i want answers of mca question paper and i will provide you the answers
ChatGPT said:
Sure, Rashid! 😋
If you share the MCA question paper (or specific questions) and your answers, I can:

Check and correct your answers (if needed).

Provide detailed solutions and explanations for each question.

Help with step-by-step solutions for programming or theoretical questions.

Would you like to upload the MCA question paper image/PDF now?

You said:
i want ooad and web technologies lab practical question papers solution i will five you the questions
ChatGPT said:
Got it, Rashid! 👍

Share your OOAD (Object-Oriented Analysis & Design) and Web Technologies Lab practical questions, and I'll give you complete solutions with explanations and code (if needed).

Do you want me to:

Provide direct answers/code for each lab question,

Or also give step-by-step explanations for better understanding?

You said:
section 1: ooad
Note: Answer the following question. It carries 20 marks

Draw collaboration diagram for online Banking System.
ChatGPT said:
Alright, Rashid! Let's solve this OOAD Question:

Question:
Draw a Collaboration Diagram for Online Banking System (20 Marks).

Answer (Explanation):
A Collaboration Diagram (also called a Communication Diagram) shows how objects interact with each other and the sequence of messages they pass to accomplish a task.

For an Online Banking System, the main use case could be “Perform Transaction (Funds Transfer)” or “Check Balance.”

Key Objects for Online Banking System:
Customer – User interacting with the system.

Login Page – Handles user authentication.

Banking System – The main system that processes requests.

Account – Represents the customer's account.

Transaction – Manages debit/credit or balance checking.

Database – Stores account details and transactions.

Steps (Scenario: Funds Transfer):
Customer logs in.

System verifies credentials.

Customer requests fund transfer.

Banking system checks account balance.

Transaction is processed (debit from sender, credit to receiver).

Confirmation is displayed.

Sample Collaboration Diagram (Description):
Objects:
Customer : User
Login : Page
Bank : BankingSystem
Acc : Account
Txn : Transaction
DB : Database

Messages (Numbered for sequence):

login(username, password) → Login

verifyUser() → Bank

validateCredentials() → DB

requestTransfer() → Bank

checkBalance() → Acc

debit() → Txn

credit() → Txn

updateAccount() → DB

showConfirmation() → Customer

Diagram (ASCII Representation):
rust
Copy
Edit

```
Customer <-> Login <-> Bank <-> Account <-> Transaction <-> Database
   1         2       4         5           6,7        8
```

How to Draw in Exam (Neat Sketch):
Use boxes for objects: :Customer, :Login, :Bank, etc.

Connect them with lines (association links).

Add numbered arrows (1, 2, 3…) to show the sequence of messages.

Mention message names like login(), verifyUser(), debit(), etc.

Would you like me to draw a clean collaboration diagram for this online banking system (as an image you can directly use)?

You said:
Note: Answer the following question. It carries 20 marks
An online food delivery company provides a range of facilities to customers through its Mobile App/Web portal. It provide options to choose Restaurant, Food items, Online and COD (Cash on delivery) options, Order Status, and Feedback facility. Perform the following for this system:
(i) Identify classes and their associations.
(ii) Draw Class Diagram for the System.

ChatGPT said:

Alright, Rashid! Let's solve this OOAD Question step by step:

Question:
An online food delivery company provides options like Restaurant selection, Food items, Online/COD payments, Order Status, and Feedback facility.

Tasks:
(i) Identify classes and their associations.
(ii) Draw a Class Diagram for the System. (20 Marks)

(i) Identify Classes and Their Associations
Main Classes:
Customer – Places orders, gives feedback.

Restaurant – Provides menu (food items).

FoodItem – Represents individual dishes with price, category, etc.

Order – Contains ordered items, status, and payment method.

Payment – Handles online/COD transactions.

Delivery – Manages delivery details (address, time, status).

Feedback – Customer reviews for restaurants or food.

Admin (Optional) – Manages restaurants and food listings.

Associations:
Customer – Order: A customer can place multiple orders (1..*).

Order – FoodItem: An order can have multiple food items (..).

Restaurant – FoodItem: A restaurant offers multiple food items (1..*).

Order – Payment: Each order has one payment (1..1).

Order – Delivery: Each order is delivered once (1..1).

Customer – Feedback: A customer can give feedback (0..*).

(ii) Class Diagram (Structure)
Attributes and Methods (Example):

Customer: name, phone, address, placeOrder(), giveFeedback()

Restaurant: name, location, getMenu()

FoodItem: name, price, quantity

Order: orderId, orderDate, status, calculateTotal()

Payment: paymentId, paymentType, processPayment()

Delivery: deliveryId, deliveryTime, trackDelivery()

Feedback: feedbackId, rating, comment

Text Representation of Class Diagram:
pgsql
Copy
Edit

```
Customer ----- places -----> Order ----- contains -----> FoodItem
   |                    |                    ^
   | gives                 | paid via             |
   | feedback               v                    |
```

Feedback Payment offered by
Restaurant
Order ---- delivered via ----> Delivery
Would you like me to create a clean Class Diagram (image)
with UML notation for this Online Food Delivery System, so you can directly use it for your exam?

You said:
Note: Answer the following question. It carries 20 marks
Implement following association in C++/Java.
Order Customer
OrdNumber: String name:string
Price : Real * 1 address:string
IsPrepaid : Boolean -------------
dateoforder : Date

dispatch() creditRating()
close() cancelorder()
ChatGPT said:
Alright, Rashid! Let's solve this OOAD Implementation Question step by step.

Question:
Implement the association between Order (*) and Customer (1) in C++/Java, with the following details:

Order class:
OrdNumber : String
Price : Real
IsPrepaid : Boolean
dateoforder : Date
Methods: dispatch(), close()

Customer class:

name : String
address : String
Methods: creditRating(), cancelorder()

Explanation of Association:
A Customer can have multiple Orders (1..* relationship).

Each Order is associated with exactly one Customer.

Java Implementation
java
Copy
Edit

```java
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

// Customer Class
class Customer {
    private String name;
    private String address;
    private List<Order> orders;  // One customer can have multiple orders

    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
        this.orders = new ArrayList<>();
    }

    public void addOrder(Order order) {
        orders.add(order);
    }
    public void creditRating() {
        System.out.println(name + " has a good credit rating.");
    }

    public void cancelOrder(String orderNumber) {
        for (Order order : orders) {
            if (order.getOrdNumber().equals(orderNumber)) {
                System.out.println("Order " + orderNumber + " cancelled.");
                orders.remove(order);
                break;
            }
        }
    }

    public void displayOrders() {
        System.out.println("Orders of " + name + ":");
        for (Order order : orders) {
            System.out.println(order);
        }
    }
}

// Order Class
class Order {
    private String ordNumber;
    private double price;
    private boolean isPrepaid;
    private Date dateOfOrder;
    private Customer customer;  // Associated customer

    public Order(String ordNumber, double price, boolean isPrepaid, Date
dateOfOrder, Customer customer) {
        this.ordNumber = ordNumber;
        this.price = price;
        this.isPrepaid = isPrepaid;
        this.dateOfOrder = dateOfOrder;
        this.customer = customer;
        customer.addOrder(this); // Link order with customer
    }

    public String getOrdNumber() {
        return ordNumber;
    }

    public void dispatch() {
        System.out.println("Order " + ordNumber + " has been dispatched.");
    }

    public void close() {
        System.out.println("Order " + ordNumber + " is closed.");
    }

    @Override
    public String toString() {
        return "Order{" +
                "ordNumber='" + ordNumber + '\'' +
                ", price=" + price +
                ", isPrepaid=" + isPrepaid +
                ", dateOfOrder=" + dateOfOrder +
                '}';
    }
}

// Main Class to Test
public class Main {
    public static void main(String[] args) {
        Customer c1 = new Customer("Rashid", "Mangalore");
        Order o1 = new Order("O101", 2500.50, true, new Date(), c1);
        Order o2 = new Order("O102", 1200.75, false, new Date(), c1);

        c1.displayOrders();
        o1.dispatch();
        c1.cancelOrder("O102");
        c1.displayOrders();
    }
}
```

Key Points in the Code:
Customer has a List<Order> to maintain multiple orders.

When an Order is created, it is linked to a Customer (customer.addOrder(this)).

Methods dispatch() and close() are implemented in Order.

Methods creditRating() and cancelOrder() are implemented in Customer.

Would you like me to create a similar clean C++ version for the same association (Order–Customer) so you have both languages ready for exam answers?

You said:
now i will provide you about web technologies
ChatGPT said:
Great, Rashid! 👍
Share your Web Technologies lab questions, and I'll give you complete solutions with code and explanations.

Would you like me to provide solutions in HTML, CSS, JavaScript, and PHP (if needed), or just the basic HTML/JS approach?

You said:
Note: Answer the following question. It carries 20 marks
Create a user table into the database and bind the user entity with Spring Security for Login.
ChatGPT said:
Alright, Rashid! This Web Technologies question is based on Spring Boot + Spring Security. Let's break it down and write the complete solution.

Question:
Create a user table in the database and bind the user entity with Spring Security for login. (20 marks)

Answer (Step-by-Step):
1. Create User Table in Database
sql
Copy
Edit

```
CREATE TABLE users (
   id BIGINT PRIMARY KEY AUTO_INCREMENT,
   username VARCHAR(50) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL,
   role VARCHAR(50) NOT NULL
);
```

Insert Sample User:

sql
Copy
Edit

```
INSERT INTO users (username, password, role)
VALUES ('admin', '{noop}admin123', 'ROLE_ADMIN');
```

{noop} means "no password encoder" in Spring Security (for testing).

2. User Entity in Spring Boot

java

Copy

Edit

```java
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;
    private String role;

    // 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 getRole() { return role; }
    public void setRole(String role) { this.role = role; }
}
```

3. User Repository

java
Copy
Edit

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

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

4. UserDetailsService Implementation

java
Copy
Edit

```
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        return org.springframework.security.core.userdetails.User
                .withUsername(user.getUsername())
                .password(user.getPassword())
                .roles(user.getRole())
                .build();
    }
}
```

5. Spring Security Configuration

java

Copy

Edit

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {

    private final CustomUserDetailsService customUserDetailsService;

    public SecurityConfig(CustomUserDetailsService customUserDetailsService)
{
        this.customUserDetailsService = customUserDetailsService;
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/login").permitAll()
                .anyRequest().authenticated()
            )
            .formLogin(withDefaults -> {})
            .userDetailsService(customUserDetailsService);

        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance(); // For testing only
    }
}
```

6. Run and Test

Start Spring Boot.

Access http://localhost:8080/login.

Login with username: admin, password: admin123.

Marks Distribution (20 Marks)

SQL users table (4 marks)

User entity (3 marks)

Repository (3 marks)

CustomUserDetailsService (4 marks)

SecurityConfig (6 marks)

Next Step:
Would you like me to give you a ready-to-copy minimal Spring Boot project structure (with all files) for this login using Spring Security + MySQL?

You said:
Note: Answer the following question. It carries 20 marks
Write a program using JSP and JDBC create CRUD (Create/Save, Read, Edit/Update, Delete) application for students attendance management in MCSL-222 counselling classes.
ChatGPT said:
Alright, Rashid! Let's solve this Web Technologies question step by step.

Question:
Write a program using JSP and JDBC to create a CRUD application for student attendance management in MCSL-222 counselling classes. (20 Marks)

Answer (Step-by-Step Solution):
We'll create a JSP + JDBC application with the following features:

Create/Save: Add a student with attendance details.

Read: View all students and their attendance.

Edit/Update: Update a student's attendance.

Delete: Remove a student's record.

1. Database Setup
SQL Table:
sql
Copy
Edit

```sql
CREATE DATABASE mcsl222;

USE mcsl222;

CREATE TABLE students (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   roll_no VARCHAR(20) NOT NULL,
   attendance INT NOT NULL
);
```

2. Database Connection (dbconnect.jsp)
jsp
Copy
Edit

```jsp
<%@ page import="java.sql.*" %>
<%
   String url = "jdbc:mysql://localhost:3306/mcsl222";
   String user = "root";
   String pass = "your_password";
   Connection conn = null;

   try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      conn = DriverManager.getConnection(url, user, pass);
   } catch (Exception e) {
      out.println("Database connection error: " + e.getMessage());
   }
%>
```

3. Add Student (addStudent.jsp)

jsp
Copy
Edit

```
<%@ include file="dbconnect.jsp" %>
<%
   String name = request.getParameter("name");
   String roll = request.getParameter("roll_no");
   String attendance = request.getParameter("attendance");

   if (name != null && roll != null && attendance != null) {
      PreparedStatement ps = conn.prepareStatement("INSERT INTO
students(name, roll_no, attendance) VALUES (?, ?, ?)");
      ps.setString(1, name);
      ps.setString(2, roll);
      ps.setInt(3, Integer.parseInt(attendance));
      ps.executeUpdate();
      out.println("<p>Student added successfully!</p>");
   }
%>

<form method="post" action="addStudent.jsp">
   Name: <input type="text" name="name"><br>
   Roll No: <input type="text" name="roll_no"><br>
   Attendance: <input type="number" name="attendance"><br>
   <input type="submit" value="Add Student">
</form>
```

4. Display Students (listStudents.jsp)

jsp
Copy
Edit

```
<%@ include file="dbconnect.jsp" %>
<h2>Student Attendance List</h2>
```

```
<table border="1">
<tr>
   <th>ID</th><th>Name</th><th>Roll No</th><th>Attendance</th><th>Actions</th>
</tr>
<%
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery("SELECT * FROM students");
   while (rs.next()) {
%>
<tr>
   <td><%= rs.getInt("id") %></td>
   <td><%= rs.getString("name") %></td>
   <td><%= rs.getString("roll_no") %></td>
   <td><%= rs.getInt("attendance") %></td>
   <td>
      <a href="editStudent.jsp?id=<%= rs.getInt("id") %>">Edit</a> |
      <a href="deleteStudent.jsp?id=<%= rs.getInt("id") %>">Delete</a>
   </td>
</tr>
<%
   }
%>
</table>
<a href="addStudent.jsp">Add New Student</a>
```

5. Edit Student (editStudent.jsp)

jsp

Copy

Edit

```
<%@ include file="dbconnect.jsp" %>
<%
   int id = Integer.parseInt(request.getParameter("id"));
   String update = request.getParameter("update");
```

```
    if ("true".equals(update)) {
        String name = request.getParameter("name");
        String roll = request.getParameter("roll_no");
        int attendance = Integer.parseInt(request.getParameter("attendance"));

        PreparedStatement ps = conn.prepareStatement("UPDATE students SET name=?, roll_no=?, attendance=? WHERE id=?");
        ps.setString(1, name);
        ps.setString(2, roll);
        ps.setInt(3, attendance);
        ps.setInt(4, id);
        ps.executeUpdate();
        response.sendRedirect("listStudents.jsp");
    } else {
        PreparedStatement ps = conn.prepareStatement("SELECT * FROM students WHERE id=?");
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
%>
<form method="post" action="editStudent.jsp">
    <input type="hidden" name="id" value="<%= id %>">
    Name: <input type="text" name="name" value="<%= rs.getString("name") %>"><br>
    Roll No: <input type="text" name="roll_no" value="<%= rs.getString("roll_no") %>"><br>
    Attendance: <input type="number" name="attendance" value="<%= rs.getInt("attendance") %>"><br>
    <input type="hidden" name="update" value="true">
    <input type="submit" value="Update Student">
</form>
<%
```

```
        }
    }
%>
```

6. Delete Student (deleteStudent.jsp)
jsp
Copy
Edit

```
<%@ include file="dbconnect.jsp" %>
<%
    int id = Integer.parseInt(request.getParameter("id"));
    PreparedStatement ps = conn.prepareStatement("DELETE FROM students WHERE id=?");
    ps.setInt(1, id);
    ps.executeUpdate();
    response.sendRedirect("listStudents.jsp");
%>
```

7. Run Application
Place all .jsp files in webapp/ folder of your project.

Use Tomcat Server to run.

Visit http://localhost:8080/YourApp/listStudents.jsp.

Marks Distribution (20 Marks)
Database and table creation (3 marks)

Insert (CRUD - Create) logic (4 marks)

Read (list students) (4 marks)

Edit/Update functionality (4 marks)

Delete functionality (3 marks)

Clean JSP structure and JDBC code (2 marks)

Next Step:
Would you like me to bundle all JSP files into a ready-to-use zip (mini project) for this CRUD Attendance Management system so you can run it directly?

You said:
Note: Answer the following question. It carries 20 marks
Write simple program for role-based authentication using Spring Boot.
ChatGPT said:
Alright, Rashid! Here's how we can answer this Web Technologies question:

Question:
Write a simple program for role-based authentication using Spring Boot. (20 marks)

Answer (Step-by-Step Solution)
We will create a Spring Boot application with Spring Security and define roles like ADMIN and USER. Pages will be accessible based on roles.

1. Add Dependencies (pom.xml)
xml
Copy
Edit

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
   </dependency>
</dependencies>
```

2. Security Configuration

java
Copy
Edit

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails admin = User.withUsername("admin")
                .password("{noop}admin123")   // {noop} means no password encoding
                .roles("ADMIN")
                .build();

        UserDetails user = User.withUsername("user")
                .password("{noop}user123")
                .roles("USER")
                .build();
        return new InMemoryUserDetailsManager(admin, user);
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/admin/**").hasRole("ADMIN")
                .requestMatchers("/user/**").hasRole("USER")
                .anyRequest().authenticated()
            )
            .formLogin(withDefaults -> {})
            .httpBasic(withDefaults -> {});
        return http.build();
    }
}
```

3. Create Controllers

java

Copy

Edit

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "Welcome! Login as USER or ADMIN.";
    }

    @GetMapping("/user/dashboard")
    public String userPage() {
        return "Hello User, this is your dashboard!";
    }

    @GetMapping("/admin/dashboard")
    public String adminPage() {
        return "Hello Admin, this is the admin panel!";
    }
}
```

4. Run the Application

Steps to Test:

Start Spring Boot (mvn spring-boot:run or run the main class).

Go to http://localhost:8080/user/dashboard and login as:

Username: user

Password: user123

For admin panel:

Username: admin

Password: admin123

5. Marks Distribution (20 Marks)

Spring Boot setup and dependencies (4 marks)

SecurityConfig with role-based rules (6 marks)

UserDetailsService with roles (4 marks)

Controller endpoints and testing (4 marks)

Correct usage of Spring Security (2 marks)

Next Step:
Would you like me to create a ready-to-run Spring Boot mini project (ZIP) with this role-based authentication (admin/user)?

You said:
can you please give this answers in simple form for web technologies it is very complicated
ChatGPT said:
Got it, Rashid! Let me simplify the answers for your Web Technologies questions so you can easily write them in exams.

1. Spring Security – User Table Binding (Simplified Answer)
Steps:

Create users table:

sql
Copy
Edit

```
CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50),
   password VARCHAR(100),
   role VARCHAR(20)
);
```

Add one user:

sql
Copy
Edit

```
INSERT INTO users (username, password, role)
VALUES ('admin', '{noop}admin123', 'ROLE_ADMIN');
```

User Entity (Java Class):

java
Copy
Edit

```
@Entity
public class User {
   @Id @GeneratedValue
   private Long id;
   private String username;
   private String password;
   private String role;
   // getters & setters
}
```

Security Config (Simple):

java
Copy
Edit

```
@Configuration
public class SecurityConfig {
   @Bean
   SecurityFilterChain security(HttpSecurity http) throws Exception {
      http.authorizeHttpRequests(auth -> auth.anyRequest().authenticated())
         .formLogin();
      return http.build();
   }
}
```

In short:

Create a table with username & password.

Map it using a User entity.

Use Spring Security for login.

2. JSP + JDBC CRUD (Simplified Answer)

Steps:

Create Table:

sql
Copy
Edit

```
CREATE TABLE students (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50),
   roll_no VARCHAR(20),
   attendance INT
);
```

Database Connection (dbconnect.jsp):

jsp
Copy
Edit

```
<%
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn = DriverManager.getConnection(
   "jdbc:mysql://localhost:3306/mcsl222","root","password");
%>
```

Add Student (Create):

jsp
Copy

Edit

```
<form action="addStudent.jsp" method="post">
  Name:<input name="name"><br>
  Roll:<input name="roll_no"><br>
  Attendance:<input name="attendance"><br>
  <input type="submit" value="Save">
</form>
```

Read / List Students:

jsp

Copy

Edit

```
<%
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM students");
while(rs.next()) {
    out.print(rs.getString("name")+"<br>");
}
%>
```

Edit (Update) – Use UPDATE students SET ....

Delete – Use DELETE FROM students WHERE id=?.

3. Role-based Authentication in Spring Boot (Simplified Answer)

Steps:

Add Spring Security dependency.

In SecurityConfig:

java

Copy

Edit

```
@Configuration
public class SecurityConfig {
    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails admin = User.withUsername("admin")
                                .password("{noop}admin123")
                                .roles("ADMIN").build();
        UserDetails user = User.withUsername("user")
                               .password("{noop}user123")
                               .roles("USER").build();
        return new InMemoryUserDetailsManager(admin, user);
    }
}
```

Create two pages: /admin (only for ADMIN) and /user (only for USER).