**USER MODULE**

User.java

package com.Springbootpace.SpringBoot.Entity;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "users")

public class User {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String fullName;

private String email;

private String password;

private String mobile;

private String role; // Add this field

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getFullName() {

return fullName;

}

public void setFullName(String fullName) {

this.fullName = fullName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

// Getters and Setters

}

UserRepository

package com.Springbootpace.SpringBoot.Repositary;

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

import com.Springbootpace.SpringBoot.Entity.User;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {

Optional<User> findByEmail(String email);

}

UserService

package com.Springbootpace.SpringBoot.Service;

import java.util.List;

import com.Springbootpace.SpringBoot.Entity.User;

import com.Springbootpace.SpringBoot.dto.LoginRequest;

import com.Springbootpace.SpringBoot.dto.SignupRequest;

public interface UserService {

User registerUser(SignupRequest request);

boolean validateLogin(LoginRequest request);

User findUserByEmail(String email); // ✅ Add this line

User updateUser(Long id, User updatedUser);

List<User> getAllUsers();

}

UserServiceImpl

package com.Springbootpace.SpringBoot.Service;

import com.Springbootpace.SpringBoot.Entity.User;

import com.Springbootpace.SpringBoot.Repositary.UserRepository;

import com.Springbootpace.SpringBoot.dto.LoginRequest;

import com.Springbootpace.SpringBoot.dto.SignupRequest;

import jakarta.annotation.PostConstruct;

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

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class UserServiceImpl implements UserService {

@Autowired

private UserRepository userRepository;

@Autowired

private PasswordEncoder passwordEncoder;

// @PostConstruct

// public void printEncodedPassword() {

// String raw = "admin123";

// String encoded = passwordEncoder.encode(raw);

// System.out.println("Encoded admin password: " + encoded);

// }

@Override

public User findUserByEmail(String email) {

return userRepository.findByEmail(email).orElse(null);

}

@Override

public User registerUser(SignupRequest request) {

User user = new User();

user.setFullName(request.getFullName());

user.setEmail(request.getEmail());

user.setMobile(request.getMobile());

user.setPassword(passwordEncoder.encode(request.getPassword())); // Encrypt the password

user.setRole("USER");

return userRepository.save(user);

}

@Override

public boolean validateLogin(LoginRequest request) {

Optional<User> userOpt = userRepository.findByEmail(request.getEmail());

if (userOpt.isPresent()) {

User user = userOpt.get();

return passwordEncoder.matches(request.getPassword(), user.getPassword()); // Compare encrypted password

}

return false;

}

@Override

public User updateUser(Long id, User updatedUser) {

Optional<User> optionalUser = userRepository.findById(id);

if (optionalUser.isEmpty()) {

throw new RuntimeException("User not found");

}

User existingUser = optionalUser.get();

existingUser.setFullName(updatedUser.getFullName());

// Optional: allow updating email or mobile if needed

existingUser.setEmail(updatedUser.getEmail());

existingUser.setMobile(updatedUser.getMobile());

return userRepository.save(existingUser);

}

@Override

public List<User> getAllUsers() {

return userRepository.findAll();

}

}

UserController

package com.Springbootpace.SpringBoot.Controller;

import com.Springbootpace.SpringBoot.Entity.User;

import com.Springbootpace.SpringBoot.Service.UserService;

import com.Springbootpace.SpringBoot.dto.LoginRequest;

import com.Springbootpace.SpringBoot.dto.SignupRequest;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

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

import org.springframework.http.\*;

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

import jakarta.servlet.http.HttpSession;

@RestController

@RequestMapping("/api/users")

@CrossOrigin(origins = "\*")

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/all")

public List<User> getAllUsers() {

return userService.getAllUsers();

}

@PostMapping("/signup")

public User signup(@RequestBody SignupRequest request) {

return userService.registerUser(request);

}

@PostMapping("/login")

public ResponseEntity<Map<String, String>> login(@RequestBody LoginRequest request, HttpSession session) {

boolean valid = userService.validateLogin(request);

Map<String, String> response = new HashMap<>();

if (valid) {

session.setAttribute("userEmail", request.getEmail());

User user = userService.findUserByEmail(request.getEmail());

response.put("message", "Login Successful");

response.put("id", String.valueOf(user.getId())); // ✅ Include ID

response.put("email", user.getEmail());

response.put("fullName", user.getFullName());

response.put("mobile", user.getMobile());

response.put("role", user.getRole());

return ResponseEntity.ok(response);

} else {

response.put("message", "Invalid credentials");

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(response);

}

}

@PutMapping("/{id}")

public User updateUser(@PathVariable Long id, @RequestBody User updatedUser) {

return userService.updateUser(id, updatedUser);

}

@GetMapping("/profile")

public ResponseEntity<User> getProfile(HttpSession session) {

String email = (String) session.getAttribute("userEmail");

if (email == null) {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

}

User user = userService.findUserByEmail(email);

return ResponseEntity.ok(user);

}

}