SWE-642
HW3-Assignment

Team Details:

Keerthi Ramireddy	**G-01450961**	**Implemented Backend services and have done the setup with database**
Mary Rithika Reddy Gade	**G-01460702**	**Worked on documentation and checked and added the validations to survey form**
Karthik Reddy Musku	**G-01446785**	**Implemented the frontend using angular and connected the frontend and backend**
Meghana Tummala	**G-01448137**	**Worked on documentation and validated backend survey apis using postman.**

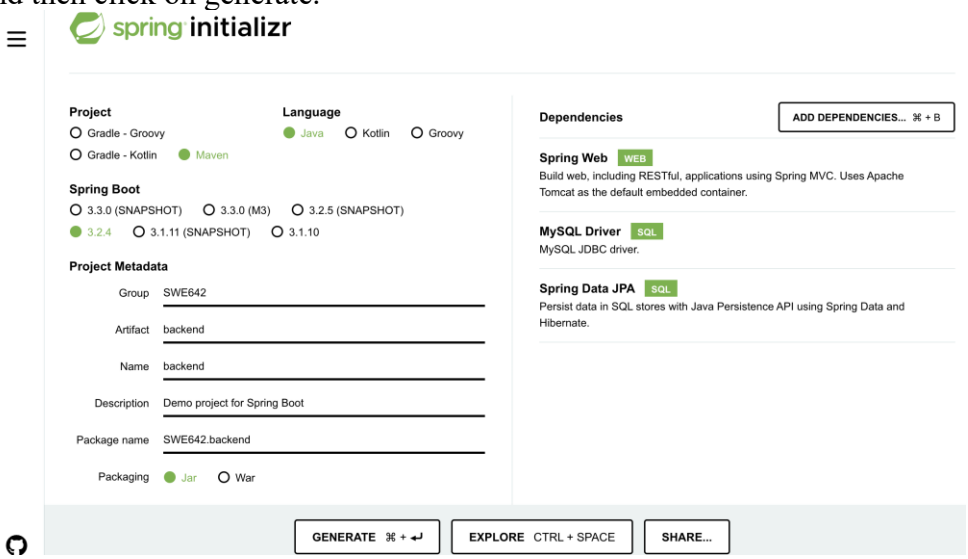
Backend: Java Springboot
Frontend: Angular
Database: SQL Server

Prerequisites:
Installed JAVA JDK 17 in local machine and used eclipse IDE for Springboot(downloaded eclipse EE IDE)
Installed node (npm package manager) and angular using npm
Installed SQL Server

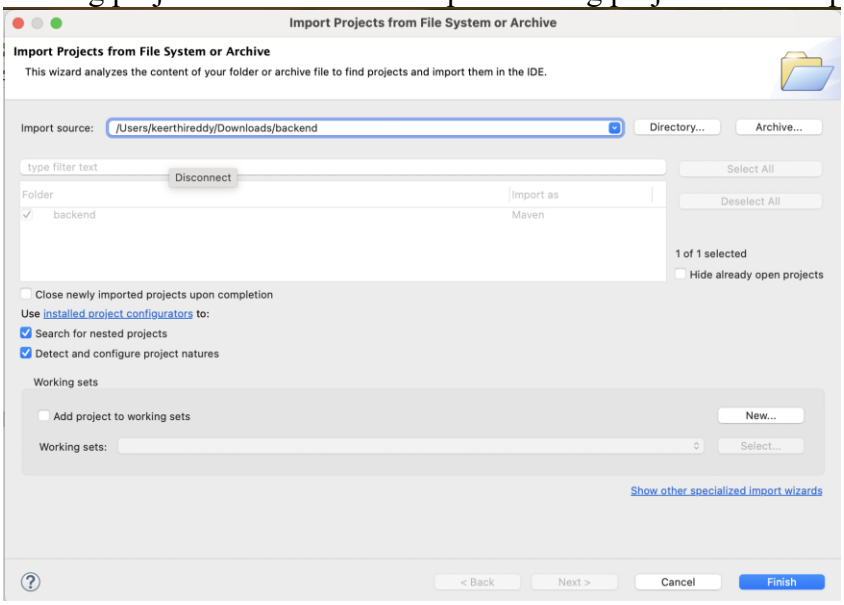
Backend Setup:

1. After installation of required prerequisites. Navigate to https://start.spring.io/

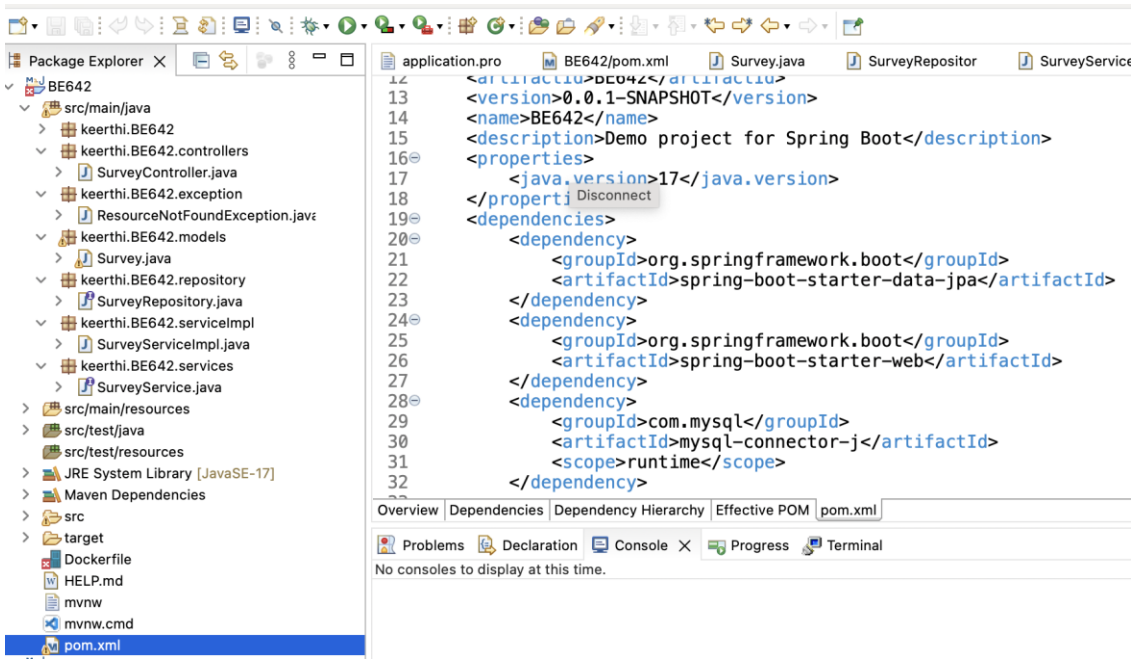
2. Provide the name and artifact details and choose the details like below. Add dependencies and then click on generate.



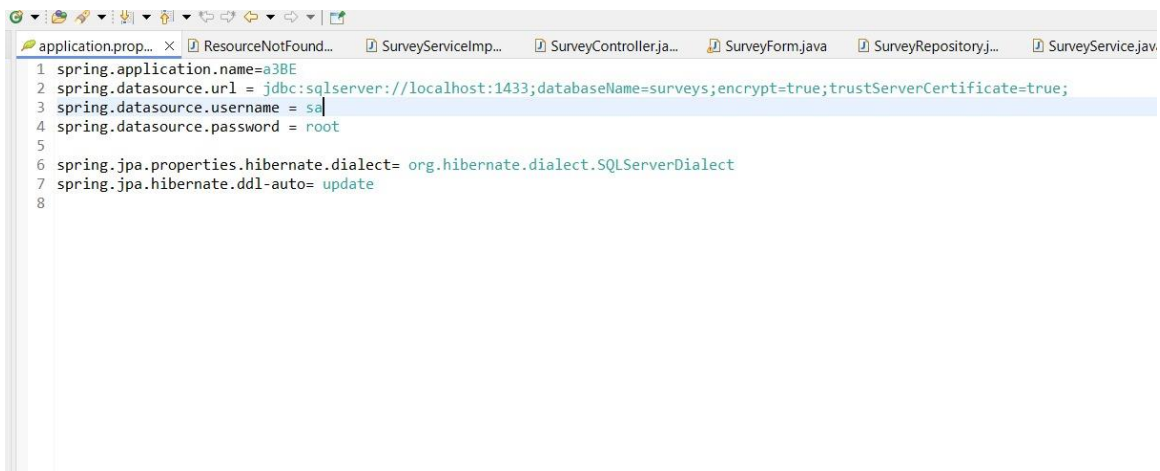
3. A zip file will be downloaded which can be extracted and loaded on to the eclipse IDE as existing project.Click on file and open existing projects.After importing click on finish.



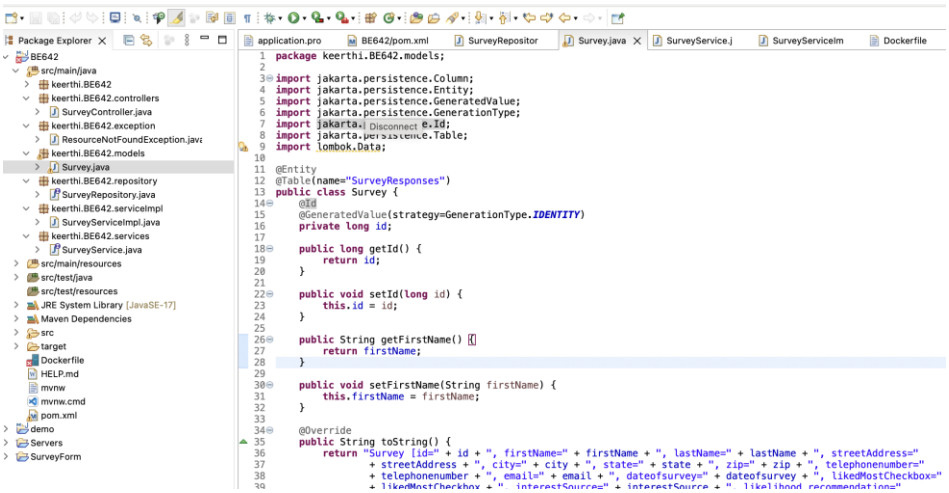
4. Create packages for controllers,services,model,exception,Service Implementation and repository.



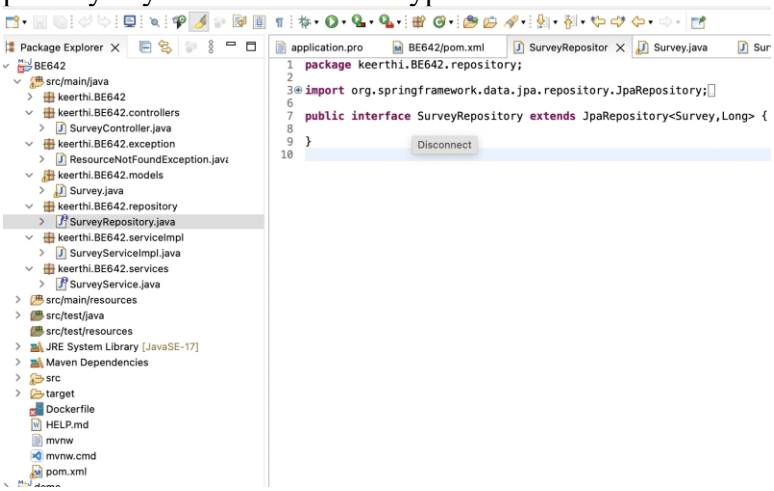
5. Set the database properties like DB connection string,DB NAME,DB USERNAME,DB PASSWORD in application.properties file



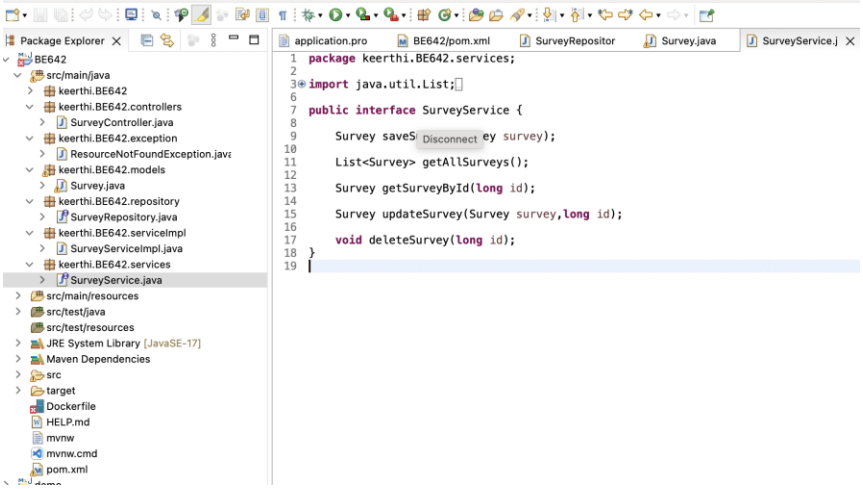
6. Create a class for model under model package and define the columns and table name Annotate the class with tags @Entity and define table name with tag @Table



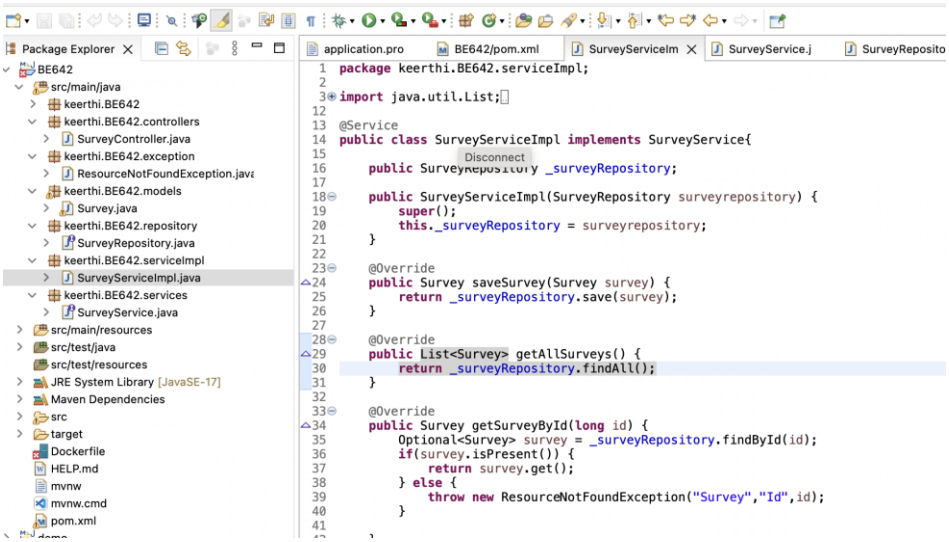
7. Create an interface for repository and extend it with Jpa Repository with model and primary key for the model datatypes.



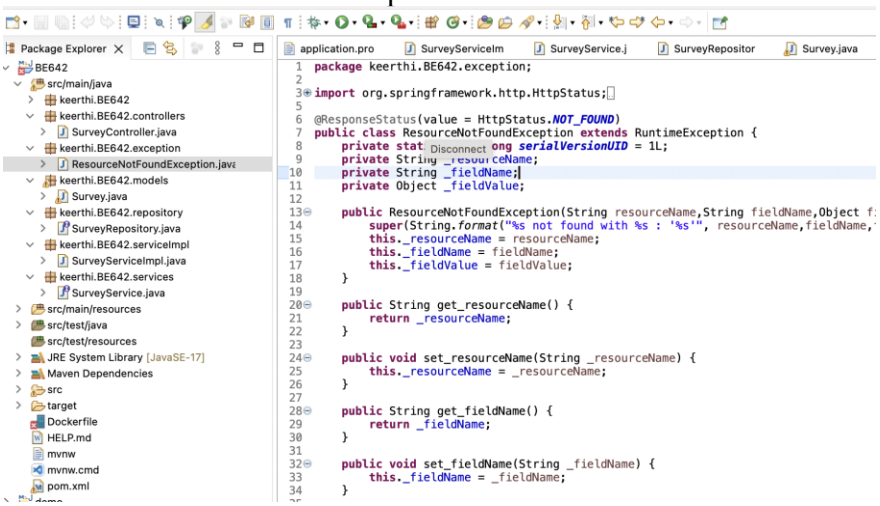
8. Create an interface for service and declare the methods that the apis will use.



9. Implement the methods that we declared in the interface of services in the service implementation class. Also annotate with the @Service tag for the implementation class.



10. Create an exception class by defining a custom exception as a resource not found if invoked a resource that is not present.



11. Now create a controller by annotating it with @RestController and implement all the methods for GET, POST, PUT, DELETE

```java
import java.util.List;

@CrossOrigin("*")
@RestController
@RequestMapping("/survey")
public class SurveyController {

    @Autowired
    private SurveyService surveyService;

    @GetMapping(value = "/findAll")
    public  ResponseEntity<List<SurveyForm>> findAll() {
        return ResponseEntity.ok(surveyService.allSurveys());
    }

    @PostMapping(value = "/saveSurvey")
    @ResponseStatus(HttpStatus.CREATED)
    public ResponseEntity<SurveyForm> create(@RequestBody SurveyForm resource) {
        System.out.println(resource);
        return  ResponseEntity.ok(surveyService.saveSurvey(resource));
    }

    @PutMapping(value = "/editSurvey/{id}")
    public ResponseEntity<SurveyForm> updateRequest(@RequestBody SurveyForm resource, @PathVariable Long id) {
        System.out.println(resource);

        return  ResponseEntity.ok(surveyService.updateSurvey(resource, id));
    }

    @DeleteMapping(value = "/deleteSurvey/{id}")
    public void deleteSurvey(@PathVariable Long id) {
        System.out.println(id);

       surveyService.deleteSurvey(id);
    }
```

Note: I have used @CrossOrigin so that the CORs error is avoided when interacting from Angular Application to Springboot.

This Spring boot application can be run as Java Application

FRONTEND (ANGULAR)

1. Install angular using npm with the command ***npm install -g @angular/cli***
2. Check whether it is installed with **ng –version**
3. Then create a new project using **ng new angular-survey –standalone=true**
4. After creating the project, check whether app.module.ts is created.
5. Run the application using **ng serve –watch**
6. **The application runs on localhost:4200**
7. We have created few specific components such as **home, view-surveys, survey, page-not-found**
   I have created these components using **ng g c <component-name>**
8. I have created a model folder where I have created a survey-form model.

```
export class SurveyForm {
    id:number = 0;
    firstName: string='';
    lastName: string='';
    streetAddress: string ='';
    city: string='';
    state: string='';
    zip: string='';
    telephoneNumber: string='';
    email: string='';
    dateOfSurvey: Date | null = null;
    likedMost: any;
    interestedMost: string='';
    likelihood: string='';
    additionalComments: string='';

    constructor(){}
}
```

9. Then I created an environments folder where I created an environment.ts to store the URL.

```
export const environment = {
  production: false,
  BASE_URL: "http://localhost:8080/survey"
};
```

10. I have created a service called survey.service.ts using **ng g s survey**
11. I have installed bootstrap using
    npm install bootstrap
    Then I have added

```
"styles": [
  "@angular/material/prebuilt-themes/indigo-pink.css",
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": []
```

12. I have installed sweetalert and angular-material in the application using
    npm install sweetalert2
    npm install angular-material
13. After installing angular-material, I have added the relevant import in the import array in app.module.ts

Steps involved in app.module.ts

1. I added **FormsModule , HttpClientModule** in the imports array
2. I have added **SurveyService** in the **providers** array.

```
],
imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule,
  FormsModule,
  MatDialogModule,
  MatFormFieldModule,
  MatSelectModule,
  MatInputModule,
  MatDatepickerModule,
  MatNativeDateModule,
  MatButtonModule,
  MatCheckboxModule,
  MatRadioModule
],
providers: [
  provideClientHydration(),
  SurveyService,
  provideAnimationsAsync()
],
bootstrap: [AppComponent]
)
```

3. Import the relevant modules.

App.routing.module.ts

1. I have added the relevant routes

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { SurveyComponent } from './survey/survey.component';
import { InfoComponent } from './info/info.component';
import { ViewSurveysComponent } from './view-surveys/view-surveys.component';
import { HomeComponent } from './home/home.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

const routes: Routes = [
  {path:'home', component: HomeComponent},
  {path:'view-surveys', component:ViewSurveysComponent},
  {path:'add-survey', component:SurveyComponent},
  {path: '', redirectTo: 'home', pathMatch: 'full'},
  {path:'**', component:PageNotFoundComponent}
];

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

Setting up survey.service.ts

- I have imported **HttpClient** in the service file by injecting it into the constructor.
- I have added fetchSurveys(), createSurvey(),deleteSurvey(),updateSurvey() methods

```
import { environment } from '../environments/environment';

const API_URL = `${environment.BASE_URL}`;

@Injectable({
  providedIn: 'root'
})
export class SurveyService {

  constructor(private http: HttpClient) { }

  tabnine: test | explain | document | ask
  fetchSurveys(): Observable<SurveyForm[]> {
    return this.http.get<SurveyForm[]>(API_URL+"/findAll").pipe(catchError((error: HttpErrorResponse) => {
      return throwError(error.message || 'Server error');
  }));
}

  tabnine: test | explain | document | ask
  createSurvey(survey:SurveyForm): Observable<any>{
    return this.http.post(API_URL+"/saveSurvey", survey);
  }

  tabnine: test | explain | document | ask
  deleteSurvey(id: number): Observable<any>{
    return this.http.delete(API_URL+"/deleteSurvey/"+id);
  }

  tabnine: test | explain | document | ask
  updateSurvey(survey:SurveyForm): Observable<any>{
    return this.http.post(API_URL+"/editSurvey", survey);
  }
```

Implementing Components:

- **view-surveys**
  - This component contains the html and ts code for displaying the survey details.
  - In view-survey.component.html, I have implemented a table to display the data.
  - In view-survey.component.ts, I have injected the following in the constructor,
    - **SurveyService** --- to call the get function
    - **ActivatedRoute** --- provides access to information about a route associated with a component
    - **Router** ---- Managing the navigation
    - **MatDialog** --- Injecting the modal feature for updating the record
  - We have implemented the listSurveys(), updateRecord() and deleteRecord() in the view-surveys.component.ts

```html
<table class="table table-dark">
    <thead>
        <tr class="text-center text-light">
            <th scope="col" class="text-danger">Delete</th>
            <th scope="col" class="text-primary">Edit</th>
            <th scope="col">FirstName</th>
            <th scope="col">LastName</th>
            <th scope="col">StreetAddress</th>
            <th scope="col">City</th>
            <th scope="col">State</th>
            <th scope="col">Zip</th>
            <th scope="col">TelephoneNumber</th>
            <th scope="col">Email</th>
            <th scope="col">DateOfSurvey</th>
            <th scope="col">LikedMost</th>
            <th scope="col">InterestedMost</th>
            <th scope="col">Likelihood</th>
            <th scope="col">AdditionalComments</th>

        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let survey of surveys">
            <td><button type="button" (click)="deleteRecord(survey)" class="btn btn-danger">Delete</button></td>
            <td><button type="button" (click)="updateRecord(survey)"  class="btn btn-primary">Edit</button></td>
            <td>{{survey.firstName}}</td>
            <td>{{survey.lastName}}</td>
            <td>{{survey.streetAddress}}</td>
            <td>{{survey.city}}</td>
            <td>{{survey.state}}</td>
            <td>{{survey.zip}}</td>
            <td>{{survey.telephoneNumber}}</td>
            <td>{{survey.email}}</td>
            <td>{{survey.dateOfSurvey | date}}</td>
            <td>{{survey.likedMost}}</td>
            <td>{{survey.interestedMost}}</td>
```

```
this.router.events.subscribe((evt) => {
    if (evt instanceof NavigationEnd) {
        this.router.navigated = false;
        this.listSurveys();
        window.scrollTo(0, 0);
    }
});
}

tabnine: test | explain | document | ask
ngOnInit(): void {

}

tabnine: test | explain | document | ask
listSurveys() {

  this.surveyService.fetchSurveys()
  .subscribe(surveys => this.surveys = surveys);

  console.log(this.surveys);
}

tabnine: test | explain | document | ask
updateRecord(survey: SurveyForm) {
  console.log(survey);
  let d=this.dialog.open(ModalComponent,{height:'600px',width:'600px',data:{survey}});
    d.afterClosed().subscribe(data => {

      console.log(data);
      window.location.reload()

    })
```

- **Survey**
  - This component is used to implement the survey form and post the form values to the database.
  - In .html, we have implemented the survey form using bootstrap.
  - In .ts, we have to post the values after validation for required fields.
  - Inject **SurveyService, Route** in the survey.ts

```
<div class="container-fluid">
    <div class="row">
      <div class="col-2"></div>
      <div class="col-8">
        <h1 class="div2 text-center">Survey</h1>
        <br>
        <form #f="ngForm" style="background-color: #006633; color: #FFCC33;" class="container p-5 border" id="surveyForm"
          name="surveyForm" autocomplete="on" (ngSubmit) = "onSubmit()">

          <div class="form-group">
            <label for="firstName">First Name<small style="display:inline; color: red;">*</small></label>
            <input type="text" name="firstName" class="form-control" id="firstName" placeholder="Enter firstName" [(ngModel)]
              autofocus>
              <div *ngIf="firstNameInput.invalid && (firstNameInput.dirty || firstNameInput.touched)" class="text-danger">
                First name is required.
              </div>
          </div>

          <div class="form-group">
            <label for="lastName">Last Name<small style="display:inline; color: red;">*</small></label>
            <input type="text" name="lastName" class="form-control" id="lastName" placeholder="Enter lastName" [(ngModel)] =
            <div *ngIf="lastName.invalid && (lastName.dirty || lastName.touched)" class="text-danger">
              Last name is required.
            </div>
          </div>


          <div class="form-group">
            <label for="email">Email address<small style="display:inline; color: red;">*</small></label>
            <input type="text" class="form-control" id="email" name="email" [(ngModel)] = survey.email placeholder="Enter ema
            <div *ngIf="email.invalid && (email.dirty || email.touched)" class="text-danger">
              Email is required.
            </div>
          </div>
          <div class="form-group">
            <label for="phone">Phone<small style="display:inline; color: red;">*</small></label>
```

```
constructor(private surveyService:SurveyService, private router: Router) { }

tabnine: test | explain | document | ask
saveSurvey(){
  this.surveyService.createSurvey(this.survey).subscribe( data =>{
    console.log(data);
    this.goToSurveyList();
  },
  error => console.log(error));
}

tabnine: test | explain | document | ask
goToSurveyList(){
  this.router.navigate(['/view-surveys']);
}

tabnine: test | explain | document | ask
validatePhoneNumber() {
  console.log(this.survey.telephoneNumber);

  const phoneNumberPattern = /^\d{10}$/;
  if (phoneNumberPattern.test(this.survey.telephoneNumber)) {
    return null;
  } else {
    return { invalidPhoneNumber: true };
  }
}

tabnine: test | explain | document | ask
displayErrorMessage(input: any) {
  if (input.control.hasError('required')) {
    console.log('Input is required');
  } else {
    input.control.markAsTouched();
```

- **Modal**
  - This is implemented using angular-material, we made an internal call in view-surveys component using **MatDialog**.

```
<h2 mat-dialog-title class="text-center">Update Survey </h2>

<form #form="ngForm" (ngSubmit)="onSubmit()">

  <div>

    <mat-form-field appearance="outline" class="full-width">
      <mat-label>First Name</mat-label>

      <input matInput name="firstName" id="firstName" #firstName="ngModel" [(ngModel)]="survey.firstName" required>

      <!-- <mat-error *ngIf="form.controls['firstName'].hasError('required')">First Name is required</mat-error> -->
    </mat-form-field>

  </div>
  <mat-form-field class="full-width" appearance="outline">
    <mat-label>Last Name</mat-label>
    <input matInput name="lastName" [(ngModel)]="survey.lastName" required>
    <!-- <mat-error *ngIf="form.controls['lastName'].hasError('required')">Last Name is required</mat-error> -->
  </mat-form-field>
  <mat-form-field class="full-width" appearance="outline">
    <mat-label>Email</mat-label>
    <input matInput name="email" [(ngModel)]="survey.email" type="email" required>

    <!-- <mat-error *ngIf="form.controls['email'].hasError('required')">Email is required</mat-error> -->
  </mat-form-field>
  <mat-form-field class="full-width" appearance="outline">
    <mat-label>Phone Number</mat-label>
    <input matInput name="phoneNumber" [(ngModel)]="survey.telephoneNumber" required>

    <!-- <mat-error *ngIf="form.controls['phoneNumber'].hasError('required')">Telephone Number is required</mat-error> -->
  </mat-form-field>

  <mat-form-field class="full-width" appearance="outline">
    <mat-label>Street Address</mat-label>
    <input matInput name="streetAddress" [(ngModel)]="survey.city" required>
```

```
export class ModalComponent implements OnInit {

  (property) ModalComponent.interested: string[]
  interested: string[] = ['Friends', 'Television', 'Internet', 'Other'];
  likedMost: any;

  likelihoodValues = ['Very Likely', 'Likely', 'Not Likely'];
  likelihood: string = '';

  constructor(@Inject(MAT_DIALOG_DATA) data: any, private surveyService: SurveyService) {
    console.log(data.survey);
    this.survey = data.survey;
    this.survey.likedMost = JSON.parse(this.survey.likedMost);
    this.likedMost = this.survey.likedMost;
  }
  tabnine: test | explain | document | ask
  ngOnInit(): void {
    console.log(this.survey);

  }

  tabnine: test | explain | document | ask
  onSubmit() {
    this.survey.likedMost = JSON.stringify(this.likedMost);
    console.log(this.survey.likedMost);
    console.log(this.survey);
    this.updateSurvey();
  }

  tabnine: test | explain | document | ask
  updateSurvey() {
    this.surveyService.createSurvey(this.survey).subscribe(data => {
      console.log(data);
    },
```

- **Home**
  - This is the starting page where I have just basic html page to display image

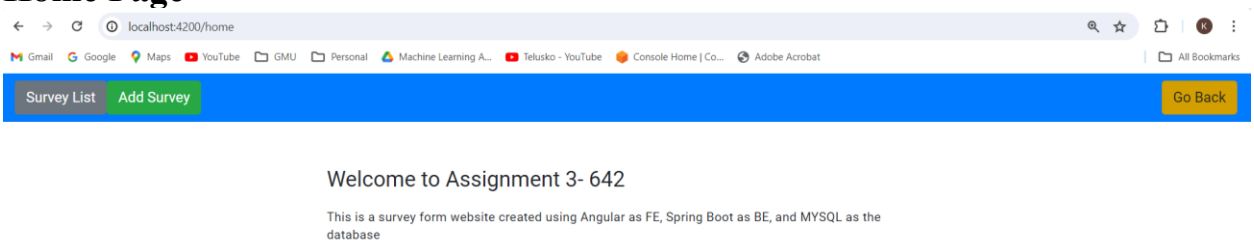
```
export class HomeComponent implements OnInit {
  tabnine: test | explain | document | ask
  ngOnInit(): void {

  }

  constructor(private surveyService:SurveyService,  private router: Router) { }

  tabnine: test | explain | document | ask
  addSurvey(){
    console.log("New survey added");

    this.router.navigate(['/add-survey']);
  }

  tabnine: test | explain | document | ask
  listSurveys() {
    this.router.navigate(['/view-surveys']);
  }

}
```

- **App.component.html**
  - Implemented a navbar to show the three links which states
    - **survey-list** – Displays the list of surveys
    - **add survey** – Add the survey
    - **go-back** --- Reroute to Home component
  - Added a **router-outlet** tag

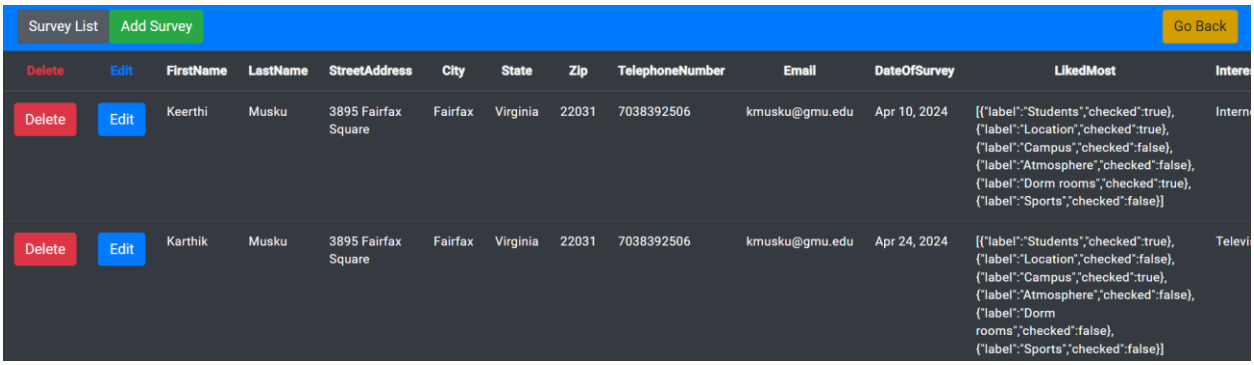
```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
    <ul class = "navbar-nav">
        <li class = "nav-item">
            <button routerLink="view-surveys" routerLinkActive="active"  class="btn btn-secondary" >Survey List</button>
        </li>
        <li class = "nav-item">
            <button routerLink="add-survey" routerLinkActive="active" class="btn btn-success" >Add Survey</button>
        </li>
    </ul>
    <ul class="navbar-nav ml-auto">
        <li class = "nav-item">
            <button routerLink="" routerLinkActive="active" class="btn btn-warning" >Go Back</button>
        </li>
    </ul>
</nav>
<router-outlet></router-outlet>
```

UI Interface

Home Page

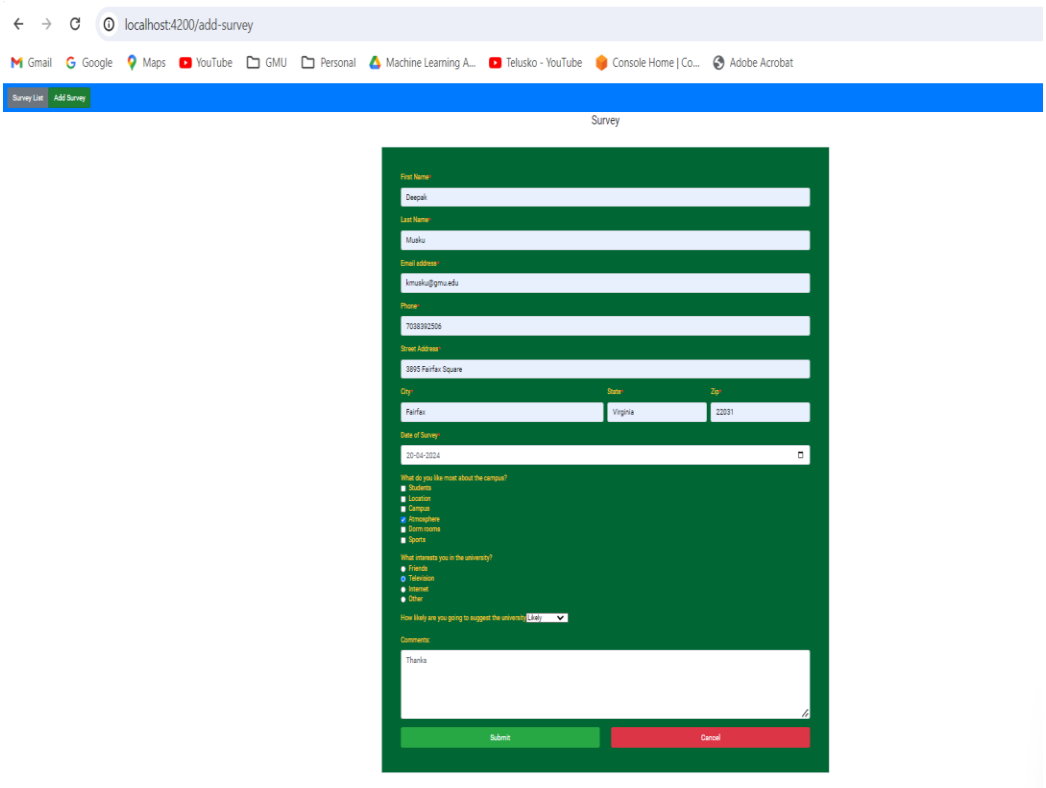


List of Surveys

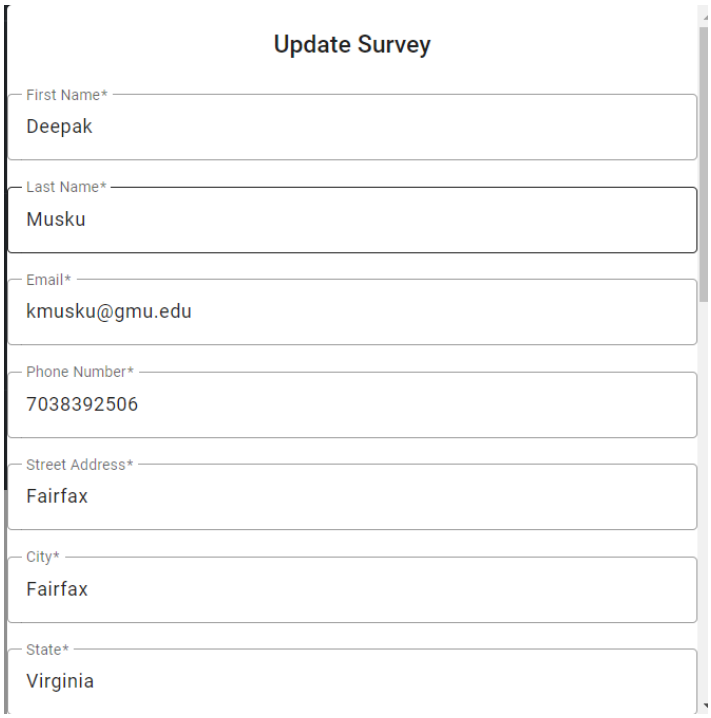


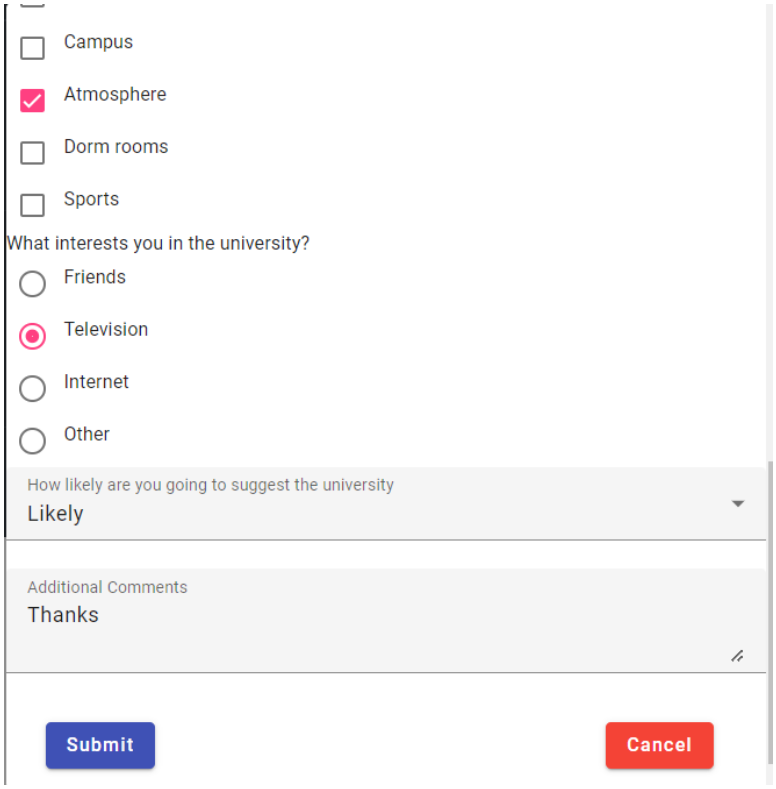
Delete	Edit	FirstName	LastName	StreetAddress	City	State	Zip	TelephoneNumber	Email	DateOfSurvey	LikedMost	Intere
Delete	Edit	Keerthi	Musku	3895 Fairfax Square	Fairfax	Virginia	22031	7038392506	kmusku@gmu.edu	Apr 10, 2024	[{"label":"Students","checked":true},{"label":"Location","checked":true},{"label":"Campus","checked":false},{"label":"Atmosphere","checked":false},{"label":"Dorm rooms","checked":true},{"label":"Sports","checked":false}]	Intern
Delete	Edit	Karthik	Musku	3895 Fairfax Square	Fairfax	Virginia	22031	7038392506	kmusku@gmu.edu	Apr 24, 2024	[{"label":"Students","checked":true},{"label":"Location","checked":false},{"label":"Campus","checked":true},{"label":"Atmosphere","checked":false},{"label":"Dorm rooms","checked":false},{"label":"Sports","checked":false}]	Televi

Add Survey

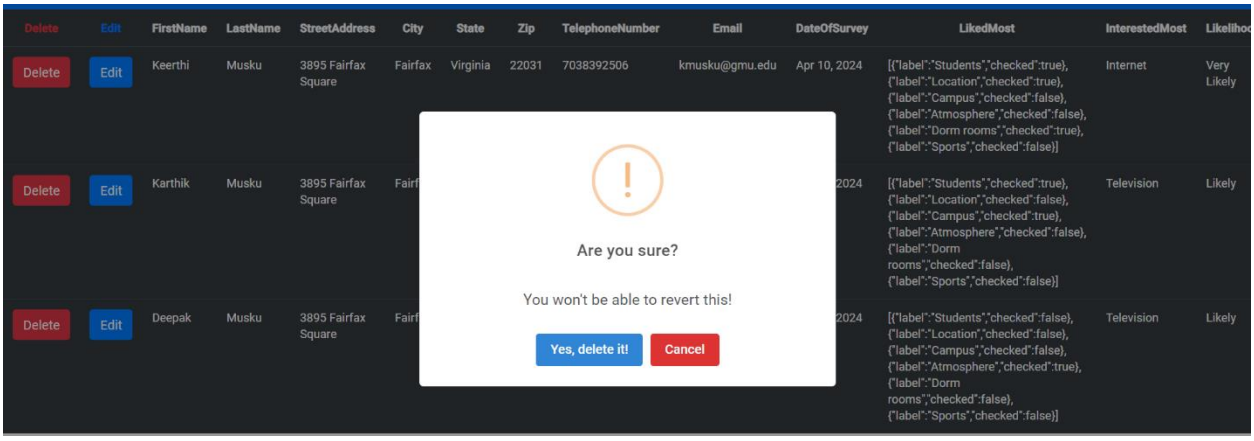


Update Survey





Delete



Delete	Edit	FirstName	LastName	StreetAddress	City	State	Zip	TelephoneNumber	Email	DateOfSurvey	LikedMost	InterestedMost	Likelihoo
Delete	Edit	Keerthi	Musku	3895 Fairfax Square	Fairfax	Virginia	22031	7038392506	kmusku@gmu.edu	Apr 10, 2024	[{"label":"Students","checked":true},{"label":"Location","checked":true},{"label":"Campus","checked":false},{"label":"Atmosphere","checked":false},{"label":"Dorm rooms","checked":true},{"label":"Sports","checked":false}]	Internet	Very Likely
Delete	Edit	Karthik	Musku	3895 Fairfax Square	Fairf					2024	[{"label":"Students","checked":true},{"label":"Location","checked":false},{"label":"Campus","checked":true},{"label":"Atmosphere","checked":false},{"label":"Dorm rooms","checked":false},{"label":"Sports","checked":false}]	Television	Likely
Delete	Edit	Deepak	Musku	3895 Fairfax Square	Fairf					2024	[{"label":"Students","checked":false},{"label":"Location","checked":false},{"label":"Campus","checked":false},{"label":"Atmosphere","checked":true},{"label":"Dorm rooms","checked":false},{"label":"Sports","checked":false}]	Television	Likely