

**RAJALAKSHMI ENGINEERING  
COLLEGE  
RAJALAKSHMI NAGAR, THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CB23332  
SOFTWARE ENGINEERING LAB**

**Laboratory Record Note Book**

Name : .....

Year / Branch / Section : .....

Register No. : .....

Semester : .....

Academic Year : .....

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**  
**RAJALAKSHMI NAGAR, THANDALAM – 602-105**

**BONAFIDE CERTIFICATE**

**NAME:** \_\_\_\_\_ **REGISTER NO.:** \_\_\_\_\_

**ACADEMIC YEAR:** 2024-25 **SEMESTER:** III **BRANCH:** \_\_\_\_\_ B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the  
**CB23332-SOFTWARE ENGINEERING - Laboratory** during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on \_\_\_\_\_

Internal Examiner

External Examiner

# INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagram		
10.	Class Diagram		

## 1. PREPARING PROBLEM STATEMENT

### **Aim :**

Traditional court case management systems face challenges such as manual processes, lack of real-time updates, and limited digital accessibility, making it difficult for users to track case statuses, manage documentation, and schedule hearings efficiently. Delays, increased administrative burdens, and fragmented access to physical and digital records further hinder operational effectiveness. A Court Case Management System (CCMS) aims to address these issues by automating workflows, integrating physical and digital case records, and enabling real-time case tracking, online documentation management, and automated notifications for hearings and deadlines. This will streamline processes, enhance accessibility, and improve overall judicial efficiency..

### **Algorithm :**

1. Initialize System: Set up the Case Management System, Court Case Database, and Lawyer & Client Database.
2. Client Submits Case Query: The client or lawyer logs in and submits a case-related query. The system saves the query with a unique ID and status "pending." The query is forwarded to the AI Case Management Assistant.
3. AI Processes Query: The AI Assistant analyzes the case query. If unclear, it requests clarification. If clear, it proceeds to retrieve relevant case information.
4. AI Retrieves Information: The AI Assistant gathers and organizes data related to the case, including hearing dates, evidence, and case progress.
5. Notify Client/Lawyer: The system notifies the client or lawyer when relevant information or updates are ready. The user reviews the information provided.
6. Case Resolution Status Update: If the client or lawyer confirms the details are correct, the query is marked as "resolved." If not, they can submit follow-up queries for additional information.

### **INPUTS FOR THE SMART LIBRARY MANAGEMENT SYSTEM:**

User Login Input:

User ID: Unique identifier for the client or lawyer.

password: Secure password for authentication.

Case Query Input:

Case Query Content: The query or question related to the case.

Example: "What is the next court date for my case?"

AI Assistant Input:

Query ID: Unique identifier for the query.

Case Query Content: Content of the query submitted by the user.

Example: "What documents are needed for the next hearing?"

Clarification Input (If Needed):

Clarification Message: Additional information provided by the user if the query is unclear.

Example: "I need clarification on the evidence requirements."

Case Update Input:

Query ID: Unique identifier of the processed query. caseStatusResult: The result after updating the case

status.

User Feedback Input (Optional):

feedback: Client's or lawyer's feedback or follow-up regarding the AI's response. Example: "Can you provide more details on case status?"

OUTPUT: Problem Statement: This system is a web application that uses AI to streamline case management by analyzing case-related queries and efficiently organizing relevant information for lawyers and clients.

Background:

The Court Case Management System is designed to serve as a virtual assistant for legal professionals and clients, providing quick access to case information, scheduling, and documentation. Accessible 24/7, it eases case tracking, enhances transparency, and reduces the workload in legal settings.

Relevance

The Court Case Management System is highly relevant for modern legal practices, as it addresses the increasing need for efficient case handling and accurate information. By providing 24/7 assistance to clients and legal professionals, the system helps courts and law firms meet the expectations of a tech-savvy client base. This system improves digital case management, enhances client-lawyer communication, and contributes to a more transparent and efficient legal process.

Objectives:

1. Enhance User Experience: Provide quick and accurate case information.
2. 24/7 Support: Ensure case information is available anytime.
3. Reduce Workload: Automate responses to common case inquiries.
4. Improve Accuracy: Deliver up-to-date info from court and case databases.
5. Continuous Learning: Improve through user feedback and machine learning.

## **Result :**

The problem statement has been successfully created, outlining the challenges in court case management and proposing a centralized solution to improve efficiency and user satisfaction in legal case handling.

## **Ex.NO. 2 : Write the software requirement specification document**

### **Aim :**

To do requirement analysis and develop Software Requirement Specification Sheet (SRS) for any Project.

### **Algorithm :**

#### **1. Introduction**

##### **1.1 Purpose**

The purpose of this document is to define the functional and non-functional requirements for the Court Case Management System (CCMS). This system aims to streamline court case management by providing a centralized platform for case tracking, documentation, scheduling, and communication. It will serve clients, lawyers, court staff, and administrators.

##### **1.2 Scope**

The CCMS is intended to simplify the process of handling court cases, from case initiation through resolution. The system will include features for case tracking, document management, automated notifications, and scheduling. It will be accessible to authorized users such as lawyers, clients, and court staff, enabling them to access relevant information 24/7.

##### **1.3 Definitions, Acronyms, and Abbreviations**

**CCMS:** Court Case Management System

**User:** Any individual interacting with the system, including clients, lawyers, and court staff

**Case ID:** A unique identifier for each court case

**AI Assistant:** An automated system that processes and responds to case queries

##### **1.4 Overview**

This document details the requirements for CCMS, specifying system functionalities, performance, and interface requirements.

#### **2. System Requirements**

##### **2.1 Functional Requirements**

###### **2.1.1 User Authentication and Authorization**

- The system shall require users to log in using a unique user ID and password.
- Different user roles (e.g., lawyer, client, court staff, administrator) shall have specific access rights.

###### **2.1.2 Case Registration and Management**

- The system shall allow court staff or lawyers to register new cases by providing details like case type, client details, and case status.
- Each case shall be assigned a unique Case ID.

###### **2.1.3 Case Tracking and Status Updates**

- The system shall enable users to view the status of ongoing cases.
- The system shall allow authorized users to update the case status, such as "Pending," "In Progress," or "Resolved."

###### **2.1.4 Document Management**

- The system shall allow users to upload and access documents related to each case, such as evidence files, affidavits, and court orders.
- Users shall be able to view, download, or update documents as per their role permissions.

#### **2.1.5 Scheduling and Notifications**

- The system shall allow court staff to schedule hearings and notify relevant users of upcoming court dates.
- The system shall automatically send notifications via email or SMS for upcoming hearings, new updates, and other case-related activities.

#### **2.1.6 Query and Response System**

- Users shall be able to submit queries related to their case, which will be processed by the AI Assistant.
- The AI Assistant shall provide case-related information or request clarification if the query is unclear.

#### **2.1.7 Case Summary and Report Generation**

- The system shall allow authorized users to generate summaries and reports for each case, including case history, documents, and hearing dates.

#### **2.1.8 User Feedback and Support**

- The system shall enable users to provide feedback on case management, service quality, or any issues faced.
- There shall be a support section where users can raise issues, which will be handled by customer support.

### **2.2 Non-Functional Requirements**

#### **2.2.1 Performance Requirements**

- The system shall respond to user queries and perform actions within 2 seconds.
- It shall support up to 1,000 concurrent users without significant performance degradation.

#### **2.2.2 Security Requirements**

- All user data shall be encrypted in transit and at rest.
- Access to sensitive information shall be restricted based on user roles.
- User sessions shall time out after 15 minutes of inactivity.

#### **2.2.3 Reliability Requirements**

- The system shall have an uptime of 99.9% to ensure that users can access case information whenever needed.
- In case of failure, the system shall recover within 5 minutes.

#### **2.2.4 Usability Requirements**

- The system shall be user-friendly and accessible, with a straightforward navigation interface.
- It shall provide a help section with FAQs and user guides for ease of use.

#### **2.2.5 Scalability**

- The system architecture shall be scalable to support increasing numbers of cases and users as the court caseload grows.

### **3. User Interface Requirements**

#### **3.1 Dashboard**

- The user dashboard shall display recent cases, case statuses, upcoming court dates, and recent notifications.
- The dashboard shall be customized based on user roles (e.g., lawyer, client, court staff).

#### **3.2 Case Details Page**

- The Case Details page shall include case status, related documents, scheduled hearings, and case history.
- Authorized users shall have the option to update case status or upload new documents.

#### **3.3 Query Submission Interface**

- The query submission page shall allow users to type questions or request updates regarding their case.
- A chat-like interface shall allow interaction with the AI Assistant for real-time responses.

#### **3.4 Document Management Interface**

- Users shall be able to upload, view, and download documents related to a specific case.
- Document access permissions shall be determined by user role.

#### **3.5 Notification Panel**

- The notification panel shall display alerts for upcoming hearings, case status changes, and document uploads.
- Notifications shall be categorized by priority (e.g., high-priority alerts for urgent updates).

### **4. Assumptions and Dependencies**

- The system assumes stable internet connectivity for real-time updates and access.
- The AI Assistant requires access to a well-maintained database with case-related data.

Email and SMS notification services depend on third-party providers and may require integration with external APIs

### **Result :**

The Software Requirement Specification (SRS) for the **Court case management system(ccms)** has been successfully developed, addressing the key features, constraints, and functionalities necessary to streamline the court cases mangement.



### 3. ENTITY RELATIONSHIP MODEL

#### **Aim :**

To Draw the Entity Relationship Diagram for Court case management system..

#### **Algorithm :**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1: N Relationship Types.

Step 5: Mapping of Binary M: N Relationship Types.

Step 6: Mapping of Multivalued attributes.

#### **Input :**

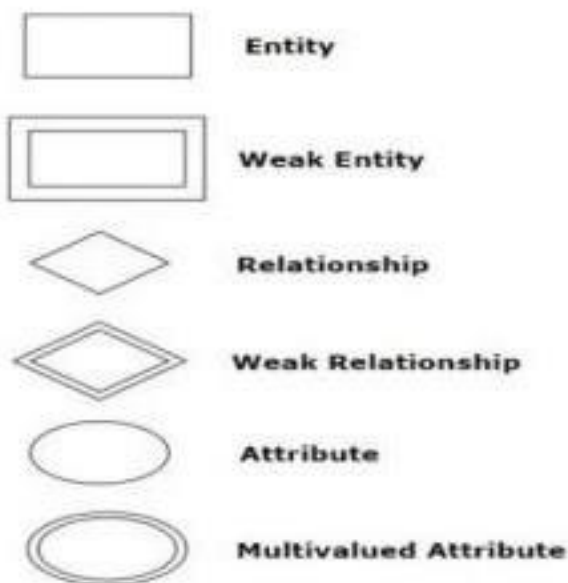
# Entities

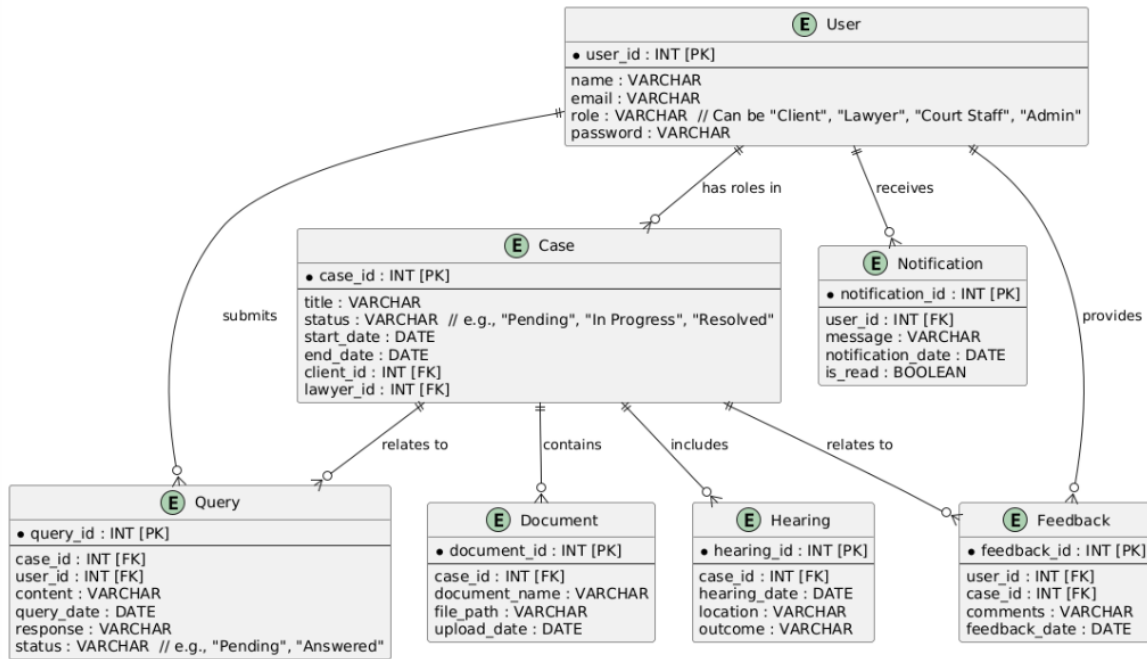
# User

# case

#### **ER DIAGRAM:**

SYMBOLS:





**Result:** The entity relationship diagram was made successfully by following the steps described above.

#### 4. DATA FLOW DIAGRAM

**Aim :** To Draw the Data Flow Diagram for COURT CASE MANAGEMENT SYSTEM and List the Modules in the Application.

**Algorithm :**

1. Open Visual Paradigm or Lucidchart.
2. Select a DFD Template and name it "Court Case Management System."
3. Add External Entities: Include 'client and 'Lawyer.'
4. Add Processes: Create processes like as Case Management, Document Management, Scheduling and Notification, Query Processing, and Feedback Management
5. Add Data Stores: Include stores like Case Records, User Data, Documents, Hearing Schedules, and Query Responses.
6. Connect Entities, Processes, and Data Stores: Draw data flows like "case management ," "scheduling," and "client query ."
7. Label Data Flows: Clearly label the data exchanged between entities and processes.
8. Customize: Adjust colors, fonts, and titles for clarity.
9. Export/Share: Export or share the completed DFD.




**Input :**

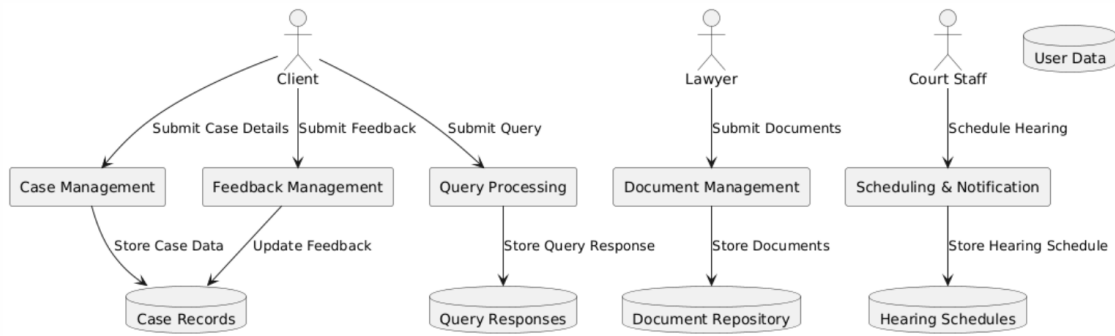
**Processes:**

- Case Management, Document Management, Scheduling & Notification, Query Processing, Feedback Management

**• Data Stores:**

User data, case records, Document repository, Hearing schedules, query responses.

Notation	Yourdon & De Marco	Gene & Sarson	SSADM	Unified
External Entity				
Process				
Data Store				
Data Flow				



**Result:** The Data Flow diagram was made successfully by following the steps described above.

## 5. USE CASE DIAGRAM

**Aim :** To Draw the Use Case Diagram for Court case management system.

**Algorithm :**

- ☐ Identify Actors:
  - Client
  - Lawyer
  - Court staff
  - Admin
- ☐ Identify Use Cases:
  - Login
  - Submit case details
  - Upload documents
  - Schedule hearing
  - View case status
  - Manage Users (Admin)
- ☐ Connect Actors to Use Cases:
  - Client interacts with Submit Case Details, Submit Query, Give Feedback, and View Case Status.
  - Lawyer interacts with Upload Documents, View Case Status, and Schedule Hearing.
  - Court Staff interacts with Schedule Hearing, View Case Status, and Manage User Accounts
    - Label the boundary as "Court Case Management System."
- ☐ Define Relationships:
  - Use include or extend where applicable. Review and Refine:
- ☐ Ensure all relevant actors and use cases are connected.
- ☐ Validate Diagram Accuracy:
  - Double-check for completeness and correct relationships.

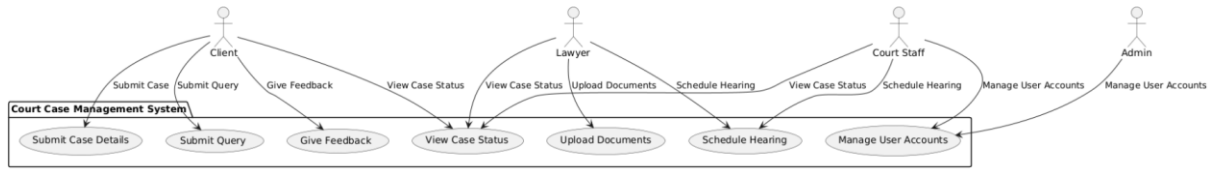
**Input :**

Actors

Use Cases

Relations

**Sample Output :**



## **Result :**

The use case diagram has been created successfully by following the steps given.

## 6. ACTIVITY DIAGRAM

### **Aim:**

To Draw the activity Diagram for COURT CASE MANAGEMENT SYSTEM

### **Algorithm :**

- ☐ Identify Initial and Final States:
  - Initial State: System Idle / Logged Out
  - Final State: User Logs Out / System Shutdown
- ☐ Identify Intermediate Activities:
  - Upload detail
  - Upload documents
  - Scheduling
  - feedback
- ☐ Identify Conditions or Constraints:
  - Successful Login
  - Case completed?
  - Hearing schedule
  - Feedback Draw the
- ☐ Diagram:
  - Use ovals for states (e.g., Login, Search, Issue).
  - Use arrows for transitions between states.
  - Use initial state symbol (solid circle) for the start and final state symbol (circle with a border) for the end.

### **Inputs :**

Activities

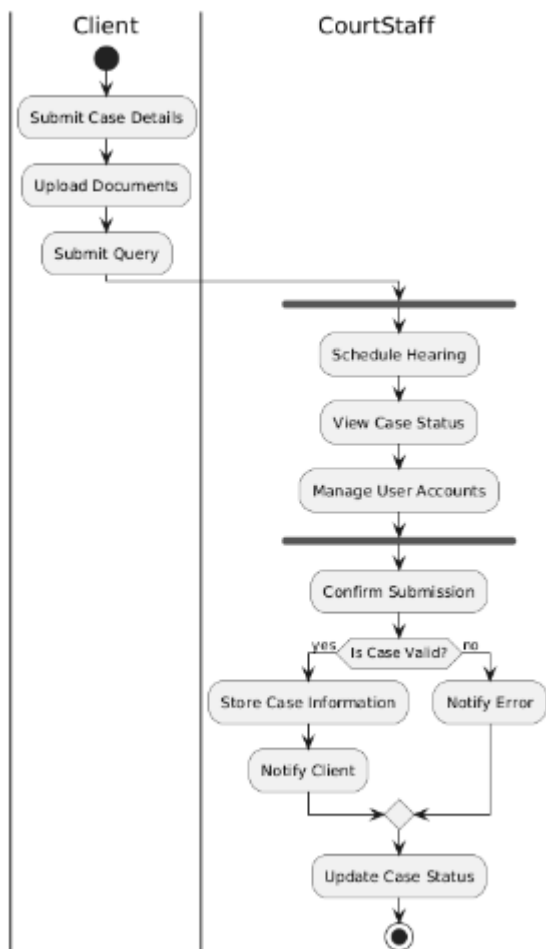
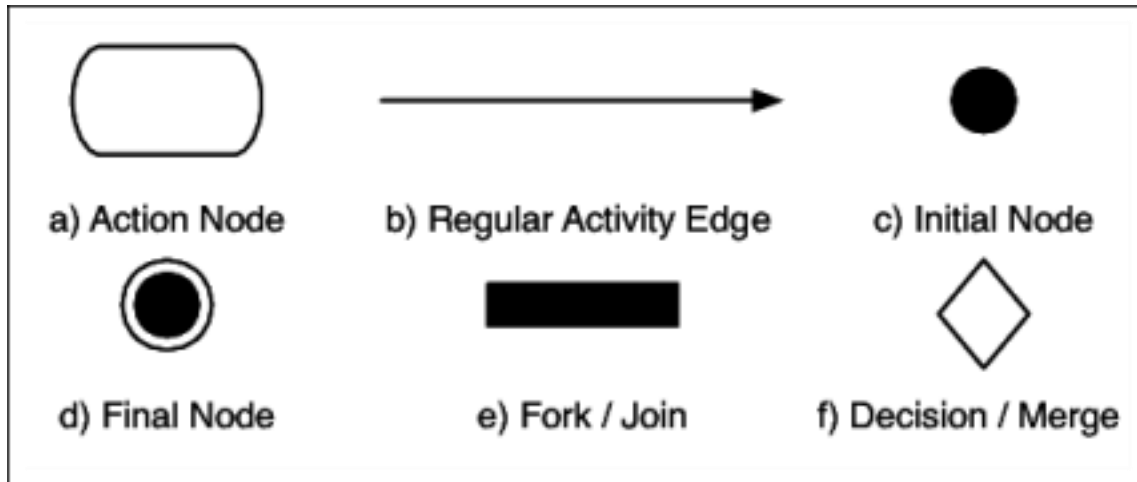
Decision Points

Guards

Parallel Activities

Conditions

### **Sample Output :**



**Result** : The Activity diagram has been created successfully by following the steps given.



## 7. STATE CHART DIAGRAM

### **Aim :**

To Draw the State Chart Diagram for COURT CASE MANAGEMENT SYSTEM.

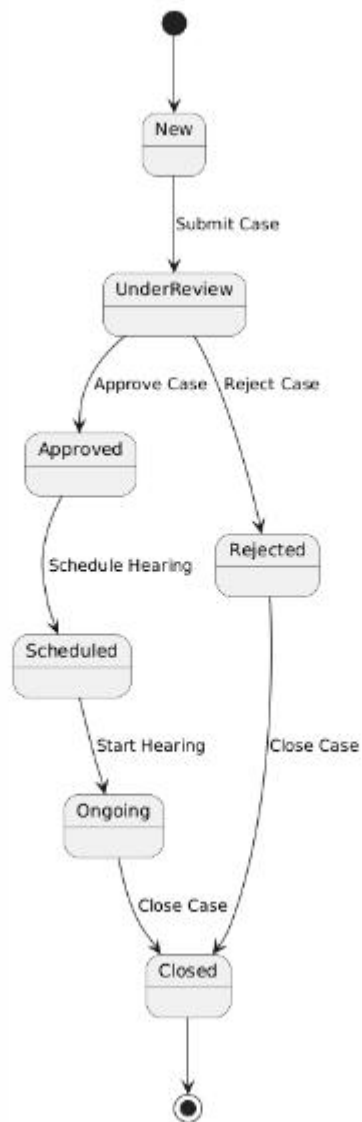
### **Algorithm :**

1. Identify Important Objects
  - Case hearing documents
2. Identify the States
  - New, Under Review, closed
3. Identify the Events
  - Submit case, Approve case, Close case.

### **Inputs :**

Objects  
States  
Events

### **Output :**



**Result** : The State Chart diagram has been created successfully by following the steps given.

## 8. SEQUENCE DIAGRAM

### **Aim :**

To Draw the Sequence Diagram for court case management system.

### **Algorithm :**

1. **Identify the Scenario:**  
define the process or use case
2. **List the Participants:**
  - Client court staff lawyer.
3. **Define Lifelines:**
  - Vertical lines for each participant.
4. **Arrange Lifelines:**
  - Place in order of interaction (User, lawyer, System).
5. **Add Activation Bars:**
  - Show active periods with vertical bars.
- Represent simultaneous actions if needed.
6. **Review and Refine:**
  - Check for accuracy.
7. **Add Annotations:**
8. **Consider parallel execution**
  - Clarify complex interactions.
9. **Document Assumptions:**
  - Note assumptions made.
10. **Use a Tool:**
  - Create the diagram using software (e.g., Lucidchart).

### **Inputs :**

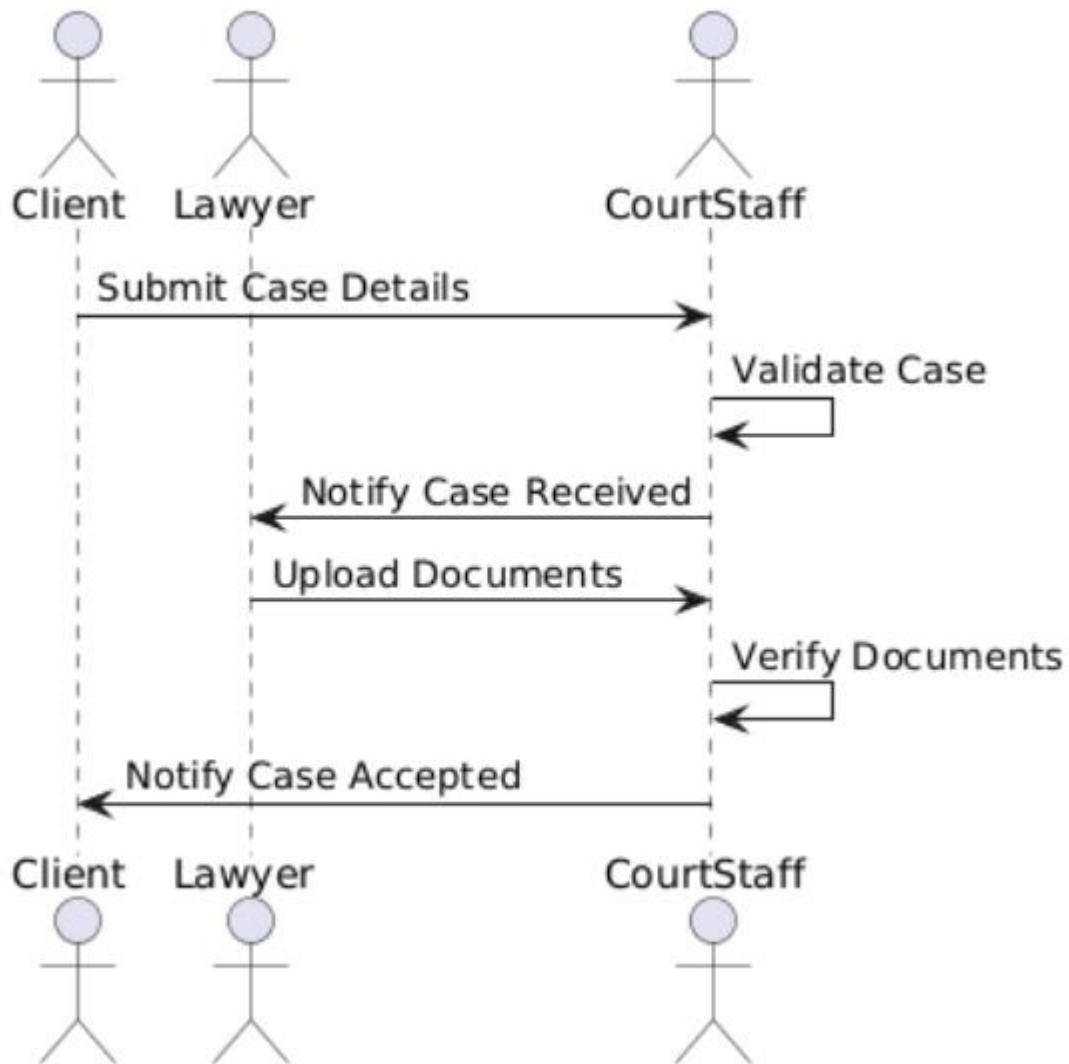
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

### **Sample Output :**



**Result :**

The Sequence diagram has been created successfully by following the steps given.

## 9. COLLABORATION DIAGRAM

### Aim :

To Draw the Collaboration Diagram for **COURT CASE MANAGEMENT SYSTEM**.

### Algorithm :

- ☐ Identify Objects/Participants:
  - client lawyer. Court staff, admin
- ☐ Interactions:
  - Client** communicates with **CourtStaff** to submit case details.
  - Lawyer** communicates with **CourtStaff** to upload documents.
  - CourtStaff** manages the case lifecycle and interacts with **Admin** to manage system updates.
- ☐ Add Messages:
  - Messages exchanged: "Request Issue," "uploading docs," "schedule conformation."
- ☐ Consider Relationships:
  - Client** submits case details to **Courtstaff**
- ☐

### Inputs :

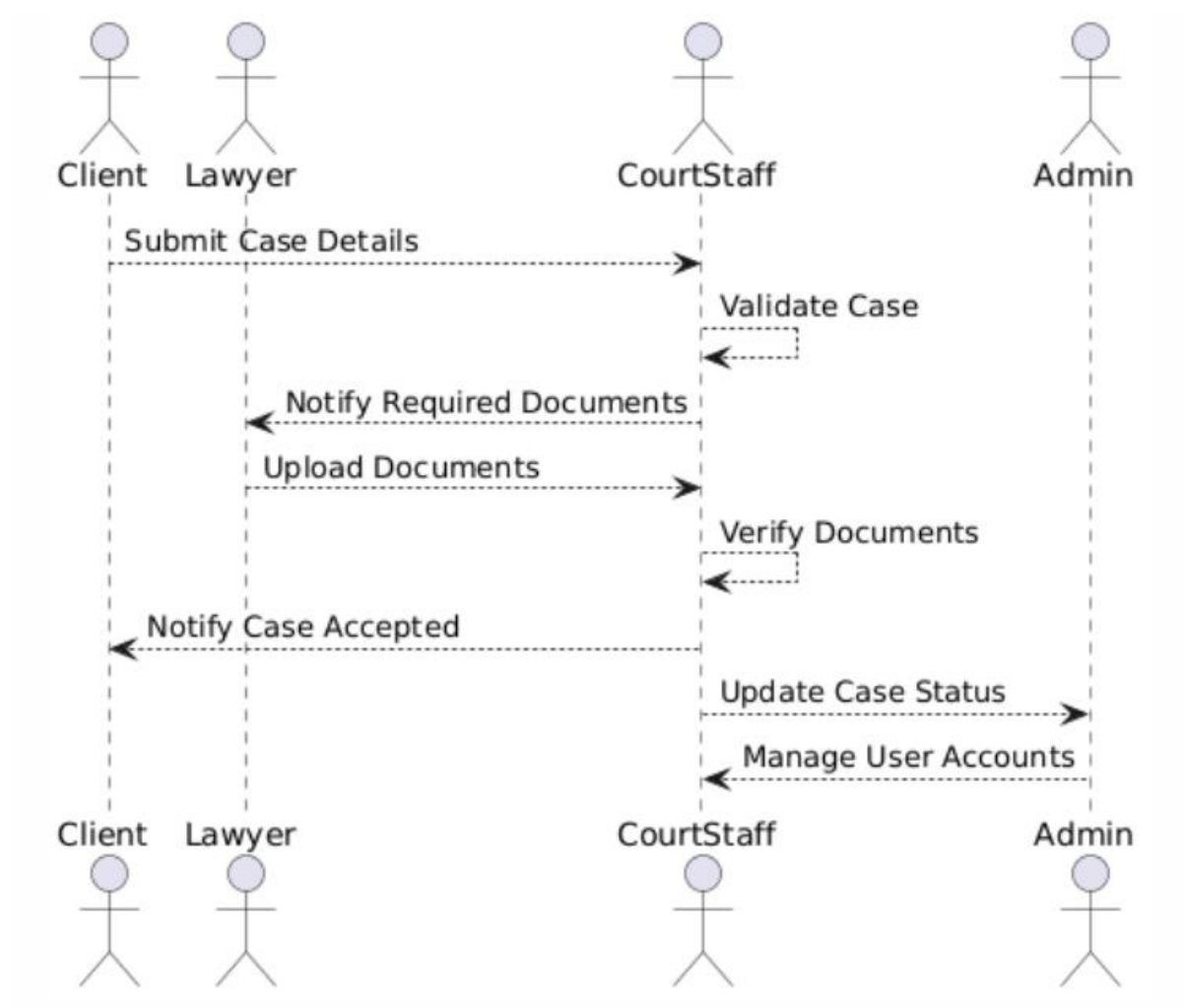
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

### Sample Output :



### **Result :**

The Collaboration diagram has been created successfully by following the steps given.

## 10. CLASS DIAGRAM

### **Aim:**

To Draw the Class Diagram for SMART LIBRARY MANAGEMENT SYSTEM

### **Algorithm :**

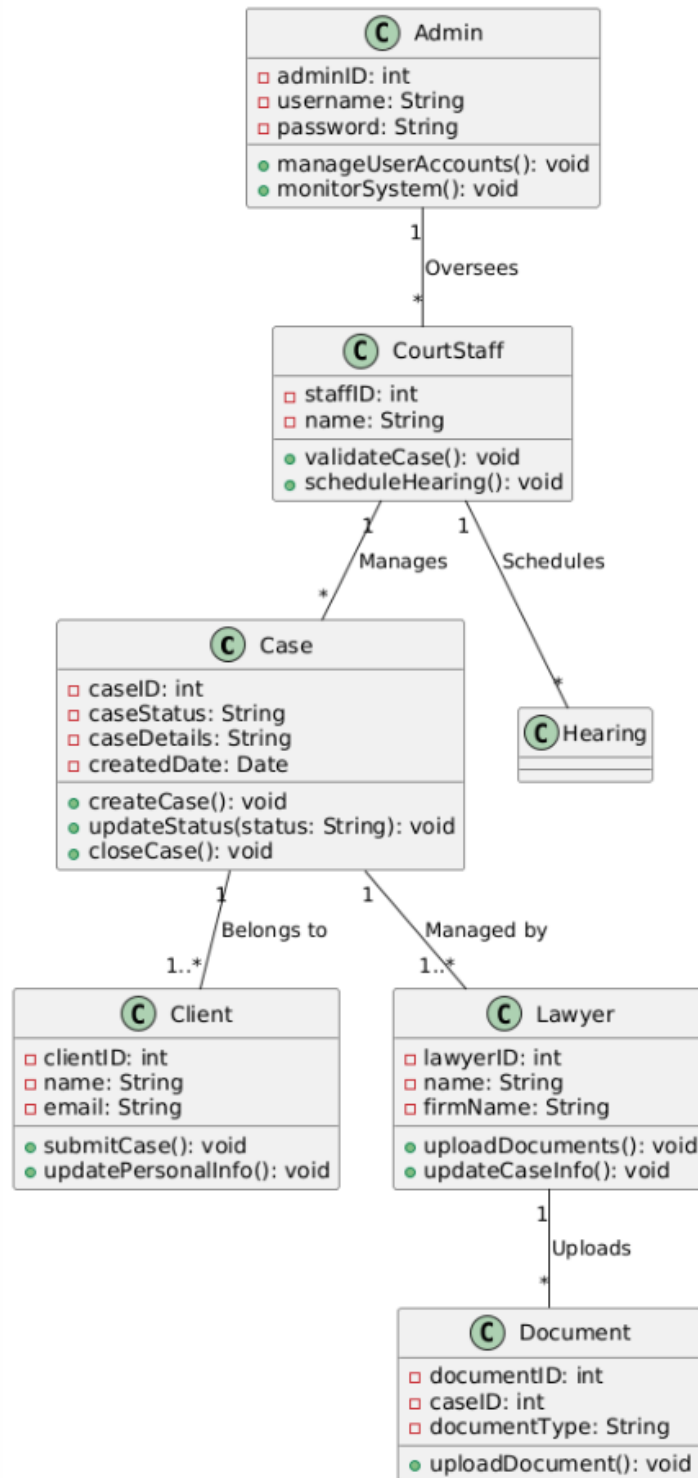
- ☐ Identify Classes:
  - Case, Client Lawyer CourtStaff Document Hearing
- ☐ List Attributes and Methods:
  - Attributes: caseID, caseStatus, caseDetails, createdDate
  - Methods: createCase(), updateStatus(), closeCase()
- ☐ Identify Relationships:
  - Case is associated with Client (a case belongs to a client).
  - Lawyer is related to Document (a lawyer uploads documents for a case).
  - CourtStaff manages Case and schedules Hearing TraCreate
- ☐ Class Boxes:
  - Draw separate boxes for each class and include their attributes and methods.
- ☐ Draw Relationships:
  - Connect classes with lines to show relationships (e.g., association).
- ☐ Label Relationships:
  - Label relationship types (e.g., one-to-many, many-to-many).
- ☐ Review and Refine:
  - Ensure accuracy and completeness.
- ☐ Use Tools for Digital Drawing:
  - Create the diagram using digital tools like Lucidchart.

### **Inputs :**

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

### **Sample Output :**

231401099



## **Result:**

The Class diagram has been created successfully by following the steps given.



## CODE FOR MINI PROJECT

### **CODE:**

### **IMPLEMENTATION CODE USING PYTHON**

```
class Client:
    def __init__(self, name, client_id):
        self.name = name
        self.client_id = client_id

    def __str__(self):
        return f'{self.name} (ID: {self.client_id})'

class Case:
    def __init__(self, case_id, case_details, client):
        self.case_id = case_id
        self.case_details = case_details
        self.client = client

    def __str__(self):
        return f'Case ID: {self.case_id}, Details: {self.case_details}, Client: {self.client}'

class CourtCaseManager:
    def __init__(self):
        self.cases = []

    def add_case(self, court_case):
        self.cases.append(court_case)
        print('Case added successfully.')

    def display_cases(self):
        if not self.cases:
            print('No cases available.')
        else:
            for court_case in self.cases:
                print(court_case)

def main():
    manager = CourtCaseManager()
```

```

while True:
    print("\nCOURT Case Management System")
    print("1. Add Case")
    print("2. Display Cases")
    print("3. Exit")
    choice = input("Choose an option: ")

    if choice == '1':
        client_name = input("Enter Client Name: ")
        client_id = input("Enter Client ID: ")
        case_id = input("Enter Case ID: ")
        case_details = input("Enter Case Details: ")

        client = Client(client_name, client_id)
        court_case = Case(case_id, case_details, client)
        manager.add_case(court_case)

    elif choice == '2':
        manager.display_cases()

    elif choice == '3':
        print("Exiting...")
        break

    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()

```

## INPUT:

```

Court Case
Management System
1. Add Case
2. Display Cases
3. Exit
Choose an option:

```

**OUTPUT:**

**Choose an option: 1**

**Enter Client Name: Rahul**

**Enter Client ID: 552**

**Enter Case ID: 445f**

**Enter Case Details: murder**

**Case added successfully...**