// BankingExpensePlannerApplication.java

package com.banking.expenseplanner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class BankingExpensePlannerApplication {

public static void main(String[] args) {

SpringApplication.*run*(BankingExpensePlannerApplication.class, args);

}

}

//BankingController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.model.Expense;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.ExpenseRepository;

import com.banking.expenseplanner.util.UserUtil;

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

import org.springframework.http.ResponseEntity;

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

import java.util.HashMap;

import java.util.List;

import java.util.Map;

*@RestController*

*@RequestMapping*("/api/banking")

public class BankingController {

*@Autowired*

private ExpenseRepository expenseRepository;

*@Autowired*

private UserUtil userUtil;

*@GetMapping*("/summary")

public ResponseEntity<?> getAccountSummary() {

User user = userUtil.getLoggedInUser();

List<Expense> expenses = expenseRepository.findByUser(user);

double totalIncome = expenses.stream()

.filter(e -> "INCOME".equalsIgnoreCase(e.getType()))

.mapToDouble(Expense::getAmount)

.sum();

double totalExpense = expenses.stream()

.filter(e -> "EXPENSE".equalsIgnoreCase(e.getType()))

.mapToDouble(Expense::getAmount)

.sum();

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

response.put("user", user.getName());

response.put("totalBalance", user.getBalance());

response.put("totalIncome", totalIncome);

response.put("totalExpense", totalExpense);

return ResponseEntity.*ok*(response);

}

*@GetMapping*("/transactions")

public ResponseEntity<?> getTransactionHistory() {

User user = userUtil.getLoggedInUser();

List<Expense> expenses = expenseRepository.findByUser(user);

return ResponseEntity.*ok*(expenses);

}

}

//BudgetController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.model.Budget;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.BudgetRepository;

import com.banking.expenseplanner.util.UserUtil;

import jakarta.validation.Valid;

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

import org.springframework.http.ResponseEntity;

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

*@RestController*

*@RequestMapping*("/api/budget")

public class BudgetController {

*@Autowired*

private BudgetRepository budgetRepository;

*@Autowired*

private UserUtil userUtil;

*@PostMapping*("/add")

public ResponseEntity<?> addBudget(*@Valid* *@RequestBody* Budget budget) {

User user = userUtil.getLoggedInUser();

budget.setUser(user);

budgetRepository.save(budget);

return ResponseEntity.*ok*("Budget added successfully");

}

*@GetMapping*("/my")

public ResponseEntity<?> getUserBudgets() {

User user = userUtil.getLoggedInUser();

return ResponseEntity.*ok*(budgetRepository.findByUser(user));

}

*@GetMapping*("/my/month/{month}/year/{year}")

public ResponseEntity<?> getBudgetsForMonth(*@PathVariable* Integer month, *@PathVariable* Integer year) {

User user = userUtil.getLoggedInUser();

return ResponseEntity.*ok*(budgetRepository.findByUserAndMonthAndYear(user, month, year));

}

}

//ExpenseController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.model.Expense;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.ExpenseRepository;

import com.banking.expenseplanner.repository.UserRepository;

import com.banking.expenseplanner.util.UserUtil;

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

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

import jakarta.validation.Valid;

import java.util.List;

import java.util.Optional;

*@RestController*

*@RequestMapping*("/api/expenses")

public class ExpenseController {

*@Autowired*

private ExpenseRepository expenseRepository;

*@Autowired*

private UserRepository userRepository;

*@Autowired*

private UserUtil userUtil;

// ✅ 1. Create Expense

*@PostMapping*("/add")

public String addExpense(*@Valid* *@RequestBody* Expense expense) {

User user = userUtil.getLoggedInUser();

expense.setUser(user);

expenseRepository.save(expense);

// Update balance based on type

if ("EXPENSE".equalsIgnoreCase(expense.getType())) {

user.setBalance(user.getBalance() - expense.getAmount());

} else if ("INCOME".equalsIgnoreCase(expense.getType())) {

user.setBalance(user.getBalance() + expense.getAmount());

}

userRepository.save(user);

return "Expense added successfully!";

}

// ✅ 2. Get all expenses by logged-in user

*@GetMapping*("/my")

public List<Expense> getExpensesByLoggedUser() {

User user = userUtil.getLoggedInUser();

return expenseRepository.findByUser(user);

}

// ✅ 3. Update expense

*@PutMapping*("/update/{expenseId}")

public String updateExpense(*@PathVariable* Long expenseId, *@Valid* *@RequestBody* Expense updatedExpense) {

User loggedUser = userUtil.getLoggedInUser();

Optional<Expense> existingOpt = expenseRepository.findById(expenseId);

if (existingOpt.isPresent()) {

Expense expense = existingOpt.get();

// Check if the expense belongs to the logged-in user

if (!expense.getUser().getId().equals(loggedUser.getId())) {

return "Access denied: You can only update your own expenses.";

}

// Optional: Adjust user balance before & after update (advanced)

expense.setCategory(updatedExpense.getCategory());

expense.setDescription(updatedExpense.getDescription());

expense.setAmount(updatedExpense.getAmount());

expense.setDate(updatedExpense.getDate());

expense.setType(updatedExpense.getType());

expense.setNote(updatedExpense.getNote());

expense.setTags(updatedExpense.getTags());

expenseRepository.save(expense);

return "Expense updated successfully!";

} else {

return "Expense not found!";

}

}

// ✅ 4. Delete expense

*@DeleteMapping*("/delete/{expenseId}")

public String deleteExpense(*@PathVariable* Long expenseId) {

User loggedUser = userUtil.getLoggedInUser();

Optional<Expense> expenseOpt = expenseRepository.findById(expenseId);

if (expenseOpt.isPresent()) {

Expense expense = expenseOpt.get();

// Check if the expense belongs to the logged-in user

if (!expense.getUser().getId().equals(loggedUser.getId())) {

return "Access denied: You can only delete your own expenses.";

}

// Reverse the balance effect

if ("EXPENSE".equalsIgnoreCase(expense.getType())) {

loggedUser.setBalance(loggedUser.getBalance() + expense.getAmount());

} else if ("INCOME".equalsIgnoreCase(expense.getType())) {

loggedUser.setBalance(loggedUser.getBalance() - expense.getAmount());

}

userRepository.save(loggedUser);

expenseRepository.deleteById(expenseId);

return "Expense deleted successfully!";

} else {

return "Expense not found!";

}

}

}

//ReminderController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.model.Reminder;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.ReminderRepository;

import com.banking.expenseplanner.util.UserUtil;

import jakarta.validation.Valid;

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

import org.springframework.http.ResponseEntity;

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

import java.time.LocalDate;

import java.util.List;

*@RestController*

*@RequestMapping*("/api/reminders")

public class ReminderController {

*@Autowired*

private ReminderRepository reminderRepository;

*@Autowired*

private UserUtil userUtil;

*@PostMapping*("/add")

public ResponseEntity<?> addReminder(*@Valid* *@RequestBody* Reminder reminder) {

User user = userUtil.getLoggedInUser();

reminder.setUser(user);

reminderRepository.save(reminder);

return ResponseEntity.*ok*("Reminder added successfully");

}

*@GetMapping*("/my")

public ResponseEntity<?> getRemindersByUser() {

User user = userUtil.getLoggedInUser();

return ResponseEntity.*ok*(reminderRepository.findByUser(user));

}

*@GetMapping*("/upcoming")

public ResponseEntity<?> getUpcomingReminders() {

User user = userUtil.getLoggedInUser();

LocalDate today = LocalDate.*now*();

LocalDate nextWeek = today.plusDays(7);

List<Reminder> reminders = reminderRepository.findByUserAndDueDateBetween(user, today, nextWeek);

return ResponseEntity.*ok*(reminders);

}

}

//SavingsGoalController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.model.SavingsGoal;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.SavingsGoalRepository;

import com.banking.expenseplanner.util.UserUtil;

import jakarta.validation.Valid;

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

import org.springframework.http.ResponseEntity;

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

import java.util.Optional;

*@RestController*

*@RequestMapping*("/api/savings")

public class SavingsGoalController {

*@Autowired*

private SavingsGoalRepository savingsGoalRepository;

*@Autowired*

private UserUtil userUtil;

*@PostMapping*("/add")

public ResponseEntity<?> addGoal(*@Valid* *@RequestBody* SavingsGoal goal) {

User user = userUtil.getLoggedInUser();

goal.setUser(user);

savingsGoalRepository.save(goal);

return ResponseEntity.*ok*("Savings goal created successfully");

}

*@GetMapping*("/my")

public ResponseEntity<?> getUserGoals() {

User user = userUtil.getLoggedInUser();

return ResponseEntity.*ok*(savingsGoalRepository.findByUser(user));

}

*@PutMapping*("/update/{goalId}")

public ResponseEntity<?> updateGoalProgress(*@PathVariable* Long goalId, *@RequestParam* Double amountToAdd) {

Optional<SavingsGoal> goalOpt = savingsGoalRepository.findById(goalId);

if (goalOpt.isEmpty()) {

return ResponseEntity.*badRequest*().body("Goal not found");

}

SavingsGoal goal = goalOpt.get();

User user = userUtil.getLoggedInUser();

if (!goal.getUser().getId().equals(user.getId())) {

return ResponseEntity.*status*(403).body("Access denied: This goal does not belong to you.");

}

goal.setSavedAmount(goal.getSavedAmount() + amountToAdd);

if (goal.getSavedAmount() >= goal.getTargetAmount()) {

goal.setStatus("COMPLETED");

}

savingsGoalRepository.save(goal);

return ResponseEntity.*ok*("Progress updated");

}

}

//UserController.java

package com.banking.expenseplanner.controller;

import com.banking.expenseplanner.dto.UserDto;

import com.banking.expenseplanner.service.UserService;

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

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

import jakarta.validation.Valid;

*@RestController*

*@RequestMapping*("/api/users")

public class UserController {

*@Autowired*

private UserService userService;

*@PostMapping*("/register")

public String registerUser(*@Valid* *@RequestBody* UserDto userDto) {

return userService.registerUser(userDto);

}

*@PostMapping*("/login")

public String loginUser(*@Valid* *@RequestBody* UserDto userDto) {

return userService.loginUser(userDto);

}

}

//UserDto.java

package com.banking.expenseplanner.dto;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import lombok.Data;

*@Data*

public class UserDto {

*@NotBlank*(message = "Name is required")

private String name;

*@Email*(message = "Invalid email format")

*@NotBlank*(message = "Email is required")

private String email;

*@NotBlank*(message = "Password is required")

private String password;

}

//GlobalExceptionHandler.java

package com.banking.expenseplanner.exception;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.MethodArgumentNotValidException;

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

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

import java.util.HashMap;

import java.util.Map;

*@ControllerAdvice*

public class GlobalExceptionHandler {

*@ExceptionHandler*(MethodArgumentNotValidException.class)

public ResponseEntity<Map<String, String>> handleValidationErrors(MethodArgumentNotValidException ex) {

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

ex.getBindingResult().getFieldErrors().forEach(error -> {

errors.put(error.getField(), error.getDefaultMessage());

});

return new ResponseEntity<>(errors, *HttpStatus*.***BAD\_REQUEST***);

}

// Optional: handle any other exceptions here

}

// Budget.java

package com.banking.expenseplanner.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.Data;

import java.time.LocalDate;

*@Entity*

*@Table*(name = "budgets")

*@Data*

public class Budget {

*@Id*

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

private Long id;

*@ManyToOne*

*@JoinColumn*(name = "user\_id", nullable = false)

private User user;

*@NotBlank*(message = "Category is required")

private String category;

*@NotNull*(message = "Limit amount is required")

private Double limitAmount;

*@NotNull*(message = "Month is required")

private Integer month; // 1–12

*@NotNull*(message = "Year is required")

private Integer year;

private LocalDate createdDate = LocalDate.*now*();

}

// Expense.java

package com.banking.expenseplanner.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.Data;

import java.time.LocalDate;

//import java.util.List;

*@Entity*

*@Table*(name = "expenses")

*@Data*

public class Expense {

*@Id*

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

private Long id;

*@NotBlank*(message = "Category is required")

private String category;

*@NotBlank*(message = "Description is required")

private String description;

*@NotNull*(message = "Amount is required")

private Double amount;

*@NotNull*(message = "Date is required")

private LocalDate date;

private String note;

private String tags; // comma-separated values like "food,lunch"

*@ManyToOne*

*@JoinColumn*(name = "user\_id", nullable = false)

private User user;

*@NotBlank*(message = "Type is required") // Values: "INCOME" or "EXPENSE"

private String type;

}

// Reminder.java (Entity)

package com.banking.expenseplanner.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.Data;

import java.time.LocalDate;

*@Entity*

*@Table*(name = "reminders")

*@Data*

public class Reminder {

*@Id*

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

private Long id;

*@ManyToOne*

*@JoinColumn*(name = "user\_id", nullable = false)

private User user;

*@NotBlank*(message = "Title is required")

private String title;

private String note;

*@NotNull*(message = "Due date is required")

private LocalDate dueDate;

private boolean isRecurring = false;

}

//SavingsGoal.java

package com.banking.expenseplanner.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.NotNull;

import lombok.Data;

*@Entity*

*@Table*(name = "savings\_goals")

*@Data*

public class SavingsGoal {

*@Id*

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

private Long id;

*@ManyToOne*

*@JoinColumn*(name = "user\_id", nullable = false)

private User user;

*@NotBlank*(message = "Goal name is required")

private String goalName;

*@NotNull*(message = "Target amount is required")

private Double targetAmount;

private Double savedAmount = 0.0;

private String status = "IN\_PROGRESS"; // or COMPLETED

}

// User.java

package com.banking.expenseplanner.model;

import jakarta.persistence.\*;

import lombok.\*;

*@Entity*

*@Table*(name = "users")

*@Data* // Generates getters, setters, toString, equals, hashCode using Lombok

*@NoArgsConstructor*

*@AllArgsConstructor*

public class User {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***) // Auto-incremented ID

private Long id;

*@Column*(nullable = false)

private String name;

*@Column*(nullable = false, unique = true)

private String email;

*@Column*(nullable = false)

private String password;

*@Column*(nullable = false)

private Double balance = 10000.0; // Default balance

}

package com.banking.expenseplanner.repository;

import com.banking.expenseplanner.model.Budget;

import com.banking.expenseplanner.model.User;

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

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public interface BudgetRepository extends JpaRepository<Budget, Long> {

List<Budget> findByUser(User user);

List<Budget> findByUserAndMonthAndYear(User user, Integer month, Integer year);

}

package com.banking.expenseplanner.repository;

import com.banking.expenseplanner.model.Expense;

import com.banking.expenseplanner.model.User;

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

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public interface ExpenseRepository extends JpaRepository<Expense, Long> {

// Get all expenses for a specific user

List<Expense> findByUser(User user);

}

package com.banking.expenseplanner.repository;

import com.banking.expenseplanner.model.Reminder;

import com.banking.expenseplanner.model.User;

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

import org.springframework.stereotype.Repository;

import java.time.LocalDate;

import java.util.List;

*@Repository*

public interface ReminderRepository extends JpaRepository<Reminder, Long> {

List<Reminder> findByUser(User user);

List<Reminder> findByUserAndDueDate(User user, LocalDate dueDate);

List<Reminder> findByUserAndDueDateBetween(User user, LocalDate start, LocalDate end);

}

package com.banking.expenseplanner.repository;

import com.banking.expenseplanner.model.SavingsGoal;

import com.banking.expenseplanner.model.User;

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

import org.springframework.stereotype.Repository;

import java.util.List;

*@Repository*

public interface SavingsGoalRepository extends JpaRepository<SavingsGoal, Long> {

List<SavingsGoal> findByUser(User user);

}

package com.banking.expenseplanner.repository;

import com.banking.expenseplanner.model.User;

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

import org.springframework.stereotype.Repository;

import java.util.Optional;

*@Repository*

public interface UserRepository extends JpaRepository<User, Long> {

// Custom method to find user by email

Optional<User> findByEmail(String email);

}

package com.banking.expenseplanner.security;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.UserRepository;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

import java.util.Collections;

import java.util.Optional;

*@Component*

public class JwtRequestFilter extends OncePerRequestFilter {

*@Autowired*

private JwtUtil jwtUtil;

*@Autowired*

private UserRepository userRepository;

*@Override*

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain) throws ServletException, IOException {

final String authHeader = request.getHeader("Authorization");

String email = null;

String jwt = null;

if (authHeader != null && authHeader.startsWith("Bearer ")) {

jwt = authHeader.substring(7);

email = jwtUtil.extractEmail(jwt);

}

if (email != null && SecurityContextHolder.*getContext*().getAuthentication() == null) {

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

if (userOpt.isPresent() && jwtUtil.isTokenValid(jwt, email)) {

UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(

userOpt.get().getEmail(), null, Collections.*emptyList*());

authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.*getContext*().setAuthentication(authToken);

}

}

filterChain.doFilter(request, response);

}

}

package com.banking.expenseplanner.security;

import io.jsonwebtoken.\*;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

*@Component*

public class JwtUtil {

private final String SECRET\_KEY = "nikhil\_very\_secret\_key\_which\_is\_32charlong!";

private final Key key = Keys.*hmacShaKeyFor*(SECRET\_KEY.getBytes());

// Generate JWT with 5 hours expiry

public String generateToken(String email) {

return Jwts.*builder*()

.setSubject(email)

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

.setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 5)) // 5 hours

.signWith(key, *SignatureAlgorithm*.***HS256***)

.compact();

}

// Extract email from JWT

public String extractEmail(String token) {

return Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

// Validate token

public boolean isTokenValid(String token, String email) {

try {

String extractedEmail = extractEmail(token);

return extractedEmail.equals(email) && !isTokenExpired(token);

} catch (JwtException e) {

return false;

}

}

// Check expiry

private boolean isTokenExpired(String token) {

Date expiration = Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

return expiration.before(new Date());

}

}

package com.banking.expenseplanner.security;

import io.jsonwebtoken.\*;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

*@Component*

public class JwtUtil {

private final String SECRET\_KEY = "nikhil\_very\_secret\_key\_which\_is\_32charlong!";

private final Key key = Keys.*hmacShaKeyFor*(SECRET\_KEY.getBytes());

// Generate JWT with 5 hours expiry

public String generateToken(String email) {

return Jwts.*builder*()

.setSubject(email)

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

.setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 5)) // 5 hours

.signWith(key, *SignatureAlgorithm*.***HS256***)

.compact();

}

// Extract email from JWT

public String extractEmail(String token) {

return Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

// Validate token

public boolean isTokenValid(String token, String email) {

try {

String extractedEmail = extractEmail(token);

return extractedEmail.equals(email) && !isTokenExpired(token);

} catch (JwtException e) {

return false;

}

}

// Check expiry

private boolean isTokenExpired(String token) {

Date expiration = Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

return expiration.before(new Date());

}

}

package com.banking.expenseplanner.service;

import com.banking.expenseplanner.dto.UserDto;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.UserRepository;

import com.banking.expenseplanner.security.JwtUtil;

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

import org.springframework.stereotype.Service;

import java.util.Optional;

*@Service*

public class UserService {

*@Autowired*

private UserRepository userRepository;

*@Autowired*

private JwtUtil jwtUtil;

// Registration logic

public String registerUser(UserDto userDto) {

Optional<User> existingUser = userRepository.findByEmail(userDto.getEmail());

if (existingUser.isPresent()) {

return "Email already exists!";

}

User user = new User();

user.setName(userDto.getName());

user.setEmail(userDto.getEmail());

user.setPassword(userDto.getPassword()); // basic plain password (not secure for production)

userRepository.save(user);

return "User registered successfully!";

}

// Login logic

public String loginUser(UserDto userDto) {

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

if (userOpt.isPresent()) {

User user = userOpt.get();

if (user.getPassword().equals(userDto.getPassword())) {

String token = jwtUtil.generateToken(user.getEmail());

return "Bearer " + token;

} else {

return "Invalid password!";

}

} else {

return "User not found!";

}

}

}

package com.banking.expenseplanner.util;

import com.banking.expenseplanner.model.User;

import com.banking.expenseplanner.repository.UserRepository;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Component;

*@Component*

public class UserUtil {

private final UserRepository userRepository;

public UserUtil(UserRepository userRepository) {

this.userRepository = userRepository;

}

public User getLoggedInUser() {

String email = SecurityContextHolder.*getContext*().getAuthentication().getName();

return userRepository.findByEmail(email).orElseThrow(() -> new RuntimeException("User not found"));

}

}

//application.properties

spring.application.name=BankingExpensePlanner

# ---------- SERVER CONFIG ----------

server.port=8080

# ---------- DATABASE CONFIG ----------

spring.datasource.url=jdbc:mysql://localhost:3306/banking\_expense

spring.datasource.username=root

spring.datasource.password=root

# ---------- JPA CONFIG ----------

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# ---------- OTHER CONFIG ----------

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

//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>3.5.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.banking</groupId>

<artifactId>expenseplanner</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>BankingExpensePlanner</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>24</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.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- Jakarta Bean Validation (for @NotBlank, @Email, etc.) -->

<dependency>

<groupId>jakarta.validation</groupId>

<artifactId>jakarta.validation-api</artifactId>

</dependency>

<!-- Required: Hibernate Validator (Jakarta Validation implementation) -->

<dependency>

<groupId>org.hibernate.validator</groupId>

<artifactId>hibernate-validator</artifactId>

</dependency>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- JJWT: Java JWT Token -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>