



**STATE UNIVERSITY OF ZANZIBAR (SUZA)**

**COURSE NAME: ADVANCED WEBSITE**

**STUDENT REGISTRATION NUMBER: BITA/6/22/015/TZ**

**STUDENT NAME: SALHA ABDALLA MOHAMED**

**LECTURE NAME: MR MASOUD HAMAD MMANGA**

**REPORT OF THE PROJECT: INCLUSIVE EDUCATION  
WEBSITE**

My Inclusive Education Project deals with people with Special needs, on how the society can learn to communicate with them and help them and how the Society live with them without any problem.

In this website the user register then starts learning through tutorials in the videos which has subtitles then the user can attempt a quiz and see his or her score and the correct answers after the quiz.

## Frontend codes of the project

Starting with home page where it has a page where a user can register and login in the website and navigate to another page.

Here is the HTML 5 codes, JavaScript, CSS and Angular.

```
<nav>  
  <a routerLink="/home">Home</a>  
  <a routerLink="/blind-education">Education for Blind</a>  
  <a routerLink="/deaf-education">Education for Deaf</a>  
</nav>  
<div class="container">  
  <h1>Welcome to Inclusive Education</h1>  
  <div class="form-container">  
    <div class="form-group">  
      <h2>Register</h2>  
      <form id="registerForm">  
        <label for="registerUsername">Username:</label>  
        <input type="text" id="registerUsername" name="username" required>  
        <label for="registerPassword">Password:</label>  
        <input type="password" id="registerPassword" name="password" required>  
        <button type="submit">Register</button>  
      </form>  
    </div>  
    <div class="form-group">  
      <h2>Login</h2>  
      <form id="loginForm">  
        <label for="loginUsername">Username:</label>  
        <input type="text" id="loginUsername" name="username" required>  
        <label for="loginPassword">Password:</label>  
        <input type="password" id="loginPassword" name="password" required>  
        <button type="submit">Login</button>  
      </form>  
    </div>  
  </div>  
</div>  
<div>  
  <p>&copy; 2024 Inclusive Education. All rights reserved.</p>  
</div>
```

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

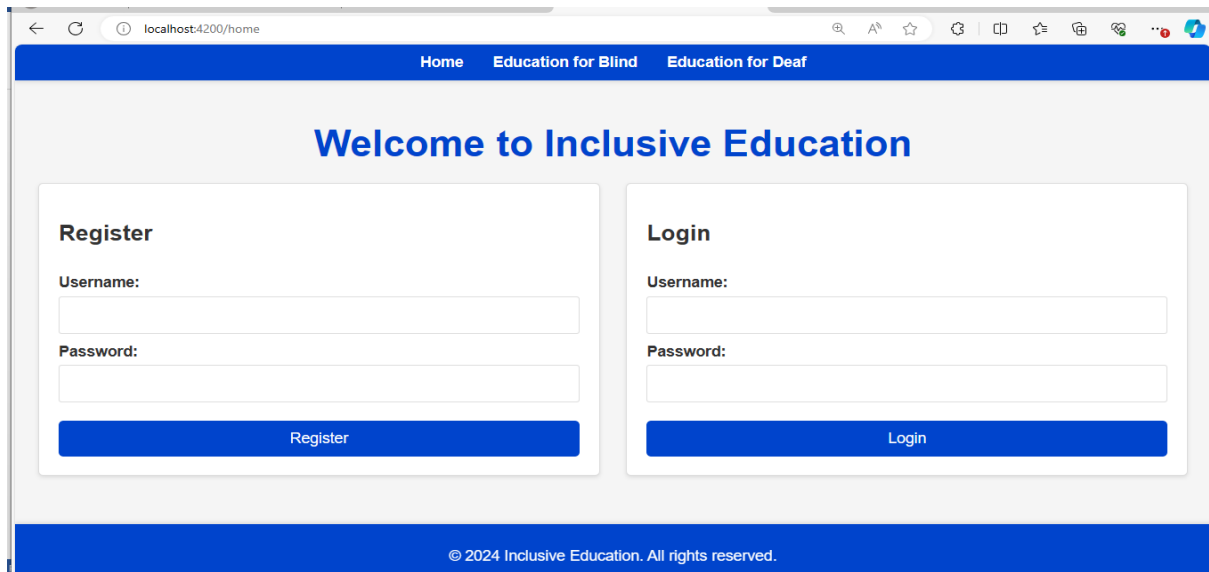
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [HomeComponent]
    });
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

}
```



Here are the codes for the second page where it has a video link and a quiz for people who learn about blind education

```
<nav>
  <a routerLink="/home">Home</a>
  <a routerLink="/blind-education">Education for Blind</a>
  <a routerLink="/deaf-education">Education for Deaf</a>
</nav>
<div class="container">
  <h1>Education for Blind People</h1>
  <a href="C:\Users\kkkkkk\Desktop\Braille.mp4"> a video </a>
  <p>The video explains the basic ways of reading through braille machine </p>
  <div id="blindQuiz" *ngIf="!quizCompleted">
    <h2>Test Your General Understanding about blind education</h2>
    <div id="questionContainer">
      <p>{{ currentQuestion.question }}</p>
      <div *ngFor="let option of currentQuestion.options">
        <label>
          <input type="radio" name="answer" [value]="option" (change)="selectedOption =
option"> {{ option }}
        </label>
      </div>
    </div>
    <button (click)="handleNextQuestion()">Next</button>
  </div>
  <div *ngIf="quizCompleted">
    <h2>Results</h2>
    <p>Score: {{ score }} / {{ questions.length }}</p>
    <h2>Correct Answers</h2>
    <div *ngFor="let question of questions; let i = index">
      <p>{{ question.question }} - {{ correctAnswers[i] }}</p>
    </div>
    <p>{{ resultMessage }}</p>
  </div>
</div>
```

```
</div>
</div>
```

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { BlindEducationComponent } from './blind-education.component';

describe('BlindEducationComponent', () => {
  let component: BlindEducationComponent;
  let fixture: ComponentFixture<BlindEducationComponent>;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [BlindEducationComponent]
    });
    fixture = TestBed.createComponent(BlindEducationComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-blind-education',
  templateUrl: './blind-education.component.html',
  styleUrls: ['./blind-education.component.css']
})
export class BlindEducationComponent {
  selectedOption: string = ""; // Initialize selectedOption
  quizCompleted: boolean = false;
  resultMessage: string = "";
  currentQuestionIndex: number = 0;
  correctAnswers: string[] = [];
  userAnswers: string[] = [];
  score: number = 0;
  questions = [
    {
      question: "What is Braille?",
      options: [
        "A type of sign language",
```

```
    "A system of raised dots used by blind people to read and write",
    "A method of audio learning"
  ],
  correctAnswer: "A system of raised dots used by blind people to read and write"
},
{
  question: "Who invented Braille?",
  options: [
    "Helen Keller",
    "Louis Braille",
    "Thomas Edison"
  ],
  correctAnswer: "Louis Braille"
},
{
  question: "What is the arrangement of dots in a Braille cell?",
  options: [
    "Two columns of three dots",
    "Three columns of two dots",
    "One row of six dots"
  ],
  correctAnswer: "Two columns of three dots"
},
{
  question: "How is capitalization indicated in Braille?",
  options: [
    "By using a dot six before the letter",
    "By using uppercase letters",
    "By underlining the letter"
  ],
  correctAnswer: "By using a dot six before the letter"
},
{
  question: "How are numbers represented in Braille?",
  options: [
    "By using special number symbols",
    "By using the first ten letters of the alphabet with a number sign",
    "By writing the numbers directly"
  ],
  correctAnswer: "By using the first ten letters of the alphabet with a number sign"
},
{
  question: "What is the purpose of a Braille embosser?",
  options: [
    "To produce Braille text",
    "To translate Braille to text",
    "To read Braille out loud"
  ],
  correctAnswer: "To produce Braille text"
}
```

```

    correctAnswer: "To produce Braille text"
  },
  {
    question: "How can Braille be read?",
    options: [
      "By touch",
      "By sight",
      "By hearing"
    ],
    correctAnswer: "By touch"
  },
  {
    question: "What materials are used to produce Braille?",
    options: [
      "Paper, plastic, and electronic devices",
      "Metal, wood, and cloth",
      "Glass, rubber, and leather"
    ],
    correctAnswer: "Paper, plastic, and electronic devices"
  },
  {
    question: "What is a Braille display?",
    options: [
      "A device that allows blind people to read digital text",
      "A screen that shows Braille characters visually",
      "A book that contains Braille text"
    ],
    correctAnswer: "A device that allows blind people to read digital text"
  }
];

get currentQuestion() {
  return this.questions[this.currentQuestionIndex];
}

handleNextQuestion() {
  // Record user's answer
  this.userAnswers.push(this.selectedOption);

  // Check if the selected option is correct
  if (this.selectedOption === this.currentQuestion.correctAnswer) {
    // Increment score for correct answer
    this.score++;
  }

  // Move to the next question or complete the quiz
  if (this.currentQuestionIndex < this.questions.length - 1) {
    this.currentQuestionIndex++;
    this.selectedOption = ""; // Reset selected option for next question
  }
}

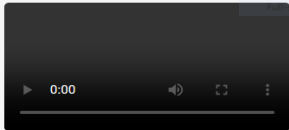
```

```

} else {
  this.quizCompleted = true;
  this.resultMessage = 'You have completed the quiz!';
  this.correctAnswers = this.questions.map(question => question.correctAnswer);
  // Additional logic to calculate the score and set resultMessage based on score
}
}
}

```

## Education for Blind People



The video explains the basic ways of reading through braille machine

### Test Your General Understanding about blind education

What is Braille?

- ☐ A type of sign language
- ☐ A system of raised dots used by blind people to read and write
- ☐ A method of audio learning

[Next](#)

The video explains the basic ways of reading through braille machine

### Results

Score: 4 / 9

#### Correct Answers

What is Braille? - A system of raised dots used by blind people to read and write

Who invented Braille? - Louis Braille

What is the arrangement of dots in a Braille cell? - Two columns of three dots

How is capitalization indicated in Braille? - By using a dot six before the letter

How are numbers represented in Braille? - By using the first ten letters of the alphabet with a number sign

What is the purpose of a Braille embosser? - To produce Braille text

How can Braille be read? - By touch

What materials are used to produce Braille? - Paper, plastic, and electronic devices

What is a Braille display? - A device that allows blind people to read digital text

You have completed the quiz!



The third page is the page where a user can navigate to it and learn about deaf education using the link which provide the basic and essential information about deaf people how they communicate daily using their sign language.

Here are the codes

```
<nav>
  <a routerLink="/home">Home</a>
  <a routerLink="/blind-education">Education for Blind</a>
  <a routerLink="/deaf-education">Education for Deaf</a>
</nav>
<div class="container">
  <h1>Education for Deaf People</h1>
  <a href="C:\Users\kkkkkk\Desktop\Sign language.mp4"> a video for sign language</a>
  <p>The video explains the most common phrases used to communicate with deaf
people</p>
  <div id="deafQuiz" *ngIf="!quizCompleted">
    <h2>Test Your Understanding about American Sign Language</h2>
    <div id="questionContainer">
      <p>{{ currentQuestion.question }}</p>
      <div *ngFor="let option of currentQuestion.options">
        <label>
          <input type="radio" name="answer"> {{ option }}
        </label>
      </div>
    </div>
    <button>Next</button>
  </div>
  <div *ngIf="quizCompleted">
    <h2>Results</h2>
    <p>Score: {{ score }} / {{ questions.length }}</p>
    <h2>Correct Answers</h2>
    <div *ngFor="let question of questions; let i = index">
      <p>{{ question.question }} - {{ correctAnswers[i] }}</p>
    </div>
    <p>{{ resultMessage }}</p>
  </div>
</div>
```

```

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { DeafEducationComponent } from './deaf-education.component';

describe('DeafEducationComponent', () => {
  let component: DeafEducationComponent;
  let fixture: ComponentFixture<DeafEducationComponent>;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [DeafEducationComponent]
    });
    fixture = TestBed.createComponent(DeafEducationComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

```

```

import { Component } from '@angular/core';

@Component({
  selector: 'app-deaf-education',
  templateUrl: './deaf-education.component.html',
  styleUrls: ['./deaf-education.component.css']
})
export class DeafEducationComponent {
  selectedOption: string = ""; // Initialize selectedOption
  quizCompleted: boolean = false;
  resultMessage: string = "";
  currentQuestionIndex: number = 0;
  correctAnswers: string[] = [];
  userAnswers: string[] = [];
  score: number = 0;

  questions = [
    {
      question: "What is fingerspelling?",
      options: [
        "A method of typing",
        "A way to spell words using hand movements",
        "A technique of writing with fingers"
      ],
      correctAnswer: "A way to spell words using hand movements"
    }
  ];

```

```
},
{
  question: "How do you indicate a question in sign language?",
  options: [
    "By raising your voice",
    "By using facial expressions and body language",
    "By writing a question mark in the air"
  ],
  correctAnswer: "By using facial expressions and body language"
},
{
  question: "What is the main purpose of sign language?",
  options: [
    "To communicate with animals",
    "To communicate with people who are deaf or hard of hearing",
    "To perform in theaters"
  ],
  correctAnswer: "To communicate with people who are deaf or hard of hearing"
},
{
  question: "How is sign language different from spoken language?",
  options: [
    "It uses visual gestures and signs",
    "It uses sounds",
    "It uses written words"
  ],
  correctAnswer: "It uses visual gestures and signs"
},
{
  question: "Can sign language be used internationally?",
  options: [
    "Yes, there is one universal sign language",
    "No, each country has its own sign language",
    "Yes, but only in English-speaking countries"
  ],
  correctAnswer: "No, each country has its own sign language"
},
{
  question: "How do deaf people learn sign language?",
  options: [
    "Through listening to audio recordings",
    "By reading books",
    "Through observation and practice"
  ],
  correctAnswer: "Through observation and practice"
},
{
  question: "What is American Sign Language (ASL)?",
```

```

    options: [
      "A written language",
      "A spoken language",
      "A complete, natural language that has the same linguistic properties as spoken languages"
    ],
    correctAnswer: "A complete, natural language that has the same linguistic properties as spoken languages"
  },
  {
    question: "Why is sign language important?",
    options: [
      "It is a secret code",
      "It is a form of entertainment",
      "It provides a means of communication for the deaf and hard of hearing"
    ],
    correctAnswer: "It provides a means of communication for the deaf and hard of hearing"
  }
];

get currentQuestion() {
  return this.questions[this.currentQuestionIndex];
}

handleNextQuestion() {
  // Record user's answer
  this.userAnswers.push(this.selectedOption);

  // Check if the selected option is correct
  if (this.selectedOption === this.currentQuestion.correctAnswer) {
    // Increment score for correct answer
    this.score++;
  }

  // Move to the next question or complete the quiz
  if (this.currentQuestionIndex < this.questions.length - 1) {
    this.currentQuestionIndex++;
    this.selectedOption = ""; // Reset selected option for next question
  } else {
    this.quizCompleted = true;
    this.resultMessage = 'You have completed the quiz!';
    this.correctAnswers = this.questions.map(question => question.correctAnswer);
    // Additional logic to calculate the score and set resultMessage based on score
  }
}
}

```

## Education for Deaf People

[a video for sign language](#)

The video explains the most common phrases used to communicate with deaf people

### Test Your Understanding about American Sign Language

What is fingerspelling?

- ☐ A method of typing
- ☐ A way to spell words using hand movements
- ☐ A technique of writing with fingers

Next

```
<div>
  <h2>User List</h2>
  <ul>
    <li *ngFor="let user of users">
      {{ user.username }} - {{ user.password }}
    </li>
  </ul>
</div>

import { Component, OnInit } from '@angular/core';
import { User } from '../user'; // Adjust the import path as necessary
import { UserService } from '../user.service';

@Component({
  selector: 'app-user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent implements OnInit {
  users: User[] = [];

  constructor(private userService: UserService) { }

  ngOnInit(): void {
    this.userService.getAllUsers().subscribe(data => {
      this.users = data;
    });
  }
}
```

```
import { HttpClientModule } from '@angular/common/http'; // Import HttpClientModule
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';

import { AppComponent } from './app.component';
```

```

import { BlindEducationComponent } from './blind-education/blind-education.component';
import { DeafEducationComponent } from './deaf-education/deaf-education.component';
import { HomeComponent } from './home/home.component';
import { UserListComponent } from './user-list/user-list.component'; // Import
UserListComponent

const appRoutes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'blind-education', component: BlindEducationComponent },
  { path: 'deaf-education', component: DeafEducationComponent },
  { path: 'users', component: UserListComponent }, // Add the route for UserListComponent
  { path: '', redirectTo: '/home', pathMatch: 'full' }
];

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    BlindEducationComponent,
    DeafEducationComponent,
    UserListComponent // Declare the UserListComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule, // Add HttpClientModule here
    RouterModule.forRoot(appRoutes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { User } from './user'; // Ensure this path is correct

@Injectable({
  providedIn: 'root'
})
export class UserService {
  private baseUrl = 'http://localhost:8080/api/users'; // Update with your actual base URL

  constructor(private http: HttpClient) { }

  getAllUsers(): Observable<User[]> {

```

```

return this.http.get<User[]>(this.baseUrl).pipe(
  catchError(this.handleError)
);
}

getUserByUsername(username: string): Observable<User> {
  return this.http.get<User>(`${this.baseUrl}/${username}`).pipe(
    catchError(this.handleError)
  );
}

createUser(user: User): Observable<User> {
  return this.http.post<User>(this.baseUrl, user).pipe(
    catchError(this.handleError)
  );
}

private handleError(error: any): Observable<never> {
  console.error('An error occurred:', error); // Log the error to the console
  return throwError(error); // Return an observable with a user-facing error message
}
}

```

```

export interface User {
  id: number;
  username: string;
  password: string;
  // user properties needed
}

```

```

/* src/styles.css */
body {
  font-family: 'Arial', sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f5f5f5;
  color: #333;
}

nav {
  background-color: #0044cc;
  color: white;
  padding: 10px;
  text-align: center;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

```

```
}

nav a {
  color: white;
  text-decoration: none;
  margin: 0 15px;
  font-weight: bold;
  transition: color 0.3s;
}

nav a:hover {
  color: #ffdd57;
}

.container {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
}

h1 {
  color: #0044cc;
  text-align: center;
  margin-bottom: 20px;
  font-size: 2.5em;
}

.form-container {
  display: flex;
  justify-content: space-around;
  margin-top: 20px;
  gap: 20px;
}

.form-group {
  background-color: #fff;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
  width: 45%;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

form {
  display: flex;
  flex-direction: column;
}
```



```
label {
  margin-top: 10px;
  font-weight: bold;
}

input {
  padding: 10px;
  margin-top: 5px;
  border: 1px solid #ddd;
  border-radius: 3px;
  font-size: 1em;
}

button {
  margin-top: 20px;
  padding: 10px;
  background-color: #0044cc;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1em;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #003399;
}

footer {
  background-color: #0044cc;
  color: white;
  text-align: center;
  padding: 10px 0;
  position: fixed;
  bottom: 0;
  width: 100%;
  box-shadow: 0 -2px 4px rgba(0, 0, 0, 0.1);
}

video {
  display: block;
  margin: 20px auto;
  max-width: 100%;
  border-radius: 5px;
}

#questionContainer {
```

```
margin-top: 20px;
}

#questionContainer p {
  font-size: 18px;
  margin-bottom: 10px;
}

#questionContainer label {
  display: block;
  margin-top: 10px;
}

#nextButton {
  display: block;
  margin: 20px auto;
  padding: 10px 20px;
  background-color: #0044cc;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1em;
  transition: background-color 0.3s;
}

#nextButton:hover {
  background-color: #003399;
}

#blindTestResult, #deafTestResult {
  text-align: center;
  font-size: 1.2em;
  margin-top: 20px;
}
```

Here are the backend codes of the project by using spring-boot with dependencies.

Starting with main class

```
package com.inclusive.education;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EducationApplication {

    public static void main(String[] args) {
        SpringApplication.run(EducationApplication.class, args);
    }

}
```

Repository.

```
package com.inclusive.education.repository;

import com.inclusive.education.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Model

```
package com.inclusive.education.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```

private Long id;

private String username;
private String password;

// Getters and setters

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}
}

```

## Controller

```

package com.inclusive.education.controller;

import com.inclusive.education.model.User;
import com.inclusive.education.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/users")
public class UserController {

```

```

private final UserService userService;

@Autowired
public UserController(UserService userService) {
    this.userService = userService;
}

@GetMapping
public List<User> getAllUsers() {
    return userService.getAllUsers();
}

@GetMapping("/{username}")
public User getUserByUsername(@PathVariable String username) {
    return userService.getUserByUsername(username);
}

@PostMapping
public User createUser(@RequestBody User user) {
    return userService.createUser(user);
}
}

```

## Service

```

package com.inclusive.education.service;

import com.inclusive.education.model.User;
import com.inclusive.education.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {

    private final UserRepository userRepository;

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }
}

```

```

    }

    public User getUserByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    public User createUser(User user) {
        return userRepository.save(user);
    }
}

```

## Configurations

### Security configuration

```

package com.inclusive.education.service;

import com.inclusive.education.model.User;
import com.inclusive.education.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {

    private final UserRepository userRepository;

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public User getUserByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    public User createUser(User user) {
        return userRepository.save(user);
    }
}

```

## Web configuration

```
package com.inclusive.education.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.lang.NonNull;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration // This tells Spring this class is a configuration class
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(@NonNull CorsRegistry registry) {
        registry.addMapping("/*") // Allow all paths
            .allowedOrigins("http://localhost:60605") // Allow requests from this origin
            .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") // Allow
these methods
            .allowedHeaders("*") // Allow all headers
            .allowCredentials(true); // Allow credentials if needed
    }
}
```

## Pom file which has dependencies used in this project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.inclusive</groupId>
    <artifactId>education</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>education</name>
    <description>Inclusve education project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
```

```
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>21</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
```



```
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin>
</plugins>
</build>

</project>
```