FINANCE MANAGEMENT SYSTEM

(SIVAGANESH NATARAJAVEL)

Database schema :

|  |
| --- |
| CREATE DATABASE FinanceDB;  USE FinanceDB;  -- Users Table  CREATE TABLE Users (  user\_id INT PRIMARY KEY,  username VARCHAR(50) NOT NULL,  password VARCHAR(255) NOT NULL,  email VARCHAR(100) UNIQUE NOT NULL  );  -- ExpenseCategories Table  CREATE TABLE ExpenseCategories (  category\_id INT PRIMARY KEY ,  category\_name VARCHAR(50) NOT NULL UNIQUE  );  -- Expenses Table  CREATE TABLE Expenses (  expense\_id INT PRIMARY KEY ,  user\_id INT NOT NULL,  amount DECIMAL(10,2) NOT NULL,  category\_id INT,  date DATE NOT NULL,  description TEXT,  FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,  FOREIGN KEY (category\_id) REFERENCES ExpenseCategories(category\_id) ON DELETE SET NULL  ); |

Code :

Main/FinanceApp.java :

|  |
| --- |
| package com.sivaganesh.finance.main;  import java.sql.Date; import java.util.List; import java.util.Scanner;  import com.sivaganesh.finance.dao.FinanceRepositoryImpl; import com.sivaganesh.finance.dao.IFinanceRepository; import com.sivaganesh.finance.entity.Expense; import com.sivaganesh.finance.entity.User; import com.sivaganesh.finance.exception.ExpenseNotFoundException; import com.sivaganesh.finance.exception.UserNotFoundException;  public class FinanceApp {   private static final Scanner *scanner* = new Scanner(System.*in*);  private static final IFinanceRepository *repo* = new FinanceRepositoryImpl();   public static void main(String[] args) {  System.*out*.println("====== Welcome to Finance Management System ======");   boolean exit = false;   while (!exit) {  System.*out*.println("\nChoose an option:");  System.*out*.println("1. Add User");  System.*out*.println("2. Add Expense");  System.*out*.println("3. Delete User");  System.*out*.println("4. Delete Expense");  System.*out*.println("5. Update Expense");  System.*out*.println("6. View All Expenses by User ID");  System.*out*.println("7. Generate Report by Date Range");  System.*out*.println("8. Exit");   int choice = *scanner*.nextInt();  *scanner*.nextLine(); // consume newline   switch (choice) {  case 1:  *addUser*();  break;  case 2:  *addExpense*();  break;  case 3:  *deleteUser*();  break;  case 4:  *deleteExpense*();  break;  case 5:  *updateExpense*();  break;  case 6:  *viewExpenses*();  break;  case 7:  *generateReport*();  break;  case 8:  exit = true;  System.*out*.println("Exiting... Thank you!");  break;  default:  System.*out*.println("Invalid choice!");  break;  }  }   *scanner*.close();  }   private static void addUser() {  System.*out*.print("Enter User ID: ");  int userId = *scanner*.nextInt();  *scanner*.nextLine(); // consume newline   System.*out*.print("Enter Username: ");  String username = *scanner*.nextLine();   System.*out*.print("Enter Password: ");  String password = *scanner*.nextLine();   System.*out*.print("Enter Email: ");  String email = *scanner*.nextLine();   User user = new User(userId, username, password, email);  if (*repo*.createUser(user)) {  System.*out*.println("User added successfully.");  } else {  System.*out*.println("Failed to add user.");  }  }   private static void addExpense() {  System.*out*.print("Enter Expense ID: ");  int expenseId = *scanner*.nextInt();   System.*out*.print("Enter User ID: ");  int userId = *scanner*.nextInt();   System.*out*.print("Enter Amount: ");  double amount = *scanner*.nextDouble();   System.*out*.print("Enter Category ID: ");  int categoryId = *scanner*.nextInt();  *scanner*.nextLine(); // consume newline   System.*out*.print("Enter Date (yyyy-mm-dd): ");  String dateStr = *scanner*.nextLine();   System.*out*.print("Enter Description: ");  String description = *scanner*.nextLine();   Expense expense = new Expense(expenseId, userId, amount, categoryId, Date.*valueOf*(dateStr), description);  if (*repo*.createExpense(expense)) {  System.*out*.println("Expense added successfully.");  } else {  System.*out*.println("Failed to add expense.");  }  }   private static void deleteUser() {  System.*out*.print("Enter User ID to delete: ");  int userId = *scanner*.nextInt();  try {  if (*repo*.deleteUser(userId)) {  System.*out*.println("User deleted successfully.");  }  } catch (UserNotFoundException e) {  System.*out*.println(e.getMessage());  }  }   private static void deleteExpense() {  System.*out*.print("Enter Expense ID to delete: ");  int expenseId = *scanner*.nextInt();  try {  if (*repo*.deleteExpense(expenseId)) {  System.*out*.println("Expense deleted successfully.");  }  } catch (ExpenseNotFoundException e) {  System.*out*.println(e.getMessage());  }  }   private static void updateExpense() {  System.*out*.print("Enter User ID: ");  int userId = *scanner*.nextInt();   System.*out*.print("Enter Expense ID to update: ");  int expenseId = *scanner*.nextInt();   System.*out*.print("Enter New Amount: ");  double amount = *scanner*.nextDouble();   System.*out*.print("Enter New Category ID: ");  int categoryId = *scanner*.nextInt();  *scanner*.nextLine(); // consume newline   System.*out*.print("Enter New Date (yyyy-mm-dd): ");  String dateStr = *scanner*.nextLine();   System.*out*.print("Enter New Description: ");  String description = *scanner*.nextLine();   Expense expense = new Expense(expenseId, userId, amount, categoryId, Date.*valueOf*(dateStr), description);  try {  if (*repo*.updateExpense(userId, expense)) {  System.*out*.println("Expense updated successfully.");  }  } catch (ExpenseNotFoundException e) {  System.*out*.println(e.getMessage());  }  }   private static void viewExpenses() {  System.*out*.print("Enter User ID to view expenses: ");  int userId = *scanner*.nextInt();  try {  List<Expense> expenses = *repo*.getAllExpenses(userId);  System.*out*.println("Expenses for User ID " + userId + ":");  for (Expense e : expenses) {  System.*out*.println("ID: " + e.getExpenseId() + ", Amount: " + e.getAmount() +  ", Category ID: " + e.getCategoryId() + ", Date: " + e.getDate() +  ", Description: " + e.getDescription());  }  } catch (UserNotFoundException e) {  System.*out*.println(e.getMessage());  }  }   private static void generateReport() {  System.*out*.print("Enter User ID: ");  int userId = *scanner*.nextInt();  *scanner*.nextLine(); // consume newline   System.*out*.print("Enter Start Date (yyyy-mm-dd): ");  String startDateStr = *scanner*.nextLine();   System.*out*.print("Enter End Date (yyyy-mm-dd): ");  String endDateStr = *scanner*.nextLine();   Date startDate = Date.*valueOf*(startDateStr);  Date endDate = Date.*valueOf*(endDateStr);   try {  List<Expense> report = *repo*.getExpensesByDateRange(userId, startDate, endDate);  System.*out*.println("\n--- Expense Report ---");  for (Expense e : report) {  System.*out*.println("ID: " + e.getExpenseId() + ", Amount: " + e.getAmount() +  ", Category ID: " + e.getCategoryId() + ", Date: " + e.getDate() +  ", Description: " + e.getDescription());  }  } catch (UserNotFoundException e) {  System.*out*.println(e.getMessage());  }  } } |

Util/DBConnection.java :

|  |
| --- |
| package com.sivaganesh.finance.util;  import java.sql.Connection; import java.sql.DriverManager; import java.util.Properties;  public class DBConnection {  private static Connection *connection* = null;   public static Connection getConnection() {  if (*connection* == null) {  try {  Properties props = PropertyUtil.*loadProperties*("db.properties");  if (props == null) return null;   String driver = props.getProperty("db.driver");  String url = props.getProperty("db.url");  String username = props.getProperty("db.username");  String password = props.getProperty("db.password");   Class.*forName*(driver);  *connection* = DriverManager.*getConnection*(url, username, password);  } catch (Exception e) {  System.*out*.println("Error establishing database connection: " + e.getMessage());  }  }  return *connection*;  } } |

Util/PropertyUtil.java:

|  |
| --- |
| package com.sivaganesh.finance.util;  import java.io.InputStream; import java.util.Properties;  public class PropertyUtil {   public static Properties loadProperties(String fileName) {  Properties props = new Properties();  try (InputStream input = PropertyUtil.class.getClassLoader().getResourceAsStream(fileName)) {  if (input == null) {  System.*out*.println("Unable to find " + fileName);  return null;  }  props.load(input);  } catch (Exception e) {  System.*out*.println("Error loading properties file: " + e.getMessage());  }  return props;  } } |

Dao/Finance RepositoryImpl :

|  |
| --- |
| package com.sivaganesh.finance.dao;  import java.sql.Connection; import java.sql.Date; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.util.ArrayList; import java.util.List;  import com.sivaganesh.finance.entity.Expense; import com.sivaganesh.finance.entity.User; import com.sivaganesh.finance.exception.ExpenseNotFoundException; import com.sivaganesh.finance.exception.UserNotFoundException; import com.sivaganesh.finance.util.DBConnection;  public class FinanceRepositoryImpl implements IFinanceRepository {   private final Connection conn;   public FinanceRepositoryImpl() {  conn = DBConnection.*getConnection*();  }   @Override  public boolean createUser(User user) {  String sql = "INSERT INTO Users (user\_id, username, password, email) VALUES (?, ?, ?, ?)";  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, user.getUserId());  stmt.setString(2, user.getUsername());  stmt.setString(3, user.getPassword());  stmt.setString(4, user.getEmail());  return stmt.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println("Error while creating user: " + e.getMessage());  }  return false;  }   @Override  public boolean createExpense(Expense expense) {  String sql = "INSERT INTO Expenses (expense\_id, user\_id, amount, category\_id, date, description) VALUES (?, ?, ?, ?, ?, ?)";  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, expense.getExpenseId());  stmt.setInt(2, expense.getUserId());  stmt.setDouble(3, expense.getAmount());  stmt.setInt(4, expense.getCategoryId());  stmt.setDate(5, expense.getDate());  stmt.setString(6, expense.getDescription());  return stmt.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println("Error while creating expense: " + e.getMessage());  }  return false;  }   @Override  public boolean deleteUser(int userId) throws UserNotFoundException {  String sql = "DELETE FROM Users WHERE user\_id = ?";  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, userId);  int rows = stmt.executeUpdate();  if (rows == 0) throw new UserNotFoundException("User not found with ID: " + userId);  return true;  } catch (SQLException e) {  System.*out*.println("Error while deleting user: " + e.getMessage());  }  return false;  }   @Override  public boolean deleteExpense(int expenseId) throws ExpenseNotFoundException {  String sql = "DELETE FROM Expenses WHERE expense\_id = ?";  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, expenseId);  int rows = stmt.executeUpdate();  if (rows == 0) throw new ExpenseNotFoundException("Expense not found with ID: " + expenseId);  return true;  } catch (SQLException e) {  System.*out*.println("Error while deleting expense: " + e.getMessage());  }  return false;  }   @Override  public List<Expense> getAllExpenses(int userId) throws UserNotFoundException {  String sql = "SELECT \* FROM Expenses WHERE user\_id = ?";  List<Expense> expenses = new ArrayList<>();  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, userId);  ResultSet rs = stmt.executeQuery();  boolean found = false;  while (rs.next()) {  found = true;  Expense exp = new Expense(  rs.getInt("expense\_id"),  rs.getInt("user\_id"),  rs.getDouble("amount"),  rs.getInt("category\_id"),  rs.getDate("date"),  rs.getString("description")  );  expenses.add(exp);  }  if (!found) throw new UserNotFoundException("No expenses found for user ID: " + userId);  } catch (SQLException e) {  System.*out*.println("Error while fetching expenses: " + e.getMessage());  }  return expenses;  }   @Override  public boolean updateExpense(int userId, Expense expense) throws ExpenseNotFoundException {  String sql = "UPDATE Expenses SET amount = ?, category\_id = ?, date = ?, description = ? WHERE expense\_id = ? AND user\_id = ?";  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setDouble(1, expense.getAmount());  stmt.setInt(2, expense.getCategoryId());  stmt.setDate(3, expense.getDate());  stmt.setString(4, expense.getDescription());  stmt.setInt(5, expense.getExpenseId());  stmt.setInt(6, userId);  int rows = stmt.executeUpdate();  if (rows == 0) throw new ExpenseNotFoundException("No expense found for update.");  return true;  } catch (SQLException e) {  System.*out*.println("Error while updating expense: " + e.getMessage());  }  return false;  }   @Override  public List<Expense> getExpensesByDateRange(int userId, Date from, Date to) throws UserNotFoundException {  String sql = "SELECT \* FROM Expenses WHERE user\_id = ? AND date BETWEEN ? AND ?";  List<Expense> expenses = new ArrayList<>();  try (PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, userId);  stmt.setDate(2, from);  stmt.setDate(3, to);  ResultSet rs = stmt.executeQuery();  boolean found = false;  while (rs.next()) {  found = true;  Expense exp = new Expense(  rs.getInt("expense\_id"),  rs.getInt("user\_id"),  rs.getDouble("amount"),  rs.getInt("category\_id"),  rs.getDate("date"),  rs.getString("description")  );  expenses.add(exp);  }  if (!found) throw new UserNotFoundException("No expenses found for user ID " + userId + " in the given date range.");  } catch (SQLException e) {  System.*out*.println("Error while generating report: " + e.getMessage());  }  return expenses;  } } |

Dao/IFinanceRepository:

|  |
| --- |
| package com.sivaganesh.finance.dao;  import java.sql.Date; import java.util.List;  import com.sivaganesh.finance.entity.User; import com.sivaganesh.finance.entity.Expense; import com.sivaganesh.finance.exception.UserNotFoundException; import com.sivaganesh.finance.exception.ExpenseNotFoundException;  public interface IFinanceRepository {  boolean createUser(User user);  boolean createExpense(Expense expense);  boolean deleteUser(int userId) throws UserNotFoundException;  boolean deleteExpense(int expenseId) throws ExpenseNotFoundException;  List<Expense> getAllExpenses(int userId) throws UserNotFoundException;  boolean updateExpense(int userId, Expense expense) throws ExpenseNotFoundException;  List<Expense> getExpensesByDateRange(int userId, Date from, Date to) throws UserNotFoundException; } |

Entity/Expense.java

|  |
| --- |
| package com.sivaganesh.finance.entity;  import java.sql.Date;  public class Expense {  private int expenseId;  private int userId;  private double amount;  private int categoryId;  private Date date;  private String description;   public Expense() {}   public Expense(int expenseId, int userId, double amount, int categoryId, Date date, String description) {  this.expenseId = expenseId;  this.userId = userId;  this.amount = amount;  this.categoryId = categoryId;  this.date = date;  this.description = description;  }   public int getExpenseId() { return expenseId; }  public void setExpenseId(int expenseId) { this.expenseId = expenseId; }   public int getUserId() { return userId; }  public void setUserId(int userId) { this.userId = userId; }   public double getAmount() { return amount; }  public void setAmount(double amount) { this.amount = amount; }   public int getCategoryId() { return categoryId; }  public void setCategoryId(int categoryId) { this.categoryId = categoryId; }   public Date getDate() { return date; }  public void setDate(Date date) { this.date = date; }   public String getDescription() { return description; }  public void setDescription(String description) { this.description = description; } } |

Entity/ExpenseCategory.java :

|  |
| --- |
| package com.sivaganesh.finance.entity;  public class ExpenseCategory {  private int categoryId;  private String categoryName;   public ExpenseCategory() {}   public ExpenseCategory(int categoryId, String categoryName) {  this.categoryId = categoryId;  this.categoryName = categoryName;  }   public int getCategoryId() { return categoryId; }  public void setCategoryId(int categoryId) { this.categoryId = categoryId; }   public String getCategoryName() { return categoryName; }  public void setCategoryName(String categoryName) { this.categoryName = categoryName; } } |

Entity/user.java :

|  |
| --- |
| package com.sivaganesh.finance.entity;  public class User {  private int userId;  private String username;  private String password;  private String email;   public User() {}   public User(int userId, String username, String password, String email) {  this.userId = userId;  this.username = username;  this.password = password;  this.email = email;  }   public int getUserId() { return userId; }  public void setUserId(int userId) { this.userId = userId; }   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; } } |

Exception/ExpenseNotFoundException:

|  |
| --- |
| package com.sivaganesh.finance.exception;  public class ExpenseNotFoundException extends RuntimeException {  public ExpenseNotFoundException() {  super("Expense not found.");  }   public ExpenseNotFoundException(String message) {  super(message);  } } |

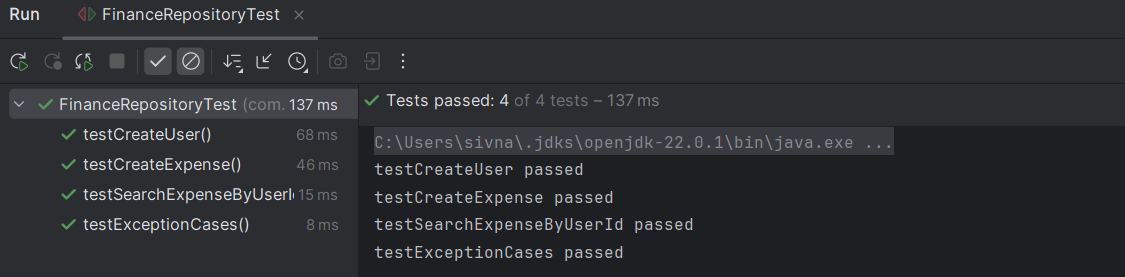
Exception/UserNotFoundException :

|  |
| --- |
| package com.sivaganesh.finance.exception;  public class UserNotFoundException extends RuntimeException {  public UserNotFoundException() {  super("User not found.");  }   public UserNotFoundException(String message) {  super(message);  } } |

Java/test/FinanceRepositoryTest.java :

|  |
| --- |
| package com.sivaganesh.finance;  import com.sivaganesh.finance.dao.FinanceRepositoryImpl; import com.sivaganesh.finance.dao.IFinanceRepository; import com.sivaganesh.finance.entity.Expense; import com.sivaganesh.finance.entity.User; import com.sivaganesh.finance.exception.ExpenseNotFoundException; import com.sivaganesh.finance.exception.UserNotFoundException; import org.junit.jupiter.api.\*;  import java.sql.Date; import java.time.LocalDate; import java.util.List;  @TestMethodOrder(MethodOrderer.OrderAnnotation.class) public class FinanceRepositoryTest {   private static IFinanceRepository *repository*;  private static final int *testUserId* = 5001;  private static final int *testCategoryId* = 1; // Ensure this category exists  private static final int *testExpenseId* = 7001;   @BeforeAll  public static void setup() {  *repository* = new FinanceRepositoryImpl();  }   @Test  @Order(1)  public void testCreateUser() {  User user = new User(*testUserId*, "testuser", "testpass", "testuser5001@example.com");  boolean result = *repository*.createUser(user);  Assertions.*assertTrue*(result, "User creation failed");  System.*out*.println("testCreateUser passed");  }   @Test  @Order(2)  public void testCreateExpense() {  Expense expense = new Expense(*testExpenseId*, *testUserId*, 999.99, *testCategoryId*,  Date.*valueOf*(LocalDate.*now*()), "Test Expense");  boolean result = *repository*.createExpense(expense);  Assertions.*assertTrue*(result, "Expense creation failed");  System.*out*.println("testCreateExpense passed");  }   @Test  @Order(3)  public void testSearchExpenseByUserId() throws UserNotFoundException {  List<Expense> expenses = *repository*.getAllExpenses(*testUserId*);  Assertions.*assertFalse*(expenses.isEmpty(), "No expenses found for user");  Assertions.*assertTrue*(expenses.stream().anyMatch(e -> e.getExpenseId() == *testExpenseId*));  System.*out*.println("testSearchExpenseByUserId passed");  }   @Test  @Order(4)  public void testExceptionCases() {  Assertions.*assertThrows*(UserNotFoundException.class, () -> {  *repository*.getAllExpenses(9999);  });  Assertions.*assertThrows*(ExpenseNotFoundException.class, () -> {  *repository*.deleteExpense(9999);  });  System.*out*.println("testExceptionCases passed");  } } |

Test Result:



Output :

