# Week 4 Hands-On Exercise

# Skill: Spring REST using Spring Boot3

## 1.Create a Spring Web Project Using Maven

### Scenario:

In this scenario, I created a simple Spring Boot application that exposes a basic REST endpoint.   
The goal was to make sure that when the application is run and the URL http://localhost:9091/hello is accessed in a browser, it returns a plain text message

“Hello World from Spring Boot!”.

# Code:

## HelloController.java

package com.webrest.webapplication;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

*@RestController*

public class HelloController {

*@GetMapping*("/hello")

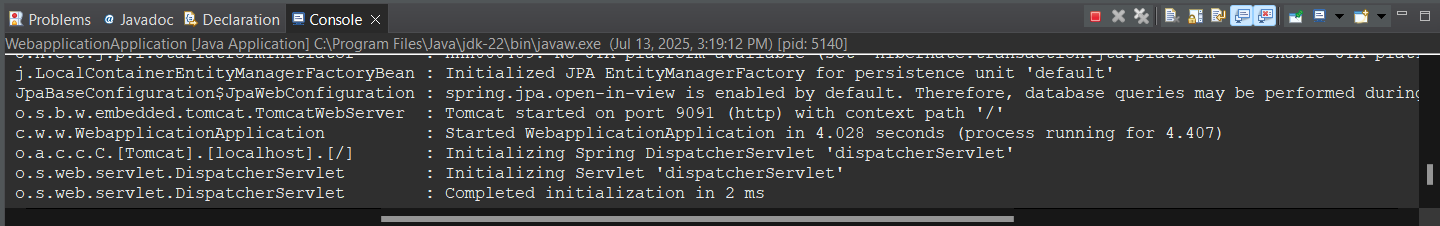
public String sayHello() {

return "Hello World from Spring Boot!";

}

}

# Output:



A black and white rectangular object

AI-generated content may be incorrect.

## 2.Spring Core – Load Country Spring Configuration XML

### Scenario:

To develop a Spring Boot REST application to manage student data.  
In this task it is required to expose a **GET endpoint** that returns the details of a student in **JSON format** when a user accesses the root URL (/).

**Objective:**

* Create a model class Student with fields: id, name, and department.
* Create a controller class that maps to /.
* The controller must return a Student object as a JSON response.

## Code:

## student.java:

package com.webrest.webapplication;

public class Student {

private int id;

private String name;

private String department;

public Student(int id, String name, String department) {

this.id = id;

this.name = name;

this.department = department;

}

public int getId() { return id; }

public String getName() { return name; }

public String getDepartment() { return department; }

}

## StudentController.java:

package com.webrest.webapplication;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

*@RestController*

public class StudentController {

*@GetMapping*("/student")

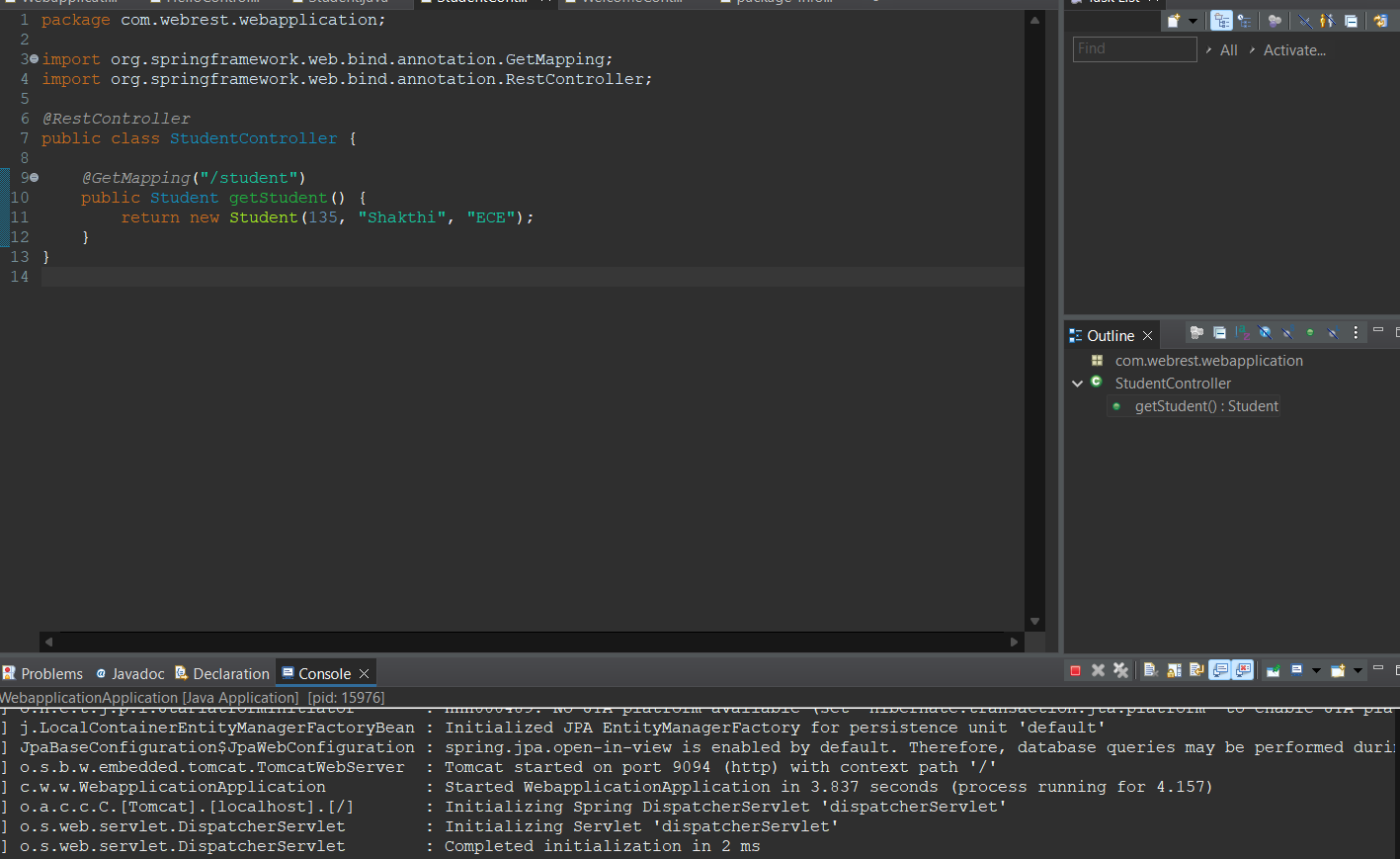
public Student getStudent() {

return new Student(135, "Shakthi", "ECE");

}

}

# Output:



A black and grey rectangular object

AI-generated content may be incorrect.

## 3.Hello World RESTful Web Service

## Scenario:

To build a student management system. We want to send student data using POST method and save in memory and return the same as confirmation.

# Code:

## Student.java:

package com.webapp.webapplication.model;

public class Student {

private int id;

private String name;

private String department;

// Constructors

public Student() {}

public Student(int id, String name, String department) {

this.id = id;

this.name = name;

this.department = department;

}

// Getters and Setters

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

## StudentController.java

package com.webapp.webapplication.controller;

import org.springframework.web.bind.annotation.\*;

import com.webapp.webapplication.model.Student;

@RestController

public class StudentController {

@PostMapping("/student")

public Student addStudent(@RequestBody Student student) {

return student;

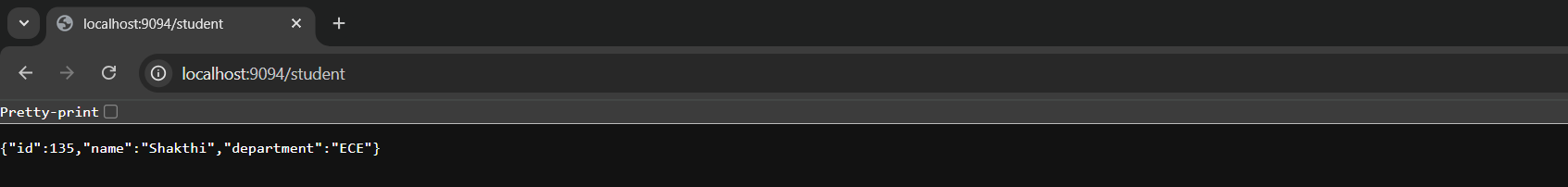
}

}

# Output:

A computer screen with many colorful text

AI-generated content may be incorrect.



# 4. REST – Country Web Service

## Scenario:

To build a RESTful web service that returns **details of a country** using a GET request.

Country.java

package com.webapp.webapplication.model;

public class Country {

private String code;

private String name;

private String capital;

public Country() {}

public Country(String code, String name, String capital) {

this.code = code;

this.name = name;

this.capital = capital;

}

public String getCode() {

return code;

}

public String getName() {

return name;

}

public String getCapital() {

return capital;

}

public void setCode(String code) {

this.code = code;

}

public void setName(String name) {

this.name = name;

}

public void setCapital(String capital) {

this.capital = capital;

}

}

# CountryController.java

package com.webapp.webapplication.controller;

import org.springframework.web.bind.annotation.\*;

import com.webapp.webapplication.model.Country;

@RestController

public class CountryController {

@GetMapping("/country")

public Country getCountry() {

return new Country("IN", "India", "New Delhi");

}

}

# Output:

A computer screen shot of a program

AI-generated content may be incorrect.A black rectangle with white text

AI-generated content may be incorrect.

# 5.REST – Get country based on country code

## Scenario:

To build a RESTful web service that returns **country information** based on a given country **code**. For example, if you send /country/IN, it should return data for **India**.

# Code:

## CountryController.java

package com.webapp.webapplication.controller;

import org.springframework.web.bind.annotation.\*;

import com.webapp.webapplication.model.Country;

@RestController

public class CountryController {

@GetMapping("/country/{code}")

public Country getCountry(@PathVariable String code) {

if (code.equalsIgnoreCase("IN")) {

return new Country("IN", "India", "New Delhi");

} else if (code.equalsIgnoreCase("US")) {

return new Country("US", "United States", "Washington, D.C.");

} else {

return new Country("UNKNOWN", "Unknown Country", "Unknown Capital");

}

}

}

## Country.java

package com.webapp.webapplication.model;

public class Country {

private String code;

private String name;

private String capital;

public Country() {}

public Country(String code, String name, String capital) {

this.code = code;

this.name = name;

this.capital = capital;

}

// Getters and Setters

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public String getName() { return name; }

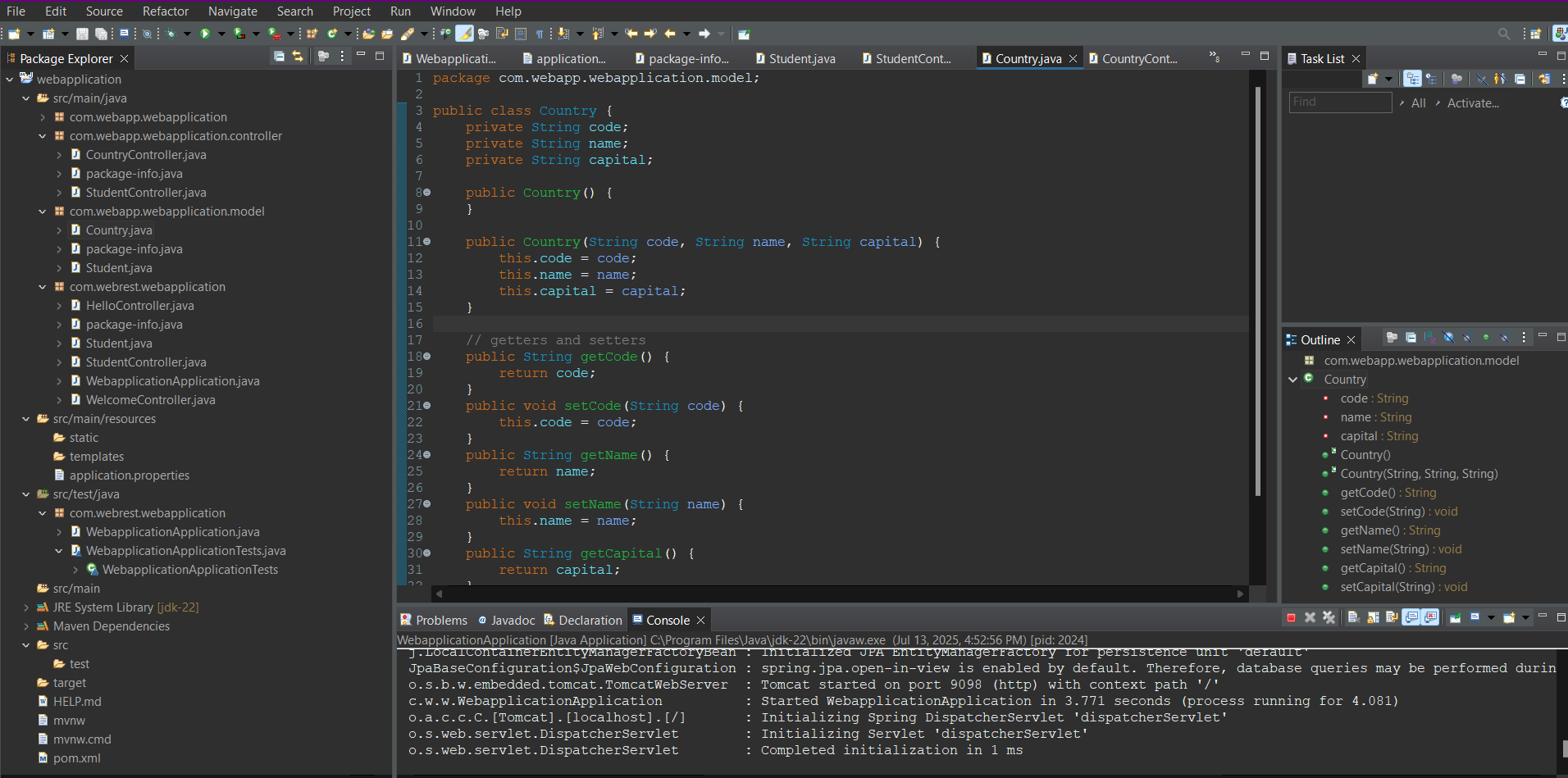
public void setName(String name) { this.name = name; }

public String getCapital() { return capital; }

public void setCapital(String capital) { this.capital = capital; }

}

# Output:



A black screen with white text

AI-generated content may be incorrect.

# 6.JWT – Handson

## Scenario:

> **Registration** (POST /auth/register):  
User sends username & password → password gets **encrypted**, then stored in the database.

> **Login** (POST /auth/login):  
User credentials are verified → if valid, server sends back a **JWT token**

# Code:

## JWT Utility.java

package com.webapp.webapplication.security;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secret = "mySecretKey";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60)) // 1 hour

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

## AuthRequest.java

package com.webapp.webapplication.model;

public class AuthRequest {

private String username;

private String password;

// Getters and Setters

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

## AuthController.java

package com.webapp.webapplication.controller;

import com.webapp.webapplication.model.AuthRequest;

import com.webapp.webapplication.model.User;

import com.webapp.webapplication.repository.UserRepository;

import com.webapp.webapplication.security.JwtUtil;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/auth")

public class AuthController {

@Autowired

private UserRepository userRepo;

@Autowired

private PasswordEncoder passwordEncoder;

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/register")

public String register(@RequestBody User user) {

user.setPassword(passwordEncoder.encode(user.getPassword()));

userRepo.save(user);

return "User registered";

}

@PostMapping("/login")

public String login(@RequestBody AuthRequest authRequest) {

User user = userRepo.findByUsername(authRequest.getUsername())

.orElseThrow(() -> new RuntimeException("User not found"));

if (passwordEncoder.matches(authRequest.getPassword(), user.getPassword())) {

return jwtUtil.generateToken(user.getUsername());

} else {

throw new RuntimeException("Invalid credentials");

}

}

}

## User.java

package com.webapp.webapplication.model;

import jakarta.persistence.\*;

import lombok.\*;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;

import java.util.Collections;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class User implements UserDetails {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

return Collections.emptyList(); // No roles for now

}

@Override public boolean isAccountNonExpired() { return true; }

@Override public boolean isAccountNonLocked() { return true; }

@Override public boolean isCredentialsNonExpired() { return true; }

@Override public boolean isEnabled() { return true; }

}

## UserRepository.java

package com.webapp.webapplication.repository;

import com.webapp.webapplication.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {

Optional<User> findByUsername(String username);

}

## Security Configuration.java

package com.webapp.webapplication.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.core.userdetails.UserDetailsService;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public UserDetailsService userDetailsService() {

return username -> null; // Optional if not using full Spring Security

}

@Bean

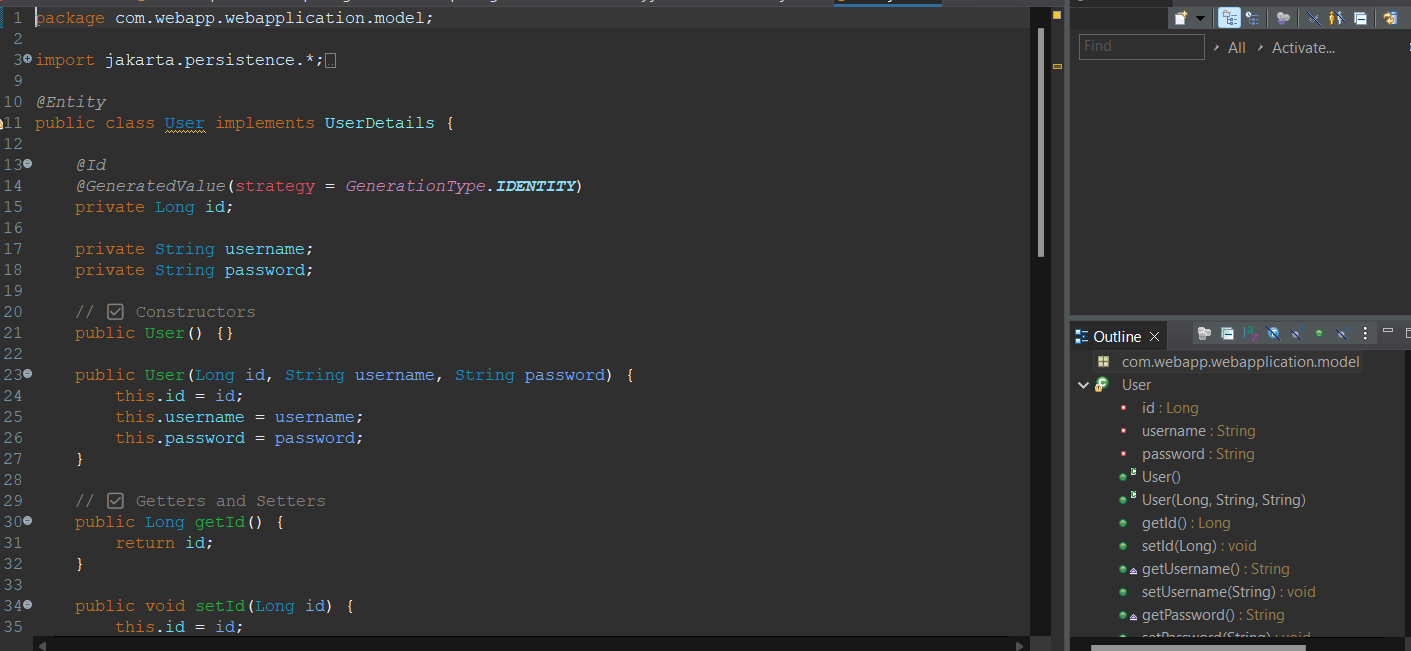
public AuthenticationManager authManager(AuthenticationConfiguration config) throws Exception {

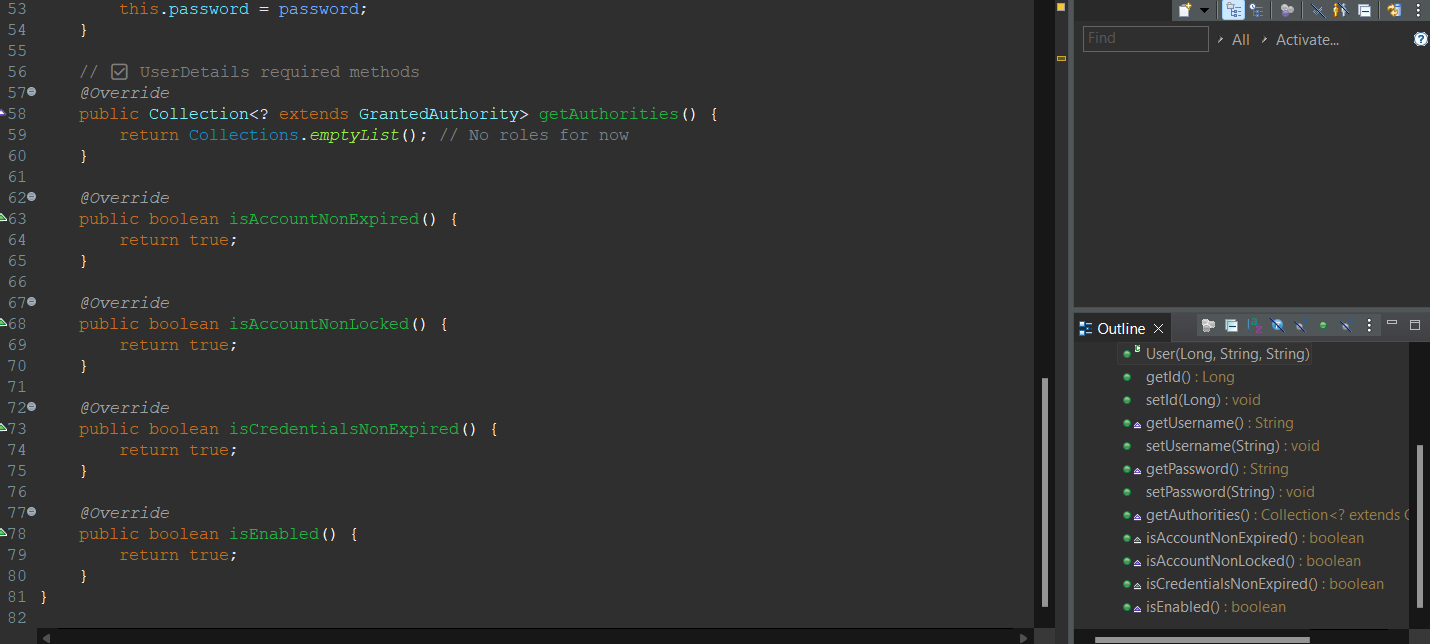
return config.getAuthenticationManager();

}

}

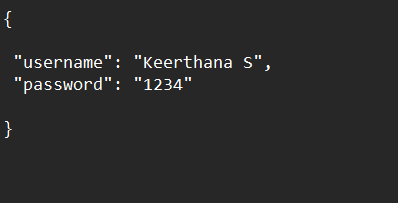
# Output:





A screenshot of a computer program

AI-generated content may be incorrect.



## When JWT Token Missed:

A black background with white text

AI-generated content may be incorrect.