SECURE SOFTWARE ENGINEERING

Activity 1

System Name : *Case Management*

Students' names :

Ahmed Abdel-Wasie Al-hakeeme

Ramzi Nasr Al-Qabati

Mohammed Fahd Al-Mughalles

Majed Ahmed Baggash

Nahal Aref Al-Adini

Subject Doctor:

Dr: Mohammed Alqmase

Activity 2

1. **Title : Case Management System**
2. **Brief description:**

The **Case Management** subsystem is designed to facilitate the efficient filing, tracking, and management of cases involving litigants, lawyers, and court officials. This subsystem provides a comprehensive framework for handling all aspects of case management, ensuring that relevant information is easily accessible and organized.

3. **Function within The JMS project :**

The Case Management subsystem plays a crucial role in the Judiciary Management System **(JMS)** by streamlining the workflow associated with case handling. It allows users **to**:

- **File Cases:** Users can submit new cases electronically, reducing paperwork and improving efficiency.
- **Track Cases:** The subsystem enables real-time tracking of case status, ensuring that all parties are informed about the progress and updates.
- **Manage Hearings and Deadlines:** It supports scheduling hearings, managing deadlines, and sending notifications to relevant stakeholders, thereby enhancing communication and coordination.

4. **Importance to the comprehensive system:**

The Case Management subsystem is essential for the overall effectiveness of the JMS. By centralizing case information and automating key processes, it minimizes delays and errors, ultimately leading to a more efficient judicial process. This subsystem not only improves the service provided to litigants and lawyers but also enhances the operational capabilities of court officials, contributing to a more transparent and accessible justice system.

Activity 3

Title : Problem formulation for a case management system

1. Problem definition:

The judicial process often suffers from inefficiencies due to the manual handling of case files, which can lead to delays, misplaced documents, and communication breakdowns among litigants, lawyers, and court officials. Specifically, the lack of a centralized system for filing and tracking cases **results in:**

- **Inefficient Case Tracking:** Cases are frequently lost or mismanaged due to reliance on paper-based systems.
- **Scheduling Conflicts:** Difficulty in coordinating hearings and deadlines leads to missed appointments and unnecessary delays.
- **Poor communication:** Stakeholders are often not updated in real-time about changes in case status or upcoming deadlines, resulting in confusion and frustration.

2. justification for the Subsystem:

The Case Management subsystem is essential for addressing these problems and enhancing the efficiency of the judicial process. Its implementation will provide several **key benefits**:

- **Centralized Case Management**: By digitizing case files, the subsystem ensures that all information is stored in one accessible location, reducing the risk of loss or mismanagement.
- **Automated Scheduling and Notifications:** The subsystem automates the scheduling of hearings and deadlines, significantly reducing the chances of conflicts and ensuring that all parties are informed well in advance.
- **Improved Stakeholder Communication:** With real-time updates and notifications, all stakeholders can stay informed about case progress, fostering better collaboration and reducing misunderstandings.

Activity 4

1. ***Title :*** Objectives and Motivations for the Case Management Subsystem

2. ***Objectives:***

The primary objectives of the Case Management subsystem are **as follows:**

- **Improve Case Processing Time:** Streamline the filing and tracking of cases to reduce delays in the judicial process.
- **Enhance User Access Management:** Ensure that all users (litigants, lawyers, court officials) have appropriate access to relevant case information, fostering collaboration and transparency.
- **Automate Notifications:** Implement automated notifications for hearings, deadlines, and updates to keep all stakeholders informed in real-time.
- **Centralize Case Information:** Provide a single platform for storing and retrieving case files, improving efficiency and reducing the risk of document loss.
- **Facilitate Reporting and Analytics:** Enable the generation of reports for better decision-making and resource allocation within the judicial system.

3. Motivations:

The motivations for implementing the Case Management subsystem **include**:

- **Alignment with Judicial Goals:** The subsystem supports the core values of the judicial system, including transparency, efficiency, and accountability. By centralizing information and automating processes, it directly addresses the need for a more effective legal system.
- **Enhanced Public Trust:** By improving the efficiency and accessibility of case management, the subsystem aims to increase public confidence in the judicial process.
- **Reduction of Administrative Burdens:** Automating routine tasks reduces the workload on court officials, allowing them to focus on more critical aspects of case management.
- **Support for Future Innovations:** Creating a robust case management framework lays the groundwork for future advancements, such as integrating AI for predictive case analytics and enhanced user experiences.
- **Security and Compliance:** The subsystem will incorporate security measures to protect sensitive information, ensuring compliance with legal and ethical standards.

Activity 5

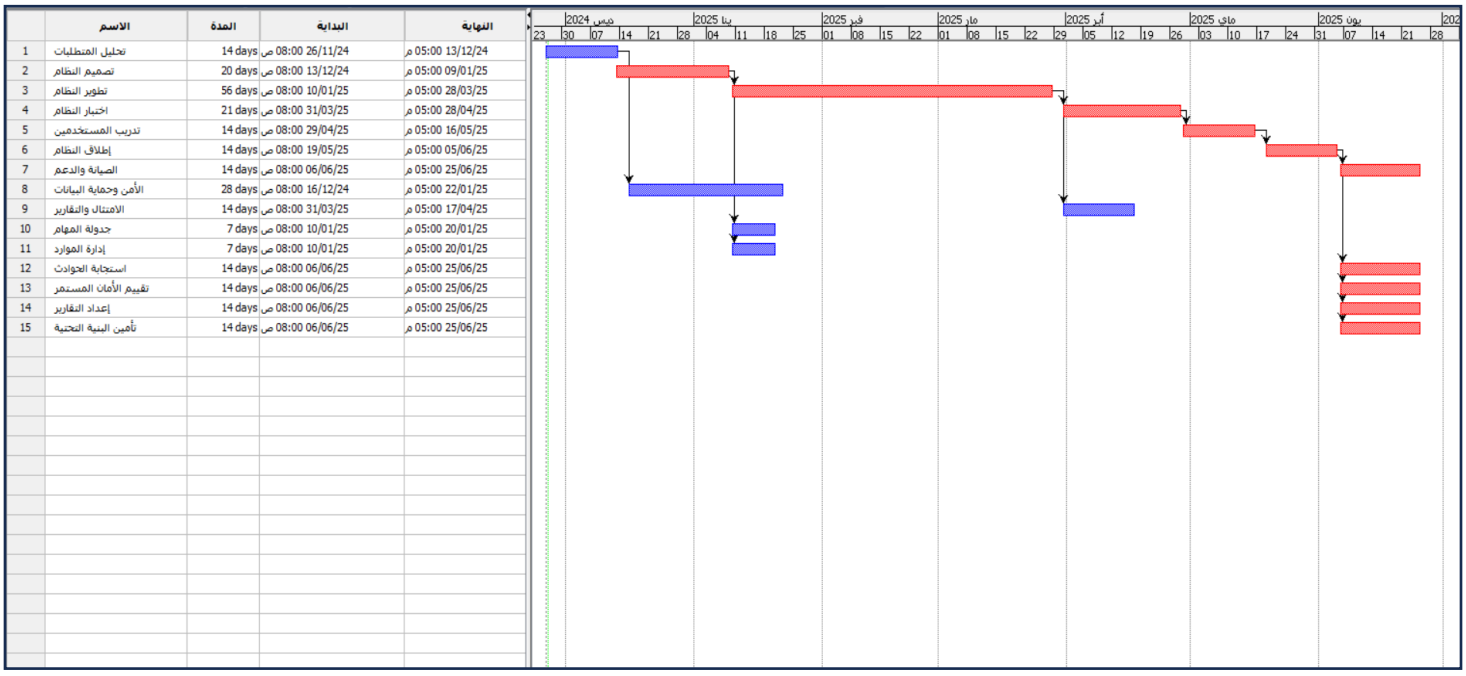
1- To-do list:

The number	Mission	Duration
1	***Analysis of judicial requirements***	2 weeks
1.1	Gathering requirements	1 week
1.2	Preparing requirements analysis document	1 week
2	**System design**	3 weeks
2.1	Judicial User Interface Design	2 weeks
2.2	Case database design	1 week
3	**System development**	8 weeks
3.1	Development of the case submission unit	3 weeks
3.1.1	Programming the case submission interface	1.5 weeks
3.1.2	Perform data validation	1.5 weeks
3.2	Development of the case tracking unit	3 weeks
3.2.1	Programming the display interface of issues	1.5 weeks

The number	Mission	Duration
3.2.2	Implement the issue search function	1.5 weeks
3.3	Develop session management unit	2 weeks
3.3.1	Session scheduling interface programming	1 week
3.3.2	Implement the system to send notifications	1 week
4	**System Test**	3 weeks
4.1	Prepare a test plan	1 week
4.2	Conduct functional tests	1 week
4.3	Conduct performance tests	1 week
5	**User training**	2 weeks
5.1	Preparing training materials	1 week
5.2	Organizing training sessions	1 week
6	**System launch**	2 weeks
6.1	Setting up the production environment	1 week
6.2	Perform data transfer	1 week

The number	Mission	Duration
7	**Maintenance and Support**	2 weeks (ongoing)
7.1	Preparing a maintenance plan	1 week (starting after (launch
7.2	Providing technical support	Continuing after launch
8	**Reporting**	2 weeks
8.1	Develop a reporting function	1 week
8.2	Create dashboards	1 week
9	**Infrastructure Security**	2 weeks
9.1	Network Security	1 week
9.2	Software update	1 week

2- *Timeline:*



	الاسم	المدة	البداية	النهاية
1	تحليل المتطلبات	14 days	08:00 ص 26/11/24	05:00 م 13/12/24
2	تصميم النظام	20 days	08:00 ص 13/12/24	05:00 م 09/01/25
3	تطوير النظام	56 days	08:00 ص 10/01/25	05:00 م 28/03/25
4	اختبار النظام	21 days	08:00 ص 31/03/25	05:00 م 28/04/25
5	تدريب المستخدمين	14 days	08:00 ص 29/04/25	05:00 م 16/05/25
6	إطلاق النظام	14 days	08:00 ص 19/05/25	05:00 م 05/06/25
7	الصيانة والدعم	14 days	08:00 ص 06/06/25	05:00 م 25/06/25
8	الأمن وحماية البيانات	28 days	08:00 ص 16/12/24	05:00 م 22/01/25
9	الاختبار والتقارير	14 days	08:00 ص 31/03/25	05:00 م 17/04/25
10	جدولة المهام	7 days	08:00 ص 10/01/25	05:00 م 20/01/25
11	إدارة الموارد	7 days	08:00 ص 10/01/25	05:00 م 20/01/25
12	استجابة الحوادث	14 days	08:00 ص 06/06/25	05:00 م 25/06/25
13	تقييم الأمان المستمر	14 days	08:00 ص 06/06/25	05:00 م 25/06/25
14	إعداد التقارير	14 days	08:00 ص 06/06/25	05:00 م 25/06/25
15	تأمين البنية التحتية	14 days	08:00 ص 06/06/25	05:00 م 25/06/25

Activity 6

1. ***Title :*** **Case Management System Methodologies and Tools**
2. ***Selected development methodology:***

Methodology: Software Development Life Cycle **(SDLC)**

Justification for Choice:

- **Structured Approach:** SDLC provides a clear and systematic framework for the development process, ensuring that all stages from planning to deployment are addressed.
- **Risk Management:** By identifying and mitigating risks at each phase, SDLC helps ensure a smoother development process and more reliable outcomes.
- **Clear Documentation:** Each phase produces documentation that aids communication among team members and stakeholders, which is crucial for the Case Management subsystem due to its complexity.
- **Phases of SDLC:**
  - **Planning:** Define project scope and objectives.
  - **Requirements Gathering:** Collect and analyze user requirements.
  - **Design:** Create system architecture and design specifications.
  - **Implementation:** Develop the subsystem based on design documents.
  - **Testing:** Conduct thorough testing to ensure functionality and security.
  - **Deployment:** Deploy the subsystem and provide user training.
  - **Maintenance:** Provide ongoing support and updates.

3. Development tools:

Version control:

Git: Used to track changes in source code and collaborate between team members:

Integrated Development Environment (IDE):

Visual Studio Code: A versatile code editor that supports multiple programming languages and extensions to improve productivity.

project management:

Trello orJIRA: Tools to manage tasks, track progress, and facilitate communication within the team:

Test frames:

JUnit To test : Java application units , ensuring that each component works correctly.

Selenium: For automated testing of web applications, allowing regression testing and ensuring overall system reliability.

Authentication tools:

Confluence orGoogle Docs: To create and share project documents, facilitating collaboration and knowledge sharing among team members.

Activity 7

1. User roles and tasks:

User role	Tasks/ jobs	Basic Security Concepts
Responsible	Manage user accounts and supervise system settings	Authentication and authorization
He rules on	Review cases, issue and access case,rulings files	Confidentiality and integrity
lawyer	Filing cases, accessing legal documents, and communicating with clients	Confidentiality and authorization
employee	Data entry, schedule management, and case processing support	Integrity and availability
The litigant	Submit cases, receive notifications, and access case status	Confidentiality and availability

2. Data classification and identification of sensitive information:

Data category	Sensitive information	Basic Security Concepts	Security techniques
Case files	Legal documents and personal details	Confidentiality and integrity	Encryption and digital signatures
User information	Personal Identification and Contact Details	Confidentiality and integrity	,Encryption access controls
Notifications	Schedule Changes and Court Dates	Availability	Secure Messaging

3. Basic security procedures and concepts:

procedure	importance	Security risks	Basic Security Concepts	Security techniques
Data entry	provide accurate information	Data entry errors, unauthorized access	Integrity and Authenticity	Input validation and role-based access control
Case approval	Validate and approve cases	Fraudulent Case Approvals	Integrity and Delegation	Digital Signatures and Registration
notice	Notify users of status updates	Information leaks, missing updates	Confidentiality and availability	Secure Messaging and Access Control

4. Basic security concept techniques for each role, data and procedure:

User role	Data category	procedure	Basic Security Concepts	Techniques/ Measures
Responsible	User information	Data entry	Authentication and authorization	Multi-factor authentication and role-based access control
judge on	Case files	Case approval	Confidentiality and integrity	Document encryption and digital signatures
lawyer	Case files	Case approval	Confidentiality and authorization	Encryption for communications, access controls
employee	User information	Data entry	Integrity and availability	Validate entries and backups
The litigant	Notifications	Notifications	Confidentiality and availability	Secure email and notification logs

Exercise 1

- **Decision Matrix for SDLC Methodology**

Criteria	Options	Selected Option	Rationale (Justify your choice)
Team Size	Small, Medium, Large	Small	With a team of 5 members, a small team structure allows for tight collaboration and effective communication.
Requirement Stability	Stable, Unstable, Mixed	Stable	Clearly defined requirements will minimize changes during the project, aiding in predictability and planning.
Project Complexity	Low, Medium, High	Medium	The project involves moderate complexity due to integration needs and technical challenges associated with the judicial system.
Time Sensitivity	High, Medium, Low	Medium	Some flexibility in timelines allows for thorough development and testing without compromising quality.
Security Requirements	High, Medium, Low	High	Given the sensitivity of legal data, high security is essential to protect against breaches and ensure confidentiality.
Expected Changes	Frequent, Occasional, Rare	Occasional	Minor changes are anticipated but won't be frequent, allowing for a stable development environment.

Stakeholder Involvement	High, Medium, Low	High	Active stakeholder engagement is crucial for addressing the needs of the judicial system effectively.
Project Size	Small, Medium, Large	Medium	A medium-sized project scope will balance complexity with manageable timelines and resources.
Risk Tolerance	High, Medium, Low	Medium	A balanced approach to risk allows for innovation while ensuring stability and predictability in processes.
Project Duration	Short (<6 months), Medium (6–12 months), Long (>12 months)	Medium (6-12 months)	A timeframe of 6-12 months is suitable for thorough development and testing of the subsystem.
Flexibility	High, Medium, Low	Medium	Some flexibility is needed to adapt to changes while still adhering to the overall project plan.
Deliverables	Iterative, Final Deliverable, Mixed	Final Deliverable	A final deliverable approach will ensure that all components are fully integrated and tested before deployment.
Customer Feedback	Frequent, Occasional, Rare	Occasional	Feedback will be collected at specific stages to ensure alignment with stakeholder expectations.

Selected Methodology: SDLC

Justification:

- **Team Size Consideration:** With a small team of 5 members, the SDLC methodology allows for structured phases that facilitate collaboration and communication.
- **Clear Requirements:** The stable requirements are ideal for a linear approach like SDLC, where phases such as requirements gathering, design, implementation, and testing are clearly defined.
- **Medium Complexity:** The SDLC is suitable for projects with medium complexity, as it provides a systematic approach to handle integration needs and technical challenges.
- **Security Focus:** Given the high security requirements, SDLC ensures that security measures are integrated into each phase, especially during design and testing.
- **Stakeholder Engagement:** High stakeholder involvement throughout the SDLC phases ensures that the subsystem meets the judicial system's needs effectively.
- **Final Deliverable:** A final deliverable approach ensures that all components are fully integrated and tested before deployment, aligning with the project's structured nature.

Exercise 2

- ***Apply Secure SDLC Practices***

SDLC Phase	Secure SDLC Practice	Description
Requirements	**Threat Modeling**	Identify potential threats to the document management system, including unauthorized access and data breaches. Clearly define security requirements alongside functional requirements.
Design	**Secure Design Principles**	Implement principles such as least privilege, data encryption, and secure storage for sensitive documents. Ensure that the architecture supports secure e-signature functionality.
Development	**Code Review**	Conduct regular code reviews focused on security vulnerabilities, such as SQL injection and cross-site scripting (XSS). Implement secure coding practices throughout development.
Testing	**Security Testing**	Perform penetration testing and vulnerability assessments to identify security flaws. Include tests for data encryption, access controls, and e-signature verification.
Deployment	**Secure Deployment Practices**	Ensure that the deployment environment is secure, including proper configuration of firewalls and access controls. Conduct a final security audit before going live.

Exercise 3

- **Determine the User Types for the Subsystem**

User Type / Role	Arabic Meaning	Responsibilities
Judge	قاضٍ	Review and approve case filings, schedule hearings, issue judgments, and oversee the progression of cases in court.
Lawyer	محامي	File legal documents, communicate case updates with clients, present cases in court, and track case progress throughout the judicial process.
Case Manager	مدير القضايا	Assign cases to lawyers, track case statuses, update records on case progress, and coordinate with judges and lawyers to ensure timely resolution.
Clerk	كاتب المحكمة	Manage the filing of case documents, maintain case files, assist in tracking deadlines, and support judges and lawyers with administrative tasks related to case management.

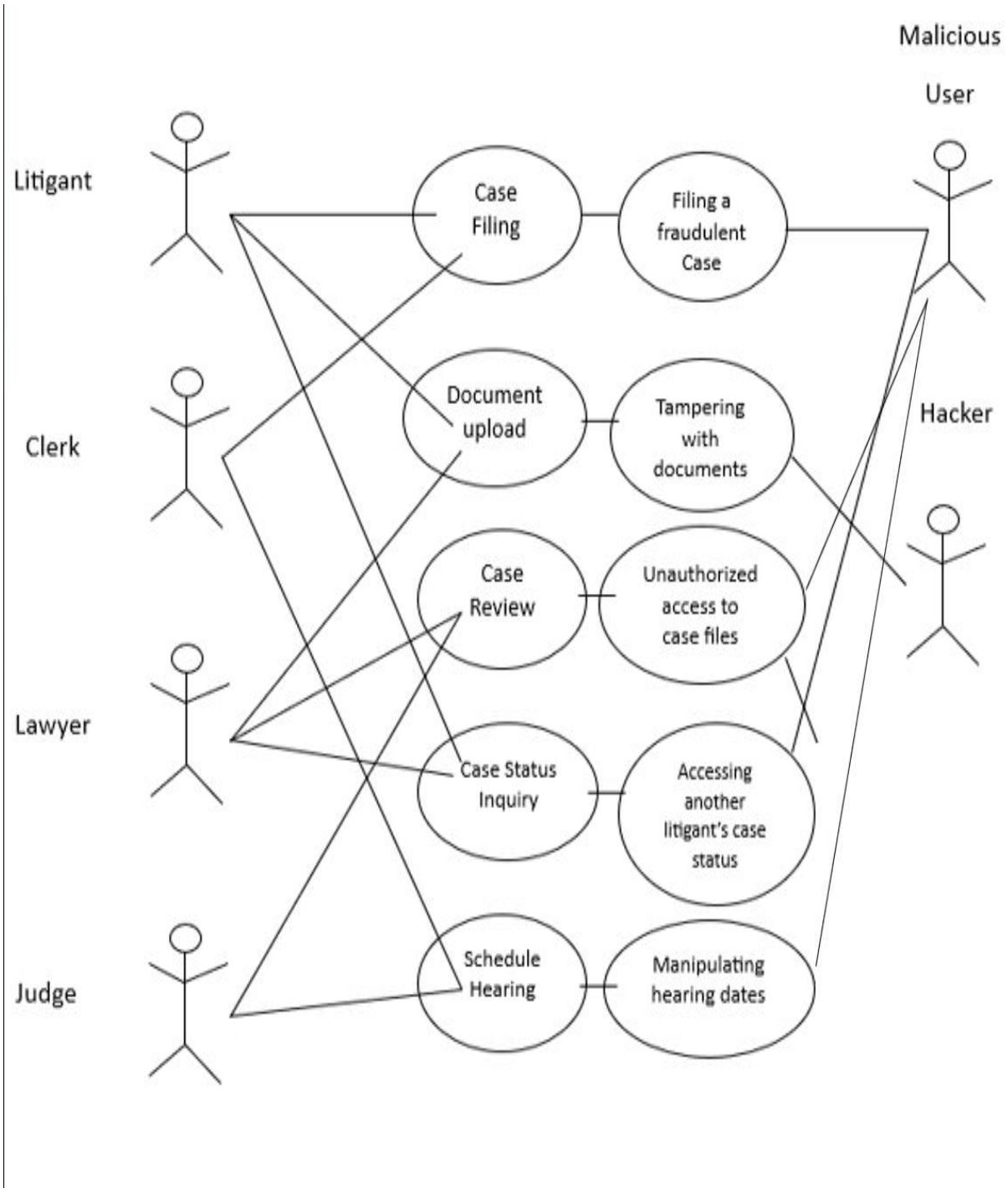
Exercise 4

- **Subject-Object Matrix**

Subject (User Role)	Case Details	Uploaded Documents	Schedules	Court Decisions	Hearing Slots
Judge	Read, Update	Read, Access	Read, Access	Read	Read, Access
Lawyer	Read, Update	Read, Update	No Access	No Access	Read, Access
Litigant	Read	No Access	No Access	No Access	Read
Clerk	Read	Read, Update	No Access	No Access	No Access
Case Manager	Read	No Access	No Access	No Access	No Access

Exercise 5

- **Misuse Case and Use Case Modeling**



Software Requirements Specification

Use the following templates to write the Requirements of the system

Templates:

1. The <system name> shall <system response>.
2. WHILE <precondition> the <system name> shall <system response>.
3. WHEN <trigger> the <system name> shall <system response>.
4. WHERE <feature is included> the <system name> shall <system response>.
5. IF <trigger> THEN the <system name> shall <system response>.
6. WHILE <precondition> WHEN <trigger> the <system name> shall <system response>.

Functional Requirements

In this section, the functional requirements are collected and formulated.

ID	Requirements Description
1	The system shall allow judges to manage case details.
2	The system shall enable lawyers to upload legal documents.
3	The system shall provide litigants with access to case status.
4	The system shall allow clerks to verify documents.
5	The system shall enable case managers to schedule hearings.

Non-Functional Requirements

In this section, the non-functional requirements are identified and formulated.

ID	Requirements Description
1	The system shall ensure data encryption for all sensitive data.
2	The system shall maintain a response time of under 2 seconds for user requests.
3	The system shall be available 99.9% of the time, excluding scheduled maintenance.

Security Requirements

In this section, the non-functional requirements are identified and formulated.

Confidentiality Requirements

The confidentiality requirements must be identified in this section.

ID	Requirements Description
1	The scheduling system shall ensure that only authorized users can **view** schedules.
2	Sensitive information in schedules must be encrypted during transmission.
3	User access to schedules shall be logged for auditing purposes.

Integrity Requirements

The Integrity requirements must be identified in this section.

ID	Requirements Description
1	The scheduling system shall ensure that only authorized users can **modify** schedules.
2	Any changes to schedules must be tracked and logged for integrity verification.
3	The system shall validate all data inputs to prevent corruption.

Availability Requirements

The Integrity requirements must be identified in this section.

ID	Requirements Description
1	The scheduling system shall ensure 99.9% uptime to maintain accessibility.
2	The system shall have backup mechanisms in place to recover from failures.
3	Scheduled maintenance must be communicated to users in advance to minimize disruption.

Authentication Requirements

The authentication requirements must be identified in this section.

ID	Requirements Description
1	The system shall require users to authenticate with a username and password.
2	Multi-factor authentication must be implemented for all user accounts.
3	User sessions shall automatically expire after a period of inactivity.

Authorization Requirements

The authorization requirements must be identified in this section.

ID	Requirements Description
1	The system shall ensure that users have role-based access controls to limit actions based on their roles.
2	Users must only access the resources necessary for their job functions.
3	The system shall provide an audit trail of all authorization changes.

Non-repudiation Requirements

The non- repudiation requirements must be identified in this section.

ID	Requirements Description
1	The system shall log all user actions to provide evidence of transactions.
2	Users must receive confirmation of actions taken, such as modifications or deletions, to prevent denial of actions.
3	Digital signatures must be used for critical transactions to verify the identity of the user.

Accountability Requirements

The accountability requirements must be identified in this section.

ID	Requirements Description
1	The system shall maintain detailed logs of all user activities for auditing purposes.
2	Users must be uniquely identifiable to ensure that actions can be traced back to them.
3	The system shall provide mechanisms for reviewing and managing access logs regularly.

Threat Model

In this document, we will provide a threat modeling about [Case Management].

Scope

In this section, we will To improve how judicial cases are managed with an integrated system that helps to follow procedures and access information.

Information

Application Name	Case Management
Application Version	Judicial Case Management System
Description	an integrated system whose purpose is to manage the cases in the juristic system. Easy to use interface for recording, following up cases, managing legal docs and Communicating amongst the parties concerned.
Document Owner	Mohammed Alqmase
Participants	1. *Majed Bagash* 2. *Mohammed Al-Mugalees* 3. *Ahmed Al-Hakeemy* 4. *Ramzi Al-Qubaty* 5. *Nehal Al-Odaini*
Reviewer	Fahd Al-Mughalles

Dependencies

ID	External Dependencies Description
1	**Case Database:** Storage of all case and stakeholder data
2	**API:** Integration with other systems such as criminal records
3	**document management system:** Storage and management of legal documents associated with cases
4	**Authentication system:** Provide an additional level of security by confirming users' identity

Entry Points

ID	Name	Description	Trust levels
1	**user interface**	The interactive interface used by lawyers and judges to manage cases.	high
2	**API**	API for integration with external systems such as court records.	medium
3	**mobile app**	A dedicated application for users to access information and manage issues.	high
4	**Safe Access Portal**	Authentication and identity verification system for users.	high

Exit Points

ID	Name	Description	Trust levels
1	**Judges' Reports**	Reports are established to follow up and manage the status of cases.	high
2	**email notifications**	Notifications sent to users regarding issue updates.	medium
3	**Export Interface**	The possibility of exporting data to other systems or formats.	medium
4	**Exit Interface**	The process of logging out of the system securely.	high

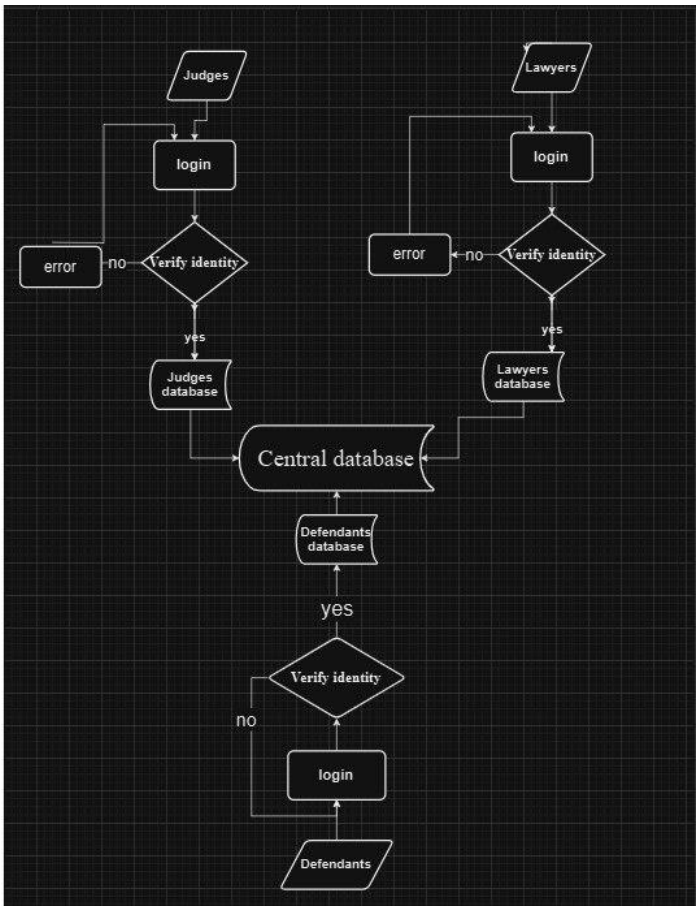
Assets

ID	Name	Description	Trust levels
1	**Case Database**	A database containing all case information, stakeholders and documents.	high
2	**users' data**	Users' information such as entry data, roles, and personal information.	high
3	**legal documents**	all documents related to issues such as complaints, replies, and judgments.	high
4	**user interface**	The interface with which users interact to manage issues.	Medium

Trust Levels

ID	Name	Description
1	**high-level**	Accredited users such as judges and lawyers with full access to sensitive data.
2	**intermediate level**	Support staff or staff of the system who need access to limited information.
3	**low-level**	Visitors or unregistered users who have access to general information only.
4	**special level**	Users with special roles such as investigators or experts who need specific access for certain purposes.

Data Flow Diagrams



Break

In this section, we will analyze the weaknesses of the case management system. To ensure maximum security of computer software or hardware, it is essential to know their vulnerabilities. Every vulnerability shall be assessed based on its impact, likelihood of occurrence and remedial measures

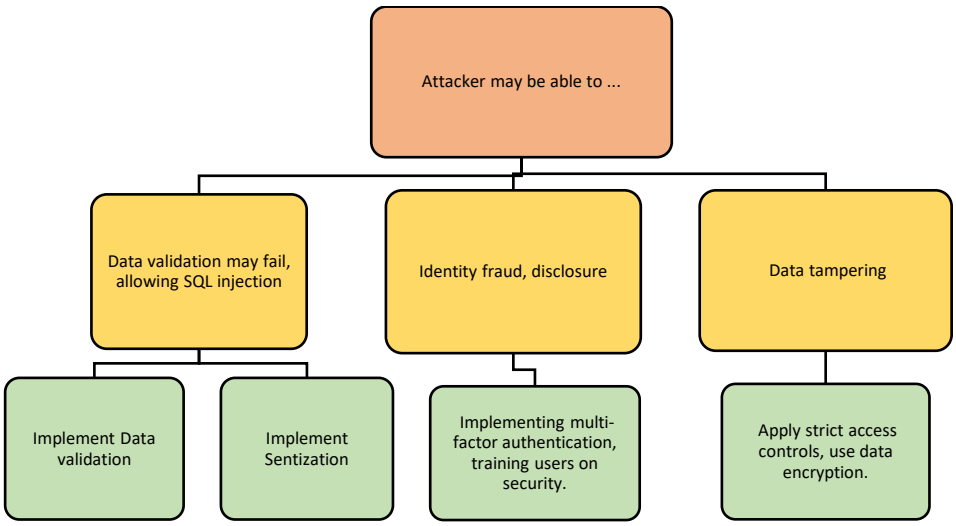
STRIDE Framework

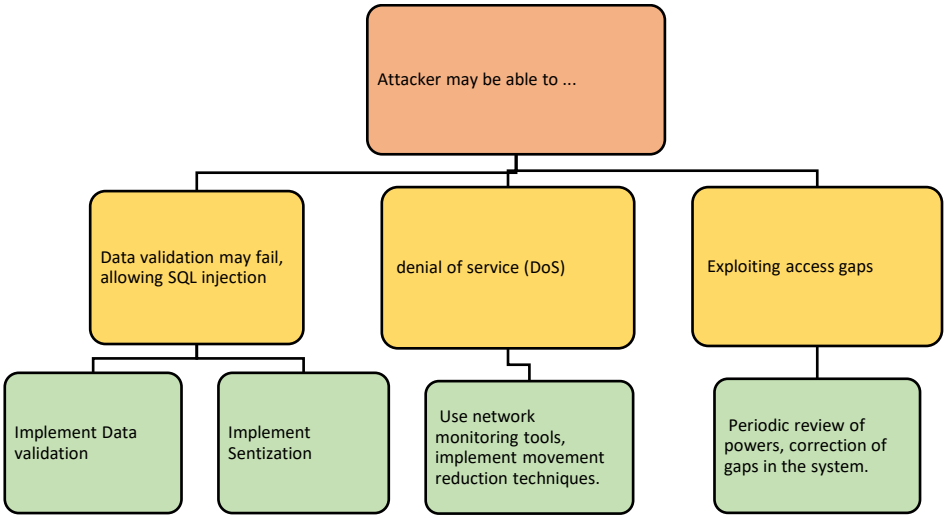
ID	Threats types	threats Description	Security control types
1	**Spoofing**	1. impersonate a user for unauthorized access. 2. Loss of data, violation of privacy.	Authentication
2	**Tampering**	1. Modify data or system settings unauthorized. 2. Impact on the integrity of information, loss of trust.	Integrity
3	**Repudiation**	1. Deny the user to do a certain job such as making a case. 2. Loss of records, inability to track procedures.	Non- Repudiation
4	**Information Disclosure**	1. Leak sensitive information to unauthorized parties. 2. Violation of privacy, legal damages.	Confidentiality
5	**Denial of Service**	1. Attacks aimed at making the system unavailable to legitimate users. 2. Loss of productivity, inability to access services.	Availability
6	**Elevation of Privileges**	1. Obtaining unauthorized higher powers. 2. Data violation, loss of control over the system.	Authorization

Threat Analysis

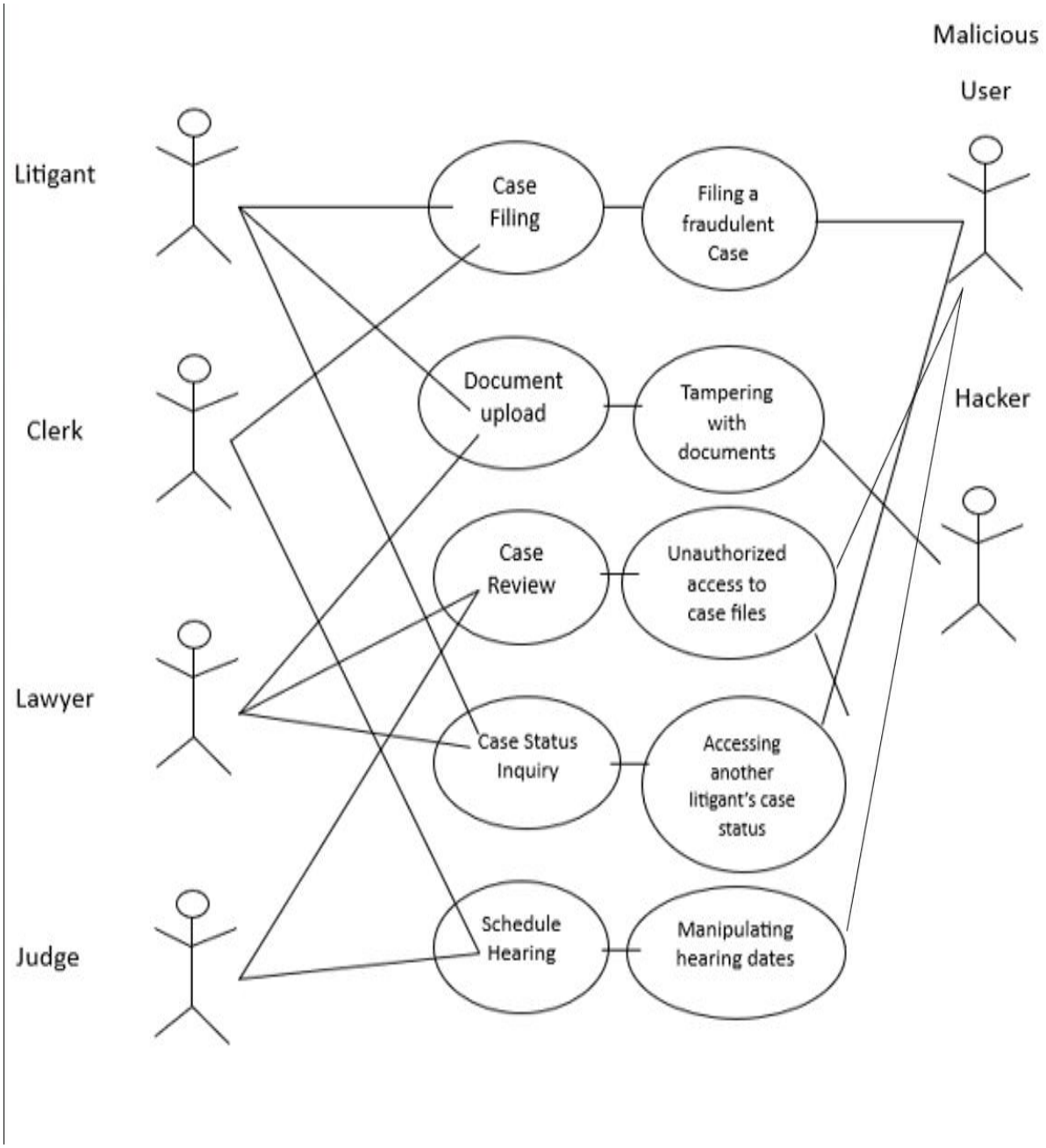
In this section, we will conduct a comprehensive threat analysis to identify, assess, and prioritize potential threats to the case management system. This analysis will help us understand the vulnerabilities within the system and the possible impacts of various threats.

Attack Trees





Misuse Cases



Threat Description Table

Threat ID	Threat Description	Threats Types
1	Fraudulent user identity to access case data	Spoofing
2	Disclosure of sensitive case information to unauthorized entities	Spoofing
3	Unauthorized modification of case statements	Tampering
4	Disruption of the system preventing access to judicial services	Tampering
5	Raise user powers to access sensitive data	Information Disclosure
6	Disclaimer of acts such as submitting cases or amending statements	Denial of Service

Ranking

In this section, we will Rank the identified threats by how serious and likely they might occur. By priority ranking this development in terms of gravity probability the mitigation steps can be directed towards the most impactful threats.

Delphi Ranking

Threat ID	Threat Title	Member 1 Rank	Member 2 Rank	Member 3 Rank	Average Rank	Final Consensus Rank	Comments
1	**User Identity Fraud**	2	1	2	1.5	1	Very high threat
2	**disclosing sensitive information**	1	2	1	1.5	1	Requires strict safety procedures
3	**Amending case data**	3	3	3	3	3	Average risk
4	**disable the system**	1	2	1	1.5	1	Significant impact or operations
5	**User permissions *power***	2	3	2	2.5	2	Powers must be monitored
6	**Disclaimer**	3	3	3	3	3	less risky

Average Ranking

Threat ID	Threat Title	D	R	E	A	Average Rank	Risk Levels
1	**User Identity Fraud**					1.5	Very high
2	**reveal sensitive information**					1.5	Very high
3	**Modify case data**					3	medium
4	**disable the system**					1.5	Very high
5	**User permissions *power***					2.5	high
6	**Disclaimer**					3	low

Probability x Impact (P x I) Ranking

Threat ID	Threat Title	P probability	I Impact	Risk Score P x I	Rank
1	**User Identity Fraud**	4	5	20	1
2	**reveal sensitive information**	4	5	20	1
3	**Modify case data**	3	4	12	3
4	**disable the system**	4	5	20	1
5	**User permissions *power***	3	3	9	4
6	**Disclaimer**	2	2	4	6

Remarks:

PROBABILITY: Ranges from 1 (low) to 5 (high).

IMPACT: ranges from 1 (low) to 5 (high).

RISK SCORE: Calculated as follows: PROBABILITY × IMPACT.

Fix

In this section, we will outline the strategies and actions necessary to mitigate the identified threats to the case management system. Our goal is to implement effective fixes that reduce vulnerabilities and enhance the overall security posture of the system.

Threat ID	T#001
Threat Description	User identity fraud Attempt unauthorized access to the system using a false identity.
Threat targets	1. users' data 2. Case records
Attack techniques	1. phishing attack 2. Exploiting passwords 3. Use of identity manipulation tools
Security Impact	1. Loss of sensitive data 2. Destroy trust between users and the system 3. Negative impact on legal processes
Risk	very high

Safeguard controls to implement	1. Implementation of Role Based Access Control System (RBAC) 2. Multi-factor authentication application (MFA) 3. Monitoring and recording of suspicious activities

Threat ID	T#002
Threat Description	Leaking sensitive data from the system to unauthorized entities.
Threat targets	1. Case Information 2. customer details
Attack techniques	1. Gaps-exploiting attacks 2. unauthorized access 3. malware attacks
Security Impact	1.Loss of sensitive data 2. Legal implications 3. Destroy the reputation of the enterprise
Risk	high
Safeguard controls to implement	1. Encryption of sensitive data 2. Implement strict safety policies 3. Regular security audits

Threat ID	T#003
Threat Description	Unauthorized change or deletion of case statements.
Threat targets	1. Case records 2. Users' Information
Attack techniques	1. Unauthorized access 2. Exploiting system gaps 3. Using data manipulation tools
Security Impact	1. Negative impact on data integrity 2. Loss of trust in the system 3. Legal implications
Risk	high
Safeguard controls to implement	1. Implementation of regular data audits 2. Use of digital signature techniques 3. Application of accurate access control system

Verify

In this section, we will outline the processes and methods used to verify the effectiveness of the implemented fixes and ensure that the case management system is secure and functioning as intended. This includes reviewing documentation, testing the system, and validating the results.

Review Documentation

We will conduct a thorough review of all relevant documentation, including:

- **Security Policies**: Ensure that all security policies are updated and reflect the current practices.
- **User Access Logs**: Analyze logs to verify that access controls are being enforced appropriately.
- **Incident Reports**: Review past incidents to identify any recurring issues and ensure lessons learned are documented.

Test cases

We will develop and execute test cases to verify that the implemented fixes are effective. This will include:

- **Functional Tests**: Verify that all system functionalities are working as intended after the fixes.
- **Security Tests**: Conduct penetration testing and vulnerability assessments to identify any remaining weaknesses.
- **User Acceptance Testing (UAT)**: Involve end-users to validate that the system meets their needs and expectations.

Validation

Validation will involve confirming that the system meets all defined requirements and standards. This will include:

- **Compliance Checks**: Ensure that the system complies with relevant legal and regulatory requirements.

- **Performance Metrics**: Measure system performance against established benchmarks to ensure that it operates efficiently.
- **Feedback Mechanism**: Collect feedback from users to identify any issues or areas for improvement.

Creating UML and ERD Diagrams

This lab provides hands-on experience in creating UML and ERD diagrams for subsystem modeling. It ensures students can visualize and design the architecture of their assigned subsystem effectively, bridging the gap between analysis and implementation.

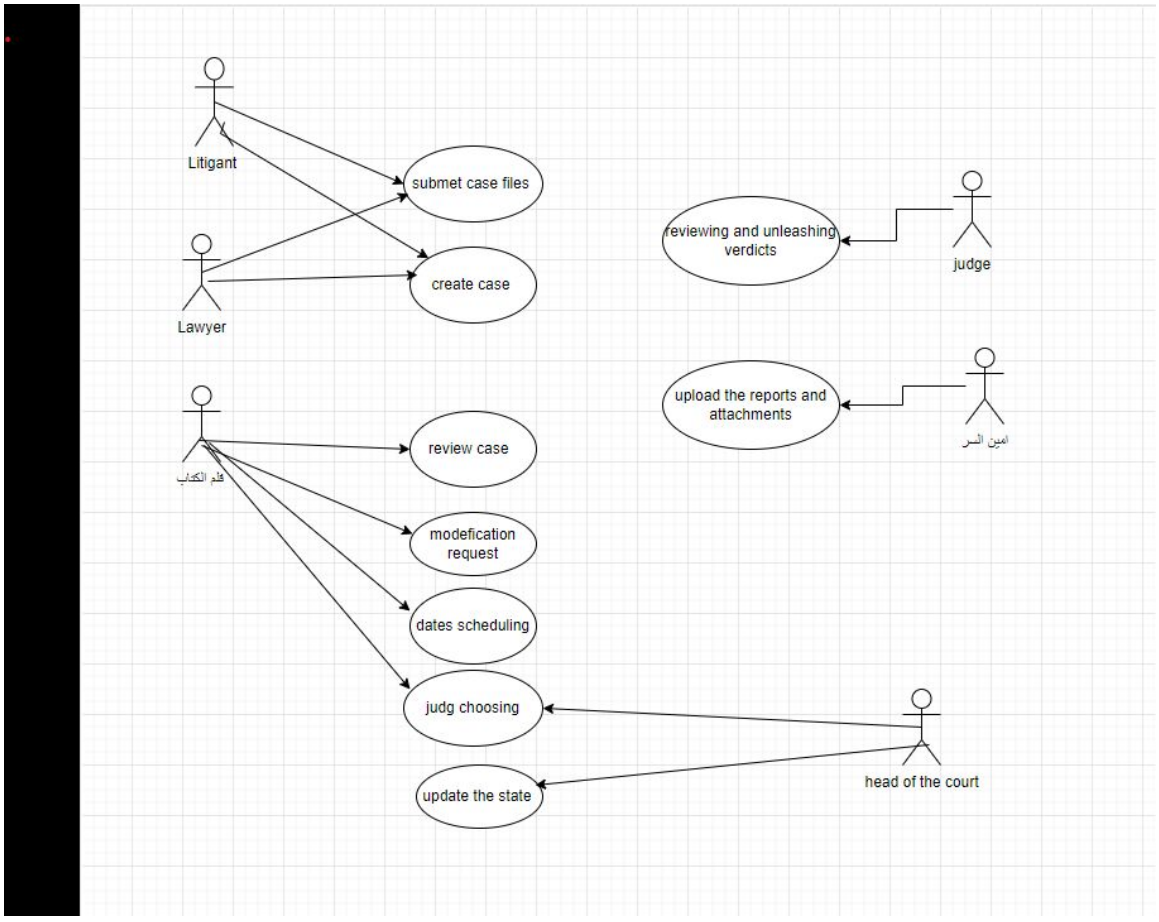
Tools Required:

- draw.io (Diagrams.net)

Task 1: Create Use Case Diagram (30 minutes)

1. Identify the core use cases for the subsystem.
    - Example: For "Litigant Portal," use cases may include “Check Case Status,” “File Complaint,” and “Receive Notifications.”
2. Draw a Use Case Diagram showing the relationships between actors and use cases.
    - Include system boundaries and extend/include relationships where applicable.

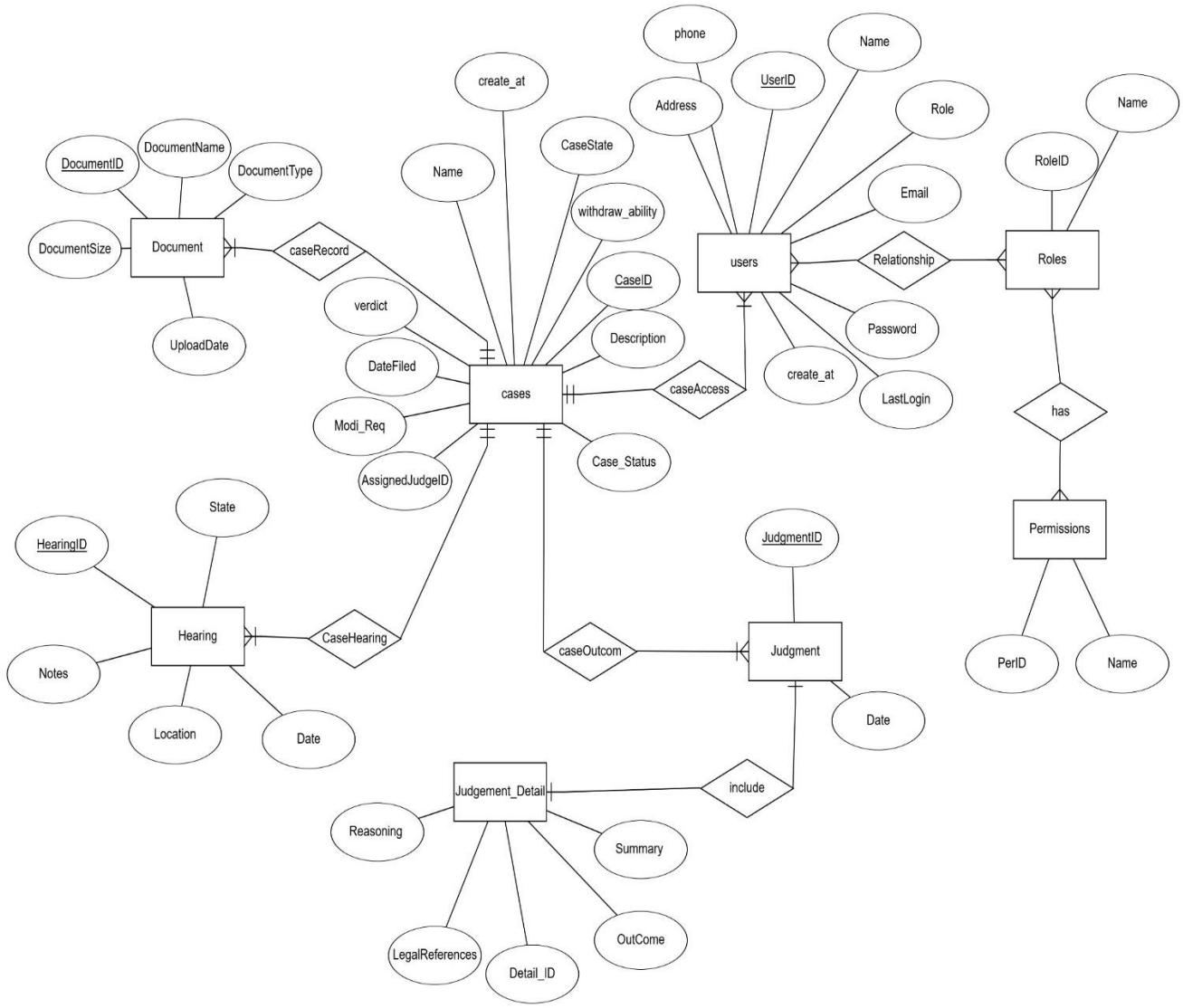
Deliverable: A Use Case Diagram highlighting the core functionalities of the subsystem.



Task 2: Create an Entity-Relationship Diagram (ERD) (40 minutes)

1. Identify the key entities within the subsystem and their relationships.
    - Example: For "Document Management," entities may include Document, Case, User, and Access Log.
2. Define attributes for each entity and primary/foreign keys for relationships.
3. Ensure cardinality (1:1, 1:N, N:M) is correctly represented.

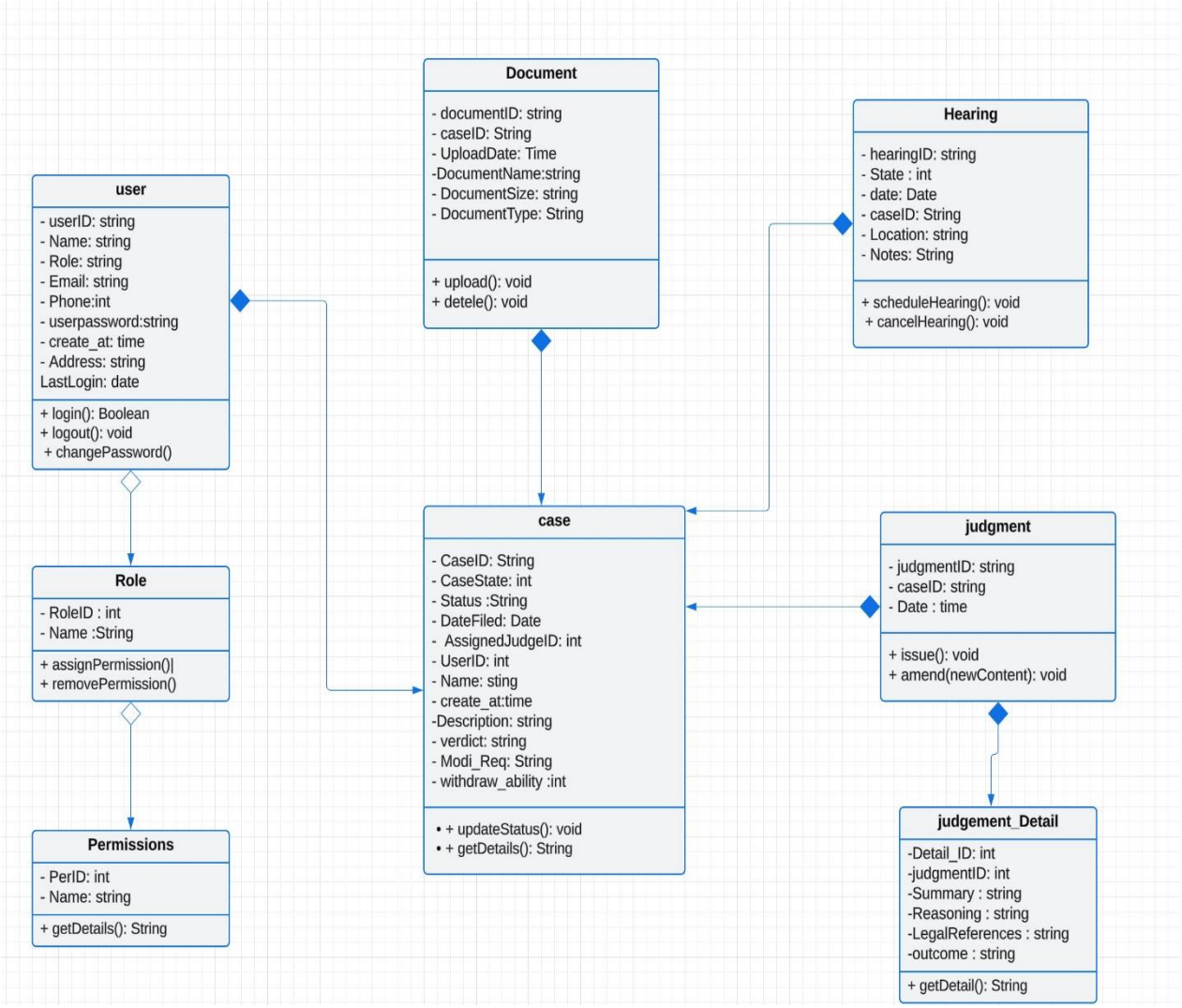
Deliverable: A complete **ERD** showing the relationships between entities in the assigned subsystem.



Task 3: Design a Class Diagram (30 minutes)

1. Identify key classes, their attributes, and methods for the subsystem.
    - Example: For "Security and Role Management," classes may include User, Role, and Permission.
2. Define relationships such as inheritance, aggregation, and composition.
3. Include visibility markers (public, private, protected) for attributes and methods.

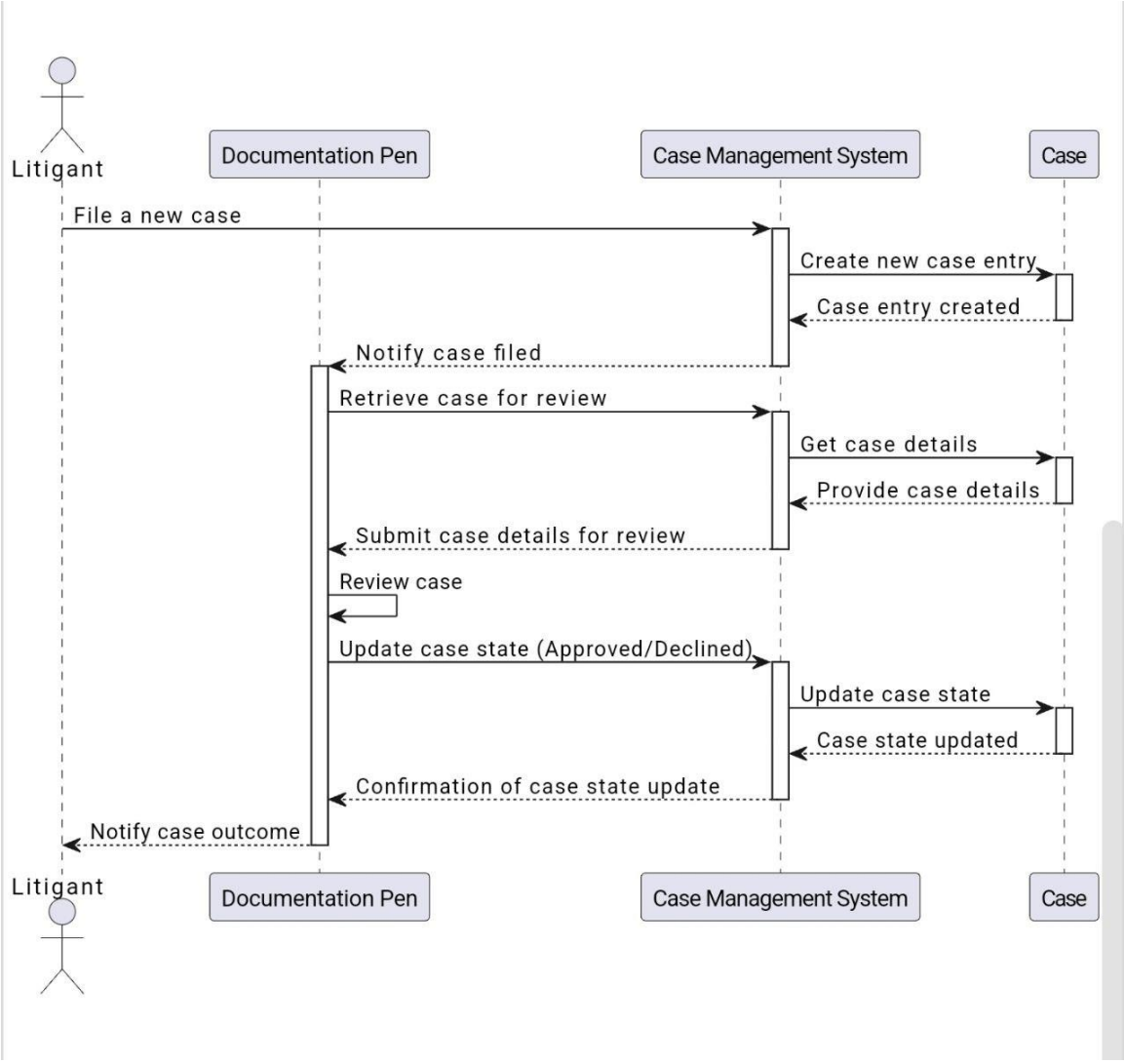
Deliverable: A **Class Diagram** detailing the structure of the subsystem.



Task 4: Create a Sequence Diagram (30 minutes)

1. Select one core functionality of the subsystem.
    - Example: For "Courtroom Scheduling," the functionality could be “Assign Judge to a Hearing.”
2. Illustrate the interaction between objects/actors over time.
    - Include lifelines, messages, and activation boxes.

Deliverable: A **Sequence Diagram** showing the flow of actions for the selected functionality.



From Bad Practices to Best Practices
Authentication System for JMS

Objectives:

1. Teach students to identify threats and vulnerabilities in a poorly implemented authentication system.
2. Enable students to understand why specific practices are insecure and how they can lead to security risks.
3. Guide students to reimplement the system following secure coding best practices.

Tools Required:

- PHP Development Environment (XAMPP).
- Browser Developer Tools for testing client-side vulnerabilities.
- MySQL database.
- VS Code for coding.
- Security guidelines like OWASP Top 10 for reference.

Lab Activities

1 Part 1: Analyze Bad Practices in Authentication

1.1 Activity 1.1: Explore the Bad Practice Implementation

- Students will implement PHP scripts for:
- Registration Form: Stores plain text passwords.
- Login Page: Vulnerable to SQL Injection.
- Logout Page: Does not securely clear sessions.
- Welcome Page: Does not validate sessions.
- Home Page: Does not redirect authenticated users.

Tasks:

- Deploy the provided scripts and test their functionality.
- Identify security issues such as:
- Plain text password storage.
- Lack of input validation.
- Session management flaws.

Deliverable:

A list of identified vulnerabilities with explanations.

Summary Table of Vulnerabilities

Component	Vulnerability	Explanation
Registration Form	SQL Injection	The Registration Form uses direct concatenation of user inputs into the SQL query: $sql = "INSERT INTO users (name, username, password) VALUES ('$name', '$username', '$password')"; This allows an attacker to manipulate the SQL query by providing malicious input. For instance, if an attacker inputs '); DROP TABLE users;-- as the username, it could lead to destructive actions on the database, such as deleting the users table. This vulnerability can be mitigated by using prepared statements.
Login Page	SQL Injection	Similar to the Registration Form, the Login Page is also vulnerable to SQL injection: $sql = "SELECT * FROM users WHERE username='$username' AND password='$password'"; An attacker could input admin' -- as the username to bypass the password check. This could allow

		unauthorized access to user accounts. To prevent this, prepared statements should be used, and passwords should be hashed and verified securely.
Logout Page	Session Fixation	The Logout Page destroys the session but does not regenerate a new session ID afterward. If an attacker can set a session ID before the user logs in, they can hijack the session after the user logs in. It's important to regenerate the session ID upon login and logout to mitigate this type of attack.
Welcome Page	Lack of Session Validation	The Welcome Page does not check if the user is logged in before displaying the username: Welcome to the Judicial Management System, <?= $_SESSION['username'] ?> ! If a user accesses this page without being logged in, it could lead to an "undefined index" error or display incorrect information. It is crucial to validate the session and ensure the user is authenticated before displaying sensitive information.
Home Page	Lack of Access Control	The Home Page allows any user to access it but does not implement any access control measures. While this may not seem like a vulnerability, it can lead to information exposure if sensitive functionalities or links are presented to unauthorized users. For example, links to log in or register should be available, but if there are hidden functionalities, they should be protected based on user authentication status.

1.2 Activity 1.2: Test the System for Vulnerabilities

- SQL Injection: Manipulate login credentials (' OR '1'='1) to bypass authentication.
- Session Hijacking: Manually copy session cookies and reuse them.
- Password Exposure: Check for plain text passwords in the database.

Tasks:

- Perform penetration tests on the system.
- Document how each vulnerability could be exploited in a real-world scenario.

Deliverable:

A report with screenshots showing exploited vulnerabilities.

Vulnerability Assessment Report

1. Overview

This report documents the vulnerabilities found in the Judicial Management System through penetration testing, including SQL Injection, Session Hijacking, and Password Exposure.

2. Vulnerabilities

2.1 SQL Injection

Description:

SQL Injection allows attackers to manipulate SQL queries by injecting malicious input.

Exploit Method:

1. During the login process, input the following credentials:

- Username: `admin' OR '1'='1`

- Password: `anything`

2. This results in the following SQL query:

sql

*SELECT * FROM users WHERE username='admin' OR '1'='1' AND password='anything';*

3. The condition `'1'='1'` always evaluates to true, allowing unauthorized access.

Impact:

An attacker could gain access to any user account, including administrative accounts, leading to unauthorized actions within the system.

Screenshot:

(Insert screenshot of the login attempt with the SQL injection payload.)

2.2 Session Hijacking

Description:

Session Hijacking occurs when an attacker gains unauthorized access to a user's session by stealing session cookies.

Exploit Method:

1. Log in to the application to obtain a session cookie.

2. Open the browser's developer tools and navigate to the "Application" or "Storage" tab to view cookies.

3. Manually copy the session cookie and set it in another browser or tab.

4. Refresh the page to access the application as the original user without needing to log in.

Impact:

An attacker can impersonate a legitimate user, potentially accessing sensitive information and performing actions on their behalf.

Screenshot:

(Insert screenshot of the session cookie and the resulting access to the application.)

2.3 Password Exposure

Description:

Storing passwords in plain text poses a significant security risk.

Exploit Method:

1. Access the database directly (e.g., via a database management tool).

2. Query the `users` table to retrieve stored passwords:

sql

SELECT username, password FROM users;

3. Review the output to identify any plain text passwords.

Impact:

Exposed passwords can be easily exploited by attackers, leading to account takeovers across the application and potentially other services if users reuse passwords.

Screenshot:

(Insert screenshot of the database query showing plain text passwords.)

3. Recommendations

- SQL Injection: Implement prepared statements or parameterized queries to sanitize user inputs.

- Session Management: Use secure, HttpOnly, and SameSite attributes for session cookies, and implement session expiration.

- Password Storage: Hash passwords using a strong hashing algorithm (e.g., bcrypt) before storing them in the database.

4. Conclusion

The Judicial Management System has several critical vulnerabilities that could be exploited by attackers. Immediate action is recommended to mitigate these risks and enhance the overall security of the application.

Additional Tips

- Ensure you have permission to conduct penetration testing on the system.

- Document each step clearly and ethically to maintain the integrity of your testing process.

- Always comply with relevant laws and regulations regarding security testing.

2 Part 2: Reimplement the System with Best Practices

2.1 Activity 2.1: Refactor Registration and Login

- Replace plain text password storage with password_hash().
- Use prepared statements to prevent SQL Injection.
- Add input sanitization and validation.

Tasks:

- Implement password hashing using password_hash().
- Modify login logic to use password_verify().

Deliverable:

- Secure versions of the registration and login pages.

Here are the refactored versions of the Registration and Login pages, incorporating best practices such as password hashing, prepared statements, and input sanitization.

Refactored Registration Form

php

```
<?php

// Database connection

$conn = new mysqli("localhost", "root", "", "jms_db");


// Check connection

if ($conn->connect_error) {

die("Connection failed: " . $conn->connect_error);

}


if ($_SERVER['REQUEST_METHOD'] == 'POST') {

$name = htmlspecialchars(trim($_POST['name']));

$username = htmlspecialchars(trim($_POST['username']));

$password = $_POST['password']; // Plain text password
```

```
// Hash the password
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Use prepared statements
$stmt = $conn->prepare("INSERT INTO users (name, username, password) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $name, $username, $hashed_password);

if ($stmt->execute()) {
echo 'Registration successful! <a href="login.php">Login</a>';
} else {
echo "Error: " . $stmt->error;
}

$stmt->close();
}
?>

<form method="POST">
Name: <input type="text" name="name" required><br>
Email: <input type="text" name="username" required><br>
Password: <input type="password" name="password" required><br>
<button type="submit">Register</button>
</form><br>
<a href="login.php">Login</a>
```

Refactored Login Page

```
php
<?php
session_start();
$conn = new mysqli("localhost", "root", "", "jms_db");

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$username = htmlspecialchars(trim($_POST['username']));
$password = $_POST['password'];

// Use prepared statements
$stmt = $conn->prepare("SELECT password FROM users WHERE username=?");
$stmt->bind_param("s", $username);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
$stmt->bind_result($hashed_password);
$stmt->fetch();

// Verify the password
if (password_verify($password, $hashed_password)) {
$_SESSION['username'] = $username;
echo "Login successful!";
header("Location: dashboard.php");
exit;
```

```
} else {
echo "Invalid username or password.";
}
} else {
echo "Invalid username or password.";
}

$stmt->close();
}
?>

<form method="POST">
Username: <input type="text" name="username" required><br>
Password: <input type="password" name="password" required><br>
<button type="submit">Login</button>
</form>

<a href="register.php">Register</a>
```

Key Changes Made

1. Password Hashing: Used `password_hash()` for storing passwords securely in the database.

2. Prepared Statements: Replaced direct SQL queries with prepared statements to prevent SQL injection.

3. Input Sanitization: Utilized `htmlspecialchars()` and `trim()` to sanitize user inputs, ensuring no malicious code can be injected.

4. Required Fields: Added `required` attributes to form inputs for basic client-side validation.

2.2 Activity 2.2: Improve Session Management

- Regenerate session IDs after login to prevent fixation.
- Properly destroy sessions on logout, including clearing cookies.

Tasks:

- Modify logout script to clear sessions securely.
- Implement session expiration with timeouts.

Deliverable:

- Updated logout script with secure session handling.

Updated Login Page with Session Regeneration

In the login logic, regenerate the session ID after successful login:

php

<?php

session_start();

$conn = new mysqli("localhost", "root", "", "jms_db");

if ($_SERVER['REQUEST_METHOD'] == 'POST') {

$username = htmlspecialchars(trim($_POST['username']));

$password = $_POST['password'];

// Use prepared statements

$stmt = $conn->prepare("SELECT password FROM users WHERE username=?");

$stmt->bind_param("s", $username);

$stmt->execute();

$stmt->store_result();

```
if ($stmt->num_rows > 0) {
$stmt->bind_result($hashed_password);
$stmt->fetch();

// Verify the password
if (password_verify($password, $hashed_password)) {
// Regenerate session ID
session_regenerate_id(true);
$_SESSION['username'] = $username;
$_SESSION['login_time'] = time(); // Store login time
echo "Login successful!";
header("Location: dashboard.php");
exit;
} else {
echo "Invalid username or password.";
}
} else {
echo "Invalid username or password.";
}

$stmt->close();
}
?>
```

Updated Logout Script

Modify the logout script to securely destroy the session and clear cookies:

```
php
<?php
session_start();

// Unset all session variables
$_SESSION = [];

// Destroy the session
session_destroy();

// Clear the session cookie
if (ini_get("session.use_cookies")) {
$params = session_get_cookie_params();
setcookie(session_name(), '', time() - 42000,
$params["path"], $params["domain"],
$params["secure"], $params["httponly"]
);
}

echo "You have been logged out.";
header("Location: home.php");
exit;
?>
```

Implementing Session Expiration with Timeouts

To implement session expiration, add a check on each page where the session is being used:

```
php

// Check for session timeout

session_start();


$timeout_duration = 1800; // 30 minutes


if (isset($_SESSION['login_time'])) {

if (time() - $_SESSION['login_time'] > $timeout_duration) {

// Session expired

session_unset();

session_destroy();

header("Location: login.php?message=Session expired. Please log in again.");

exit;

} else {

// Update login time

$_SESSION['login_time'] = time();

}

}
```

Key Changes Made

1. Session Regeneration: Used `session_regenerate_id(true)` after successful login to prevent session fixation attacks.

2. Secure Logout: The logout script now properly destroys the session and clears the session cookie.

3. Session Timeout: Implemented session expiration logic to log users out after a specified period of inactivity.

2.3 Activity 2.3: Validate Access on Protected Pages

- Add session validation to the welcome page to restrict access to authenticated users.
- Redirect unauthenticated users from the welcome page to the login page.

Tasks:

- Add session checks to the welcome page.
- Test unauthorized access scenarios.

Deliverable:

- Secured welcome page script.

Secured Welcome Page Script

Here's how you can implement session validation on the Welcome Page:

```
php

<?php

session_start();


// Check if the user is authenticated

if (!isset($_SESSION['username'])) {

// Redirect unauthenticated users to the login page

header("Location: login.php?message=Please log in to access this page.");

exit;

}

?>


<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Welcome</title>

</head>
```

```
<body>

<p>Welcome to the Judicial Management System, <?= htmlspecialchars($_SESSION['username']) ?>!</p>

<a href="logout.php">Logout</a>

</body>

</html>
```

Key Changes Made

1. Session Check: The script checks if the `username` session variable is set. If not, it redirects the user to the login page.

2. HTML Escaping: Used `htmlspecialchars()` to prevent XSS by escaping the output of the username.

3. User Feedback: Added a message parameter in the redirect URL to inform the user why they were redirected.

Testing Unauthorized Access Scenarios

1. Access the Welcome Page without Logging In:

- Open the Welcome Page URL directly in the browser.

- You should be redirected to the login page with a message prompting you to log in.

2. Access After Logging In:

- Log in successfully.

- Navigate to the Welcome Page.

- Confirm that you can access the content without being redirected.

2.4 Activity 2.4: Compare Bad and Best Practices

- Compare the functionality and security of the bad and good implementations.
- Highlight the risks mitigated in the secure version.

Deliverable:

- A comparative table showing vulnerabilities in the bad practice implementation and their fixes in the best practices implementation.

Aspect	Bad Practices Implementation	Best Practices Implementation	Risks Mitigated
Password Storage	Plain text password storage.	Uses password_hash() to securely hash passwords.	Prevents password exposure in case of a database breach.
SQL Injection	Directly concatenated user input in SQL queries.	Uses prepared statements to parameterize queries.	Protects against SQL injection attacks.
Input Validation	Minimal or no input sanitization.	Sanitizes input with htmlspecialchars() and trims whitespace.	Mitigates risks of XSS and other injection attacks.
Session Management	No session regeneration after login.	Regenerates session ID using session_regenerate_id(true).	Prevents session fixation attacks.
Session Destruction	Simply calling session_destroy() without clearing cookies.	Clears session data and cookies properly.	Ensures complete logout and prevents session hijacking.
Session Timeout	No session expiration implemented.	Implements session timeout based on inactivity.	Reduces risk of unauthorized access from inactive sessions.

Aspect	Bad Practices Implementation	Best Practices Implementation	Risks Mitigated
Access Control	No checks for user authentication on protected pages.	Session validation checks to restrict access to authenticated users.	Prevents unauthorized access to sensitive pages.
Error Handling	Displays raw error messages, revealing sensitive information.	Generic error messages without revealing specifics.	Reduces risk of information disclosure to attackers.

- **Summary**
- **Functionality**: Both implementations serve the same basic functions (registration, login, session management), but the best practices implementation enhances security significantly.
- **Security**: The secure version mitigates a variety of risks, including SQL injection, password exposure, unauthorized access, and session hijacking.