<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.example</groupId>  
 <artifactId>sms\_message</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>sms\_message</name>  
 <description>Demo project for Spring Boot</description>  
  
 <properties>  
 <java.version>17</java.version>  
 <spring-boot.version>3.0.0</spring-boot.version>  
 <hibernate.version>6.1.0.Final</hibernate.version> <!-- Updated Hibernate version -->  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 <version>${spring-boot.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 <version>${spring-boot.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 <version>${spring-boot.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>jakarta.persistence</groupId>  
 <artifactId>jakarta.persistence-api</artifactId>  
 <version>3.0.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.hibernate.orm</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>${hibernate.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId> <!-- Added Spring Security dependency -->  
 <version>${spring-boot.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <version>${spring-boot.version}</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <version>${spring-boot.version}</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

# Use an official OpenJDK as a parent image  
FROM openjdk:17-slim AS *builder*# Set the working directory  
WORKDIR /app  
  
# Install Maven  
RUN apt-get update && apt-get install -y maven  
  
# Copy the pom.xml and source code  
COPY pom.xml .  
COPY src ./src  
  
# Build the application  
RUN mvn clean package -DskipTests  
  
# Stage 2: Create the final image  
FROM openjdk:17-slim  
  
# Copy the jar file from the builder stage  
COPY --from=*builder* /app/target/sms\_message-0.0.1-SNAPSHOT.jar /app/sms\_message.jar  
  
# Set the entry point for the container  
ENTRYPOINT ["java", "-jar", "/app/sms\_message.jar"]

package com.example.sms\_message.controller;  
  
import com.example.sms\_message.model.User;  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/auth")  
public class AuthController {  
  
 @PostMapping("/login")  
 public String login(@RequestBody User user) {  
 // Implement your login logic here  
 // For demonstration, we'll just return a success message  
 return "User " + user.getUsername() + " logged in successfully!";  
 }  
  
 @PostMapping("/register")  
 public String register(@RequestBody User user) {  
 // Implement your registration logic here  
 return "User " + user.getUsername() + " registered successfully!";  
 }  
}

package com.example.sms\_message.controller;  
  
import com.example.sms\_message.model.Message;  
import com.example.sms\_message.service.MessageService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
@Controller  
@RequestMapping("/messages")  
public class MessageController {  
 @Autowired  
 private MessageService messageService;  
  
 @GetMapping("/send")  
 public String sendMessageForm(Model model) {  
 model.addAttribute("message", new Message());  
 return "sendMessage";  
 }  
  
 @PostMapping("/send")  
 public String sendMessage(@ModelAttribute Message message, Model model) {  
 messageService.save(message);  
 model.addAttribute("success", "Message sent successfully!");  
 return "sendMessage";  
 }  
}

package com.example.sms\_message.model;  
  
import jakarta.persistence.\*;  
import java.util.Date;  
  
@Entity  
public class Message {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @ManyToOne  
 @JoinColumn(name = "sender\_id")  
 private User sender;  
  
 @ManyToOne  
 @JoinColumn(name = "receiver\_id")  
 private User receiver;  
  
 private String content;  
  
 @Temporal(TemporalType.*TIMESTAMP*)  
 private Date sentAt = new Date();  
  
 // Getters and Setters  
}

package com.example.sms\_message.model;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String username;  
 private String password;  
  
 // Additional fields (optional)  
 private String email;  
 private String phone;  
  
 // Getters and Setters  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 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;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
}

package com.example.sms\_message.repository;  
  
import com.example.sms\_message.model.Message;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;  
  
public interface MessageRepository extends JpaRepository<Message, Long> {  
 // Custom query to find messages by receiver ID  
 List<Message> findByReceiverId(Long receiverId);  
}

package com.example.sms\_message.repository;  
  
import com.example.sms\_message.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends JpaRepository<User, Long> {  
 User findByUsername(String username);  
}

package com.example.sms\_message.service;  
  
import com.example.sms\_message.model.Message;  
import com.example.sms\_message.repository.MessageRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class MessageService {  
 @Autowired  
 private MessageRepository messageRepository;  
  
 public Message save(Message message) {  
 return messageRepository.save(message);  
 }  
  
 public List<Message> getReceivedMessages(Long receiverId) {  
 return messageRepository.findByReceiverId(receiverId);  
 }  
}

package com.example.sms\_message.service;  
  
import com.example.sms\_message.model.User;  
import com.example.sms\_message.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.util.Optional;  
  
@Service  
public class UserService {  
 @Autowired  
 private UserRepository userRepository;  
  
 public User save(User user) {  
 // Encrypt the user's password before saving  
 user.setPassword(new BCryptPasswordEncoder().encode(user.getPassword()));  
 return userRepository.save(user);  
 }  
  
 public Optional<User> findByUsername(String username) {  
 // Return an Optional wrapping the found User  
 return Optional.*ofNullable*(userRepository.findByUsername(username));  
 }  
}

package com.example.sms\_message;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SmsMessageApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(SmsMessageApplication.class, args);  
 }  
}

body {  
 font-family: Arial, sans-serif;  
 background-color: #f7f7f7;  
 margin: 0;  
 padding: 0;  
}  
  
.container {  
 max-width: 400px;  
 margin: 50px auto;  
 padding: 20px;  
 background: #ffffff;  
 border-radius: 8px;  
 box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);  
}  
  
h2 {  
 text-align: center;  
 margin-bottom: 20px;  
}  
  
input, textarea, button {  
 width: 100%;  
 margin: 10px 0;  
 padding: 10px;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 box-sizing: border-box;  
}  
  
button {  
 background-color: #007BFF;  
 color: #fff;  
 border: none;  
 cursor: pointer;  
}  
  
button:hover {  
 background-color: #0056b3;  
}  
  
.success {  
 color: green;  
 text-align: center;  
}

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Login</title>  
 <link rel="stylesheet" href="static/style.css">  
</head>  
<body>  
<div class="container">  
 <h2>Login</h2>  
 <form th:action="@{/auth/login}" method="post">  
 <input type="text" name="username" placeholder="Username" required />  
 <input type="password" name="password" placeholder="Password" required />  
 <button type="submit">Login</button>  
 </form>  
 <p>Don't have an account? <a th:href="@{/auth/register}">Register here</a>.</p>  
</div>  
</body>  
</html>

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Register</title>  
 <link rel="stylesheet" href="/style.css">  
</head>  
<body>  
<div class="container">  
 <h2>Register</h2>  
 <form th:action="@{/auth/register}" th:object="${user}" method="post">  
 <input type="text" th:field="\*{username}" placeholder="Username" required />  
 <input type="password" th:field="\*{password}" placeholder="Password" required />  
 <input type="email" th:field="\*{email}" placeholder="Email" required />  
 <input type="text" th:field="\*{phone}" placeholder="Phone Number" required />  
 <button type="submit">Register</button>  
 </form>  
 <p>Already have an account? <a th:href="@{/auth/login}">Login here</a>.</p>  
</div>  
</body>  
</html>

# H2 Database Configuration  
spring.datasource.url=jdbc:h2:mem:sms\_message  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update  
  
# Enable H2 Console  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
# Server Configuration  
server.port=8081  
server.error.whitelabel.enabled=false  
  
# Spring Security Configuration (Optional: For simplicity during development)  
spring.security.user.name=admin  
spring.security.user.password=admin123