Employee Mnagement System

1.Employee.java package com.example.employee;

```
import jakarta.persistence.*;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;

@Entity
@Table(name = "employees")
public class Employee {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @NotBlank(message = "First Name is required")
   private String firstName;

   @NotBlank(message = "Last Name is required")
   private String lastName;

   @Email(message = "Email should be valid")
   @NotBlank(message = "Email is required")
   private String email;

   @NotBlank(message = "Phone Number is required")
   private String phoneNumber;

   @NotBlank(message = "Position is required")
   private String position;
```

```java
@NotBlank(message = "Salary is required")
private Double salary;

// Getters and Setters
public Long getId() {
   return id;
}

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

public String getFirstName() {
   return firstName;
}

public void setFirstName(String firstName) {
   this.firstName = firstName;
}

public String getLastName() {
   return lastName;
}

public void setLastName(String lastName) {
   this.lastName = lastName;
}

public String getEmail() {
   return email;
}
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }
}
```

2.employeerepository.java

```
package com.example.employee;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
}
```

3.employeeservice.java

```
package com.example.employee;

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

import java.util.List;

@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepository;

    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    public Employee getEmployeeById(Long id) {
        return employeeRepository.findById(id).orElseThrow(() -> new RuntimeException("Employee not found"));
    }

    public Employee addEmployee(Employee employee) {
        return employeeRepository.save(employee);
    }

    public Employee updateEmployee(Long id, Employee employeeDetails) {
        Employee employee = getEmployeeById(id);
        employee.setFirstName(employeeDetails.getFirstName());
        employee.setLastName(employeeDetails.getLastName());
        employee.setEmail(employeeDetails.getEmail());
        employee.setPhoneNumber(employeeDetails.getPhoneNumber());
        employee.setPosition(employeeDetails.getPosition());
        employee.setSalary(employeeDetails.getSalary());
        return employeeRepository.save(employee);
    }

    public void deleteEmployee(Long id) {
        Employee employee = getEmployeeById(id);
        employeeRepository.delete(employee);
    }
}
```

4.employeeController.java

```
package com.example.employee;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/employees")
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;


    @GetMapping
    public List<Employee> getAllEmployees() {
        return employeeService.getAllEmployees();
    }


    @GetMapping("/{id}")
    public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {
        return ResponseEntity.ok(employeeService.getEmployeeById(id));
    }


    @PostMapping
    public Employee addEmployee(@RequestBody Employee employee) {
        return employeeService.addEmployee(employee);
    }


    @PutMapping("/{id}")
    public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee
employeeDetails) {
        return employeeService.updateEmployee(id, employeeDetails);
    }


    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {
        employeeService.deleteEmployee(id);
        return ResponseEntity.noContent().build();
    }
}
```

5.securityconfig.java

```
package com.example.employee;
```

```
import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuild
er;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

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.web.SecurityFilterChain;


import static org.springframework.security.config.Customizer.withDefaults;


@Configuration

@EnableWebSecurity

public class SecurityConfig {


  @SuppressWarnings({ "deprecation" })

        @Bean

  SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

    http

        .csrf(csrf -> csrf.disable()) // Disable CSRF for simplicity; enable it for production

        .authorizeRequests(requests -> requests

            .anyRequest().authenticated()) // Require authentication for all requests

        .httpBasic(withDefaults()); // Enable Basic Authentication

    return http.build();

  }


  @Bean

  PasswordEncoder passwordEncoder() {

    return new BCryptPasswordEncoder(); // Use BCrypt for password encoding
```

```
   }


   @Bean
   void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
      auth.inMemoryAuthentication()
         .withUser("user")
         .password(passwordEncoder().encode("password")) // Password
         .roles("USER"); // Role
   }
}
```

6.Pom.xml

```
<?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>3.3.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>employee</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>employee-managemnet-system</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
```

```xml
<developers>
	<developer/>
</developers>
<scm>
	<connection/>
	<developerConnection/>
	<tag/>
	<url/>
</scm>
<properties>
	<java.version>17</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-web</artifactId>
	</dependency>

	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-devtools</artifactId>
		<scope>runtime</scope>
		<optional>true</optional>
	</dependency>
	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
```

```xml
				<scope>runtime</scope>
			</dependency>
			<dependency>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-starter-test</artifactId>
				<scope>test</scope>
			</dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-boot-starter</artifactId>
      <version>3.0.0</version>
    </dependency>
  </dependencies>
		<build>
			<plugins>
				<plugin>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-maven-plugin</artifactId>
				</plugin>
			</plugins>
		</build>
```

```
</project>
```

7.Application.properties

```
spring.application.name=employee-managemnet-system


spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```