JPA, and any other relevant libraries.
4. MariaDB
5. DBeaver

API:

1. The first API is a POST API that allows prospective students to submit their survey data to the backend server. The API endpoint is http://localhost:8080/api/surveys and it accepts a POST request. The request body should contain the data submitted by the student in the survey form, which includes the student's personal information, their feedback on the

campus visit, and other relevant data. The API endpoint is not authenticated, meaning anyone can access and use this API.
The return type of the POST API is a JSON object that contains the submitted survey data along with a unique identifier assigned to each survey for future reference. The response will include the HTTP status code 201 indicating that the survey has been created successfully.

2. The second API is a GET API that allows users to retrieve all the survey data collected so far. The API endpoint is http://localhost:8080/api/surveys and it accepts a GET request. This API endpoint is not authenticated, meaning anyone can access and use this API.

   The return type of the GET API is a JSON object that contains an array of all the surveys that have been submitted so far. Each survey is represented as a JSON object that contains the survey data along with its unique identifier. The response will include the HTTP status code 200 indicating that the request has been processed successfully.

Database Information:

As part of the development of the full-stack web application, a database has been created to store the survey data collected from prospective students. The database has been created using **MariaDB**, which is a popular open-source relational database management system.

The database has been named "campus_survey", and it has one table named "survey". The "survey" table has been designed to store all the survey data collected from prospective students. The table has been created with the necessary columns to capture all the relevant survey data, including the student's personal information, their feedback on the campus visit, and other relevant data.

The table is being accessed and manipulated using either JDBC or JPA, which are Java technologies for interacting with relational databases. JDBC provides a low-level API for database access, while JPA is a higher-level API that abstracts the underlying database access and provides a more object-oriented approach to working with data.

DDL for Survey Table:

```
-- campus_survey.survey definition

CREATE TABLE `survey` (
 `id` bigint(20) NOT NULL AUTO_INCREMENT,
 `first_name` varchar(255) NOT NULL,
 `last_name` varchar(255) NOT NULL,
 `street_address` varchar(255) NOT NULL,
 `city` varchar(255) NOT NULL,
 `state` varchar(255) NOT NULL,
 `zip` varchar(255) NOT NULL,
 `telephone_number` varchar(255) NOT NULL,
 `email` varchar(255) NOT NULL,
 `date_of_survey` timestamp NOT NULL,
 `liked_most_about_campus` varchar(255) NOT NULL,
 `how_became_interested` varchar(255) NOT NULL,
 `likelihood_of_recommending` varchar(255) NOT NULL,
 `additional_comments` varchar(1000) DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Table view in

	Field	Type	Null	Key	Default	Extra
1	id	bigint(20)	NO	PRI	[NULL]	auto_increment
2	first_name	varchar(255)	NO		[NULL]	
3	last_name	varchar(255)	NO		[NULL]	
4	street_address	varchar(255)	NO		[NULL]	
5	city	varchar(255)	NO		[NULL]	
6	state	varchar(255)	NO		[NULL]	
7	zip	varchar(255)	NO		[NULL]	
8	telephone_number	varchar(255)	NO		[NULL]	
9	email	varchar(255)	NO		[NULL]	
10	date_of_survey	timestamp	NO		[NULL]	
11	liked_most_about_campus	varchar(255)	NO		[NULL]	
12	how_became_interested	varchar(255)	NO		[NULL]	
13	likelihood_of_recommending	varchar(255)	NO		[NULL]	
14	additional_comments	varchar(1000)	YES		[NULL]	

Instructions to start the application:

1. campus-survey-app is the front-end code open the root folder of this project. and run the below commands in sequence to start the application. This will start the application on port 4200

    Commands:

    1. **npm i**
    2. **ng serve**

2. Campus-survey-server is the REST API server built using the java spring-boot. Install the IntelliJ IDE, open the project root folder inside it. Install the dependencies and then go to the main file and run the program to start the server. This will start the server on port 8080