

SWE642 – HW3: Build survey form using Angular and Spring Boot

Team Member: Aniket Pandey, Siddhanth Kalyanpur, Adel Alkhamisy

Step 1: Write POST and GET api using Spring Tool Suite and connect to SQL

- Following are the two APIs

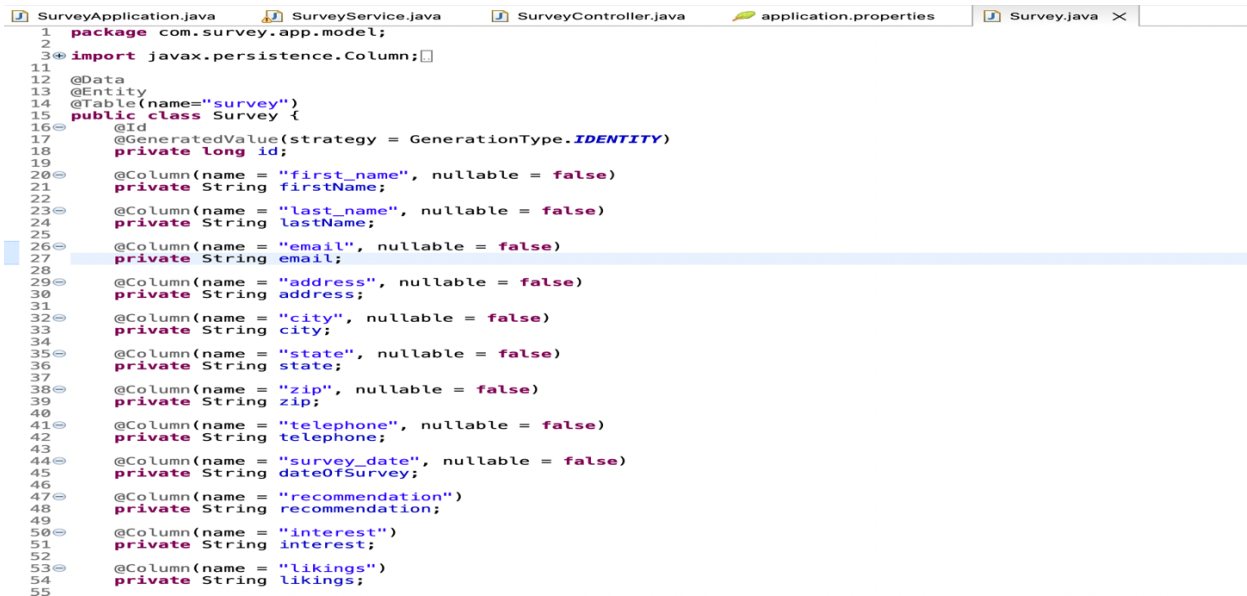
```
SurveyApplication.java  SurveyService.java  SurveyController.java  application.properties  Survey.java
1  package com.survey.app.controller;
2
3  import java.util.List;
16
17  @RestController
18  @RequestMapping("/api/v1.0")
19  public class SurveyController {
20
21      private SurveyInterface surveyService;
22
23  public SurveyController(SurveyInterface surveyInterface) {
24      super();
25      this.surveyService = surveyInterface;
26  }
27
28  @PostMapping("/surveys")
29  public ResponseEntity<Survey> saveSurvey(@RequestBody Survey survey) {
30      return new ResponseEntity<Survey>(surveyService.saveSurvey(survey), HttpStatus.CREATED);
31  }
32
33
34  @GetMapping("/surveys")
35  public List<Survey> getAllSurveys(){
36      return surveyService.getAllSurveys();
37  }
38
39
```

- GET `http://{baseurl}/api/v1.0/surveys`
- POST `http://{baseurl}/api/v1.0/surveys`

Body

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@mail.com",
  "address": "9155 Walnut St",
  "city": "Bentonville",
  "state": "AR",
  "zip": "22031",
  "telephone": "8188884114",
  "dateOfSurvey": "04-28-2022",
  "recommendation": "Very Likely",
  "interest": "Friends",
  "likings": "Students",
  "comment": "Good University",
}
```

- The Data Model looks like the following fields

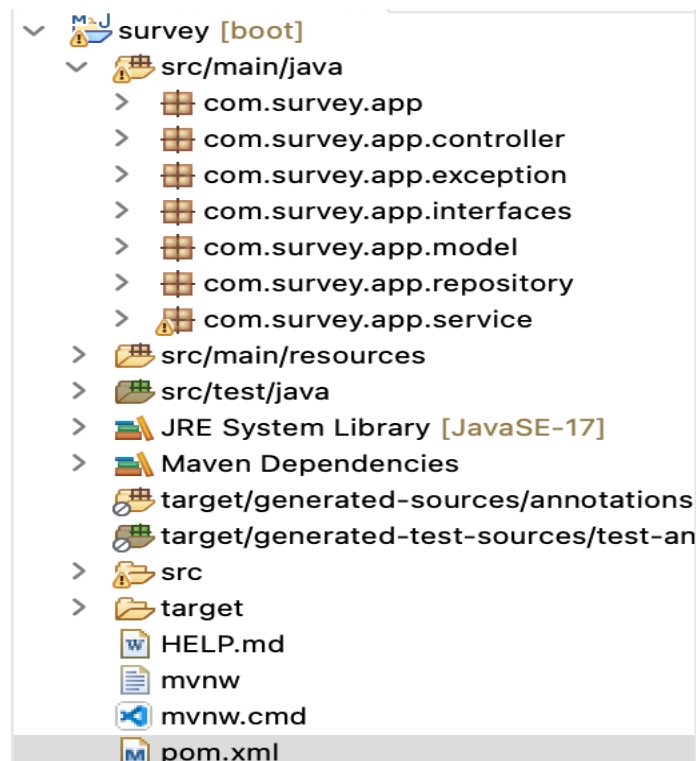


```

1 package com.survey.app.model;
2
3 import javax.persistence.Column;
4
5 @Data
6 @Entity
7 @Table(name="survey")
8 public class Survey {
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private long id;
12
13    @Column(name = "first_name", nullable = false)
14    private String firstName;
15
16    @Column(name = "last_name", nullable = false)
17    private String lastName;
18
19    @Column(name = "email", nullable = false)
20    private String email;
21
22    @Column(name = "address", nullable = false)
23    private String address;
24
25    @Column(name = "city", nullable = false)
26    private String city;
27
28    @Column(name = "state", nullable = false)
29    private String state;
30
31    @Column(name = "zip", nullable = false)
32    private String zip;
33
34    @Column(name = "telephone", nullable = false)
35    private String telephone;
36
37    @Column(name = "survey_date", nullable = false)
38    private String dateOfSurvey;
39
40    @Column(name = "recommendation")
41    private String recommendation;
42
43    @Column(name = "interest")
44    private String interest;
45
46    @Column(name = "likings")
47    private String likings;
48
49 }

```

- Write the API and create MySQL database using AWS RDS
- The API is written following the standard design pattern. First we have a controller layer which calls the service layer and finally the service layer calls the database layer to store data in database.
- The API project structure looks like below



```

survey [boot]
├── src/main/java
│   ├── com.survey.app
│   ├── com.survey.app.controller
│   ├── com.survey.app.exception
│   ├── com.survey.app.interfaces
│   ├── com.survey.app.model
│   ├── com.survey.app.repository
│   └── com.survey.app.service
├── src/main/resources
├── src/test/java
├── JRE System Library [JavaSE-17]
├── Maven Dependencies
├── target/generated-sources/annotations
├── target/generated-test-sources/test-annotations
├── src
├── target
├── HELP.md
├── mvnw
├── mvnw.cmd
└── pom.xml

```

- Package for running the application: survey.app
 - Package for controller: survey.app.controller
 - Package if any exception is thrown: survey.app.exception
 - Package for specifying contract to be followed by service layer: survey.app.interfaces
 - Package for data model: survey.app.model
 - Package for JPA and Hibernate configuration: survey.app.repository
 - Package for service layer: survey.app.service
- The JPA and Hibernate helps in auto creation of table and its column in database.
 - We hosted our database on AWS RDS.
 - Hosting database on RDS is comparatively easy just follow the step by step process as mentioned in the AWS documentation.
 - Copy the database endpoint, port number, username and password
 - Using JDBC connect to database and write the following lines in application.properties

```

1 spring.datasource.url=jdbc:mysql://survey-database.cxltcu4iezvy.us-east-1.rds.amazonaws.com:3306/survey
2 spring.datasource.username=#####
3 spring.datasource.password=#####
4
5 # Hibernate properties
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
7
8 # Create, Create-Drop
9 spring.jpa.hibernate.ddl-auto=update

```

- The api can be tested using Postman
- GET API

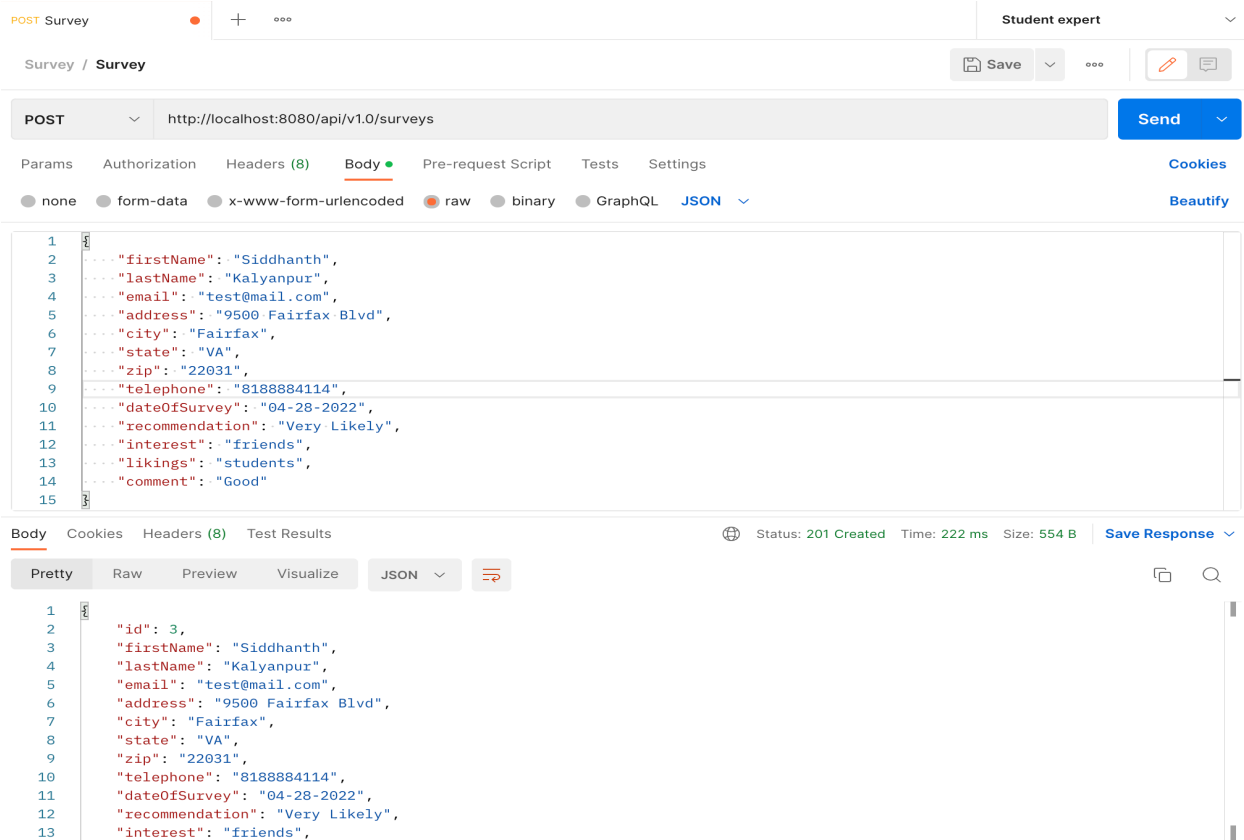
The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/v1.0/surveys`. The response status is 200 OK, with a time of 80 ms and a size of 867 B. The response body is a JSON array of two survey objects.

```

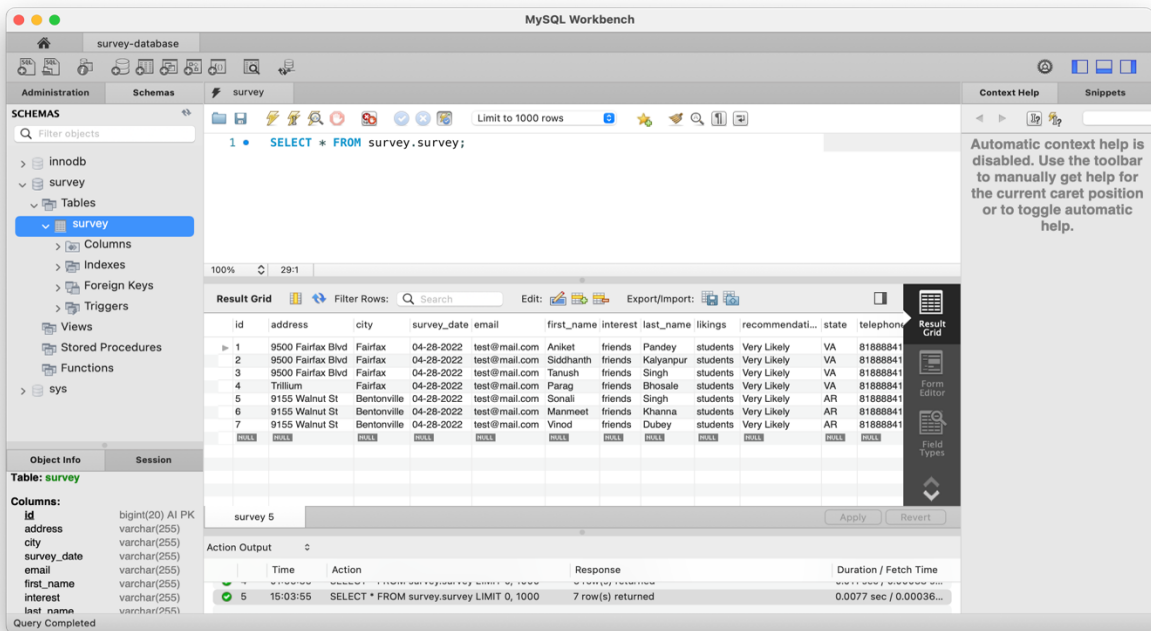
{
  "id": 1,
  "firstName": "ANIKET",
  "lastName": "PANDEY",
  "email": "aniket414@gmail.com",
  "address": "9500 Fairfax Blvd",
  "city": "Fairfax",
  "state": "Virginia",
  "zip": "22031",
  "telephone": "8188214004",
  "dateOfSurvey": "2022-11-09",
  "recommendation": "Likely",
  "interest": "Others",
  "likings": "Location,Campus",
  "comment": null
},
{
  "id": 2,
  "firstName": "ANIKET",
  "lastName": "PANDEY",
  "email": "aniket414@gmail.com",
  "address": "9500 Fairfax Blvd",
  "city": "Fairfax",

```

- POST API



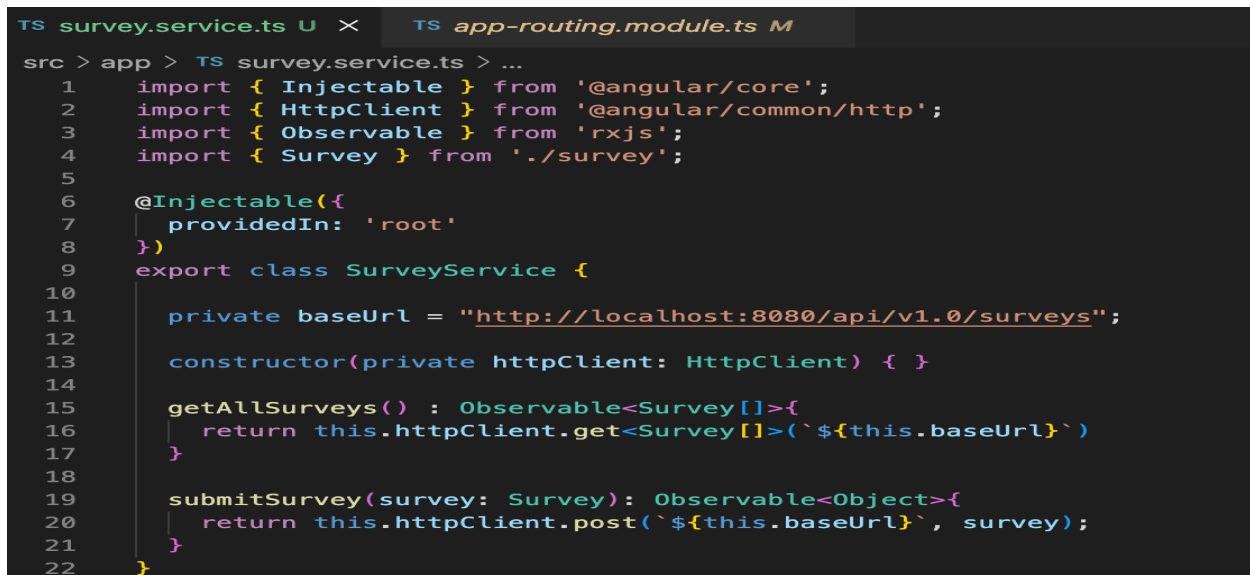
- The Database record can be viewed using MySQLWorkbench



- Now right click on the pom.xml file and inside the Goal on the next step write “clean install” and then continue with the JAR packaging. The JAR file will be present inside target folder by default.

Step 2: Create Angular application

- Create a service call and make GET and POST calls to the api using HttpClient
- The url should be the url of the spring-boot application api on which the api is hosted. Check the image below with baseUrl.

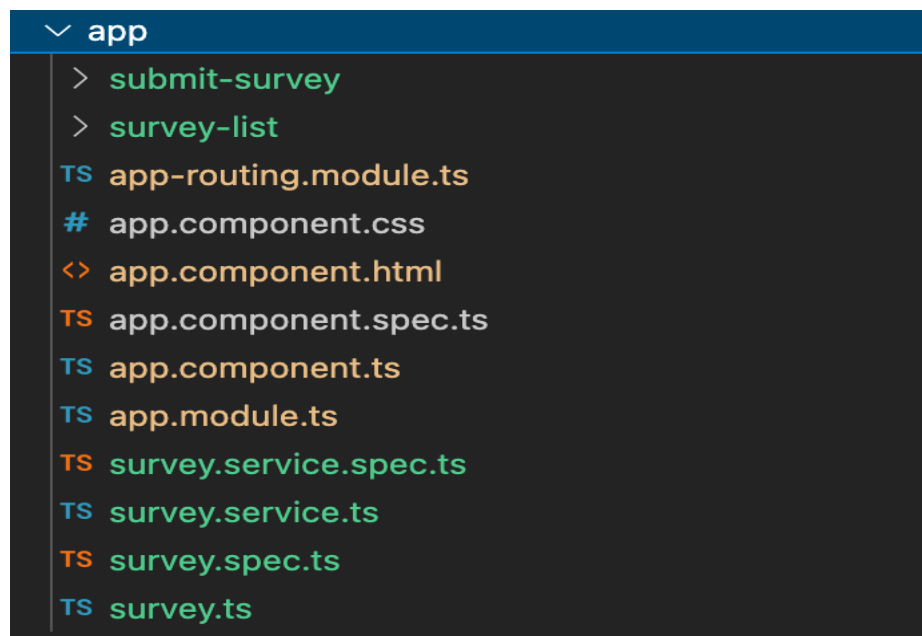


```

TS survey.service.ts U × TS app-routing.module.ts M
src > app > TS survey.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4  import { Survey } from './survey';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class SurveyService {
10
11    private baseUrl = "http://localhost:8080/api/v1.0/surveys";
12
13    constructor(private httpClient: HttpClient) { }
14
15    getAllSurveys() : Observable<Survey[]>{
16      return this.httpClient.get<Survey[]>(`${this.baseUrl}`)
17    }
18
19    submitSurvey(survey: Survey): Observable<Object>{
20      return this.httpClient.post(`${this.baseUrl}`, survey);
21    }
22  }

```

- Following is the walk through of Angular project



```

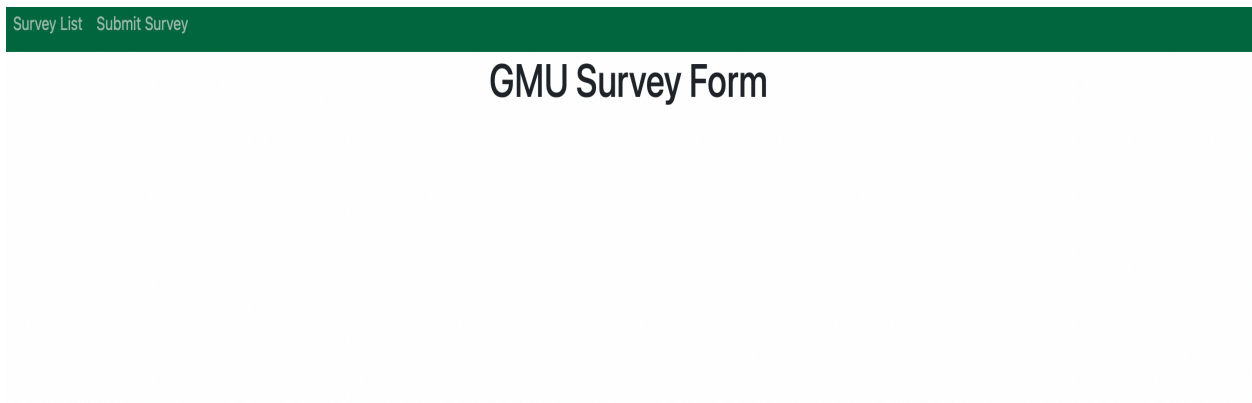
▼ app
  > submit-survey
  > survey-list
  TS app-routing.module.ts
  # app.component.css
  <> app.component.html
  TS app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
  TS survey.service.spec.ts
  TS survey.service.ts
  TS survey.spec.ts
  TS survey.ts

```

- Class for data binding: survey.ts
- Service for API calls: survey.service.ts
- Submit form component: submit-survey
- Get list of all the surveys: survey-list

Step 3: Host API and UI

- First host the API by running you application as spring-boot app.
- Second step is to run the UI by using command `ng serve`.
- Now open your browser and go to the URL: <http://localhost:4200>
- You should see a page as below.



- This is the homepage it has two navigation options one to view list of surveys done till date and the other is for submitting new survey

GMU Survey Form												
List of Survey												
First Name	Last Name	Email	Address	City	State	Zip	Telephone	Date	Recommendation	Interest	Likings	Comment
ANIKET	PANDEY	aniket414@gmail.com	9500 Fairfax Blvd	Fairfax	Virginia	22031	8188214004	2022-11-09	Likely	Others	Location,Campus	
ANIKET	PANDEY	aniket414@gmail.com	9500 Fairfax Blvd	Fairfax	Virginia	22031	8188214004	2022-11-14	Very Likely	Internet	Campus,Atmosphere	Yo
Siddhanth	Kalyanpur	test@mail.com	9500 Fairfax Blvd	Fairfax	VA	22031	8188884114	04-28-2022	Very Likely	friends	students	Good

- The above image displays list of surveys

GMU Survey Form

New Survey

First Name *

Last Name *

Email *

Street Address *

City *

State *

Zip *

Telephone *

Date of Survey *
mm/dd/yyyy

What do you like about campus?

☐ Student

☐ Location

☐ Campus

☐ Atmosphere

☐ Dorm Rooms

☐ Sports

How did you become interested in George Mason University?

☐ Friends

☐ Television

☐ Internet

☐ Others

Would you recommend George Mason University to other prospective students?

Additional Comment

Submit

Reset

Cancel

- The above image displays the survey form with all the fields.
- As seen below after the form is submitted a toaster with acknowledgement is displayed.

What do you like about campus?

☐ Student

☐ Location

☒ Campus

☐ Atmosphere

☐ Dorm Rooms

☐ Sports

How did you become interested in George Mason University?

☐ Friends

☒ Television

☐ Internet

☐ Others

Would you recommend George Mason University to other prospective students?

Likely

Additional Comment

Awesome

Submit

Thanks You! Your survey has been successfully submitted.

Reset

Cancel

- The UI calls the API can also be verified from the browser inspect.

Survey List

Submit Survey

GMU Survey Form

List of Survey

First Name	Last Name	Email	Address	City	State	Zip	Telephone	Date	Recommendation	Initials
ANIKET	PANDEY	aniket414@gmail.com	9500 Fairfax Blvd	Fairfax	Virginia	22031	8188214004	2022-11-09	Likely	C
ANIKET	PANDEY	aniket414@gmail.com	9500 Fairfax Blvd	Fairfax	Virginia	22031	8188214004	2022-11-14	Very Likely	Ir
Siddhant	Kalyanpur	test@mail.com	9500 Fairfax Blvd	Fairfax	VA	22031	8188884114	04-28-2022	Very Likely	fr
ANIKET	PANDEY	aniket414@gmail.com	9500 Fairfax Blvd	Fairfax	Virginia	22031	8188214004	2022-11-08	Likely	T

