

**North South University**

Department of Electrical & Computer Engineering

**Project Report**

Course Code: CSE327

Course Initial: **SLB**

Project Name:

**Court Case Management System** (CCMS)

Section: 10

Submitted To: **Sornali Basak**

Submitted By:

|  |  |
| --- | --- |
| **Name** | **ID** |
| 1. Ishtiaq Ahmed Moyen | 2131580642 |
| 1. Abir Bin Faruque | 2131101642 |
| 1. Rayhan Amin | 2212830042 |
| 1. Sazid Amin | 2211513042 |

Overview Of the Software

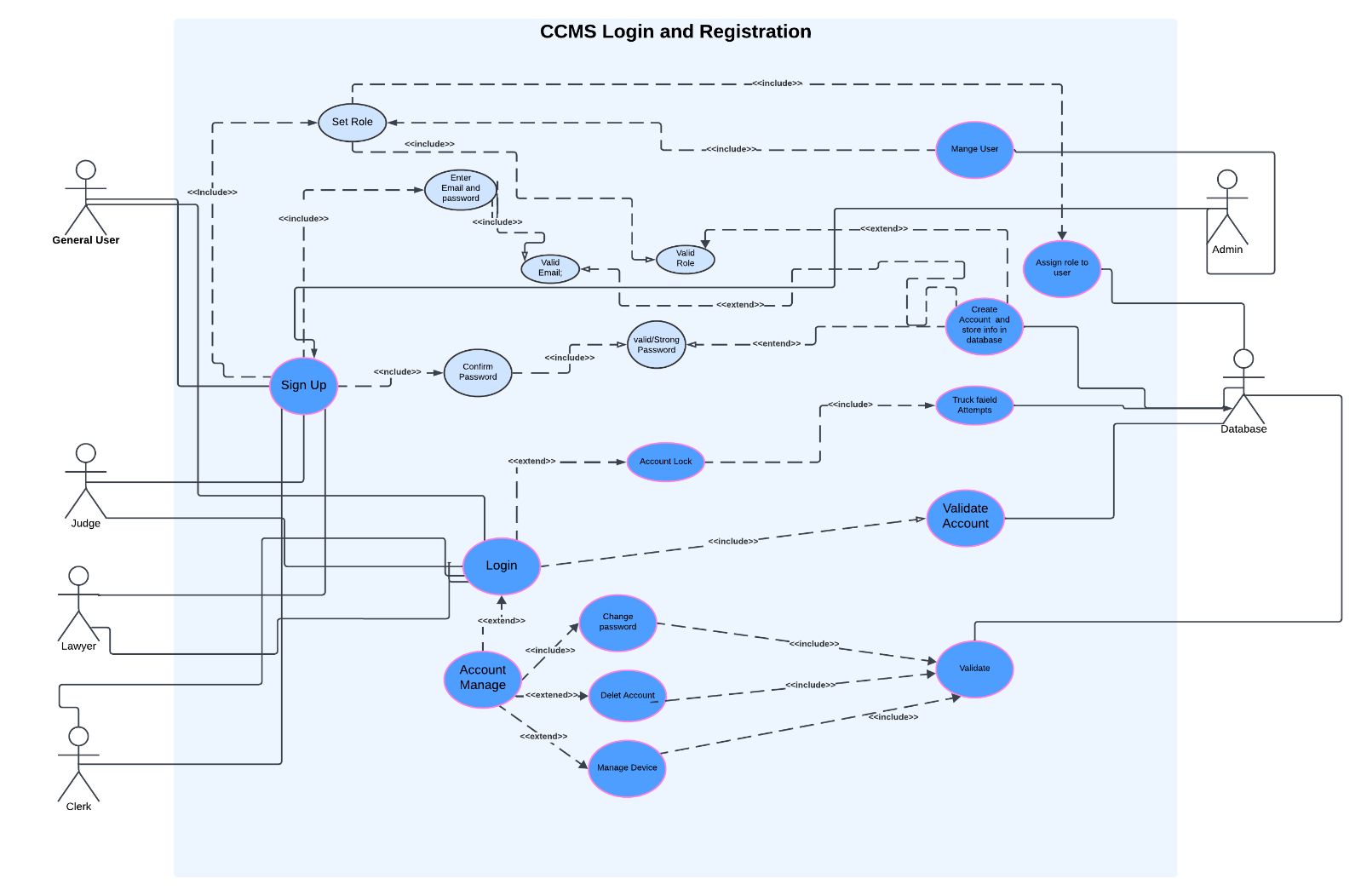
A Court Case Management System (CCMS) is a comprehensive digital platform designed to streamline the administration of legal cases, making the judicial process more efficient and transparent. It serves as a centralized system to manage court records, track case statuses, and automate administrative tasks, ensuring that all stakeholders—judges, lawyers, clerks, and the public—have seamless access to information. The system is also designed to enhance the overall workflow of court operations, from case registration to document management, making the judicial system more efficient and accessible.The system’s primary functionality revolves around Case Registration and Tracking, allowing for digital case entry and real-time tracking throughout various stages of litigation. Cases can be easily monitored from filing to hearings and rulings, with automatic updates on case status, ensuring that all involved parties stay informed. Additionally, Document Management plays a critical role in CCMS, offering secure storage and easy retrieval of legal documents. It supports e-filing, allowing lawyers and clerks to submit and access documents electronically, significantly improving the speed and accuracy of document handling in the judicial process.For ease of use, the CCMS includes a Case Search and Filtering feature that enables users to search and filter cases by various criteria such as case number, parties involved, case type, or date. This feature ensures that users can quickly retrieve relevant cases from the system. Furthermore, the Role-Based Access Control system ensures that access to sensitive data is restricted to authorized individuals based on their roles. Judges, clerks, lawyers, and the public all have different levels of access, ensuring data privacy and security.One of the core operational features of CCMS is Courtroom Assignment and Resource Management. This functionality automates the allocation of courtrooms and other resources, preventing scheduling conflicts and ensuring the smooth running of court sessions. Coupled with Billing and Fine Management, the system helps manage fees, fines, and payments, allowing for the generation of invoices and tracking of outstanding dues. This feature also facilitates the creation of detailed billing reports for better financial management of the court system.

Content Page

|  |  |
| --- | --- |
| Content Name | Pg. Num |
| 1. Software Overview | 1-4 |
| 1. Register Page | 6-12 |
| 1. Dashboard Page | 13-19 |
| 4.Case Registration |  |
| 5.Document manage |  |
| 6.Court manage |  |
| 7.Fine |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Register Page:**

**Use Case Diagram**



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

**User Registration**:

* 1. The system must allow users to create an account by entering an email, password, and confirming their password.
  2. Users must select and validate their role (Judge, Lawyer, Clerk).
  3. The system should validate the entered email and ensure the password is strong.

**User Login**:

* 1. Users must be able to log in using their registered credentials.
  2. Failed login attempts should be tracked, and accounts must be locked after a predefined number of failed attempts.

**Account Management**:

* 1. Users should be able to manage their accounts, including changing their password, deleting their account, and managing associated devices.

**Role Management**:

* 1. Administrators should be able to assign roles to users and manage user data.

**Validation**:

* 1. Accounts must be validated after registration before granting full access to the system.

**Audit and Monitoring**:

* 1. The system should keep track of failed login attempts for security purposes.

**Database Interaction**:

* 1. All account data (email, password, role) must be securely stored in a database.

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

**Performance**:

* 1. The system should handle multiple simultaneous login and registration requests without performance degradation.

**Security**

* 1. All passwords must be encrypted before storage.
  2. The system must prevent unauthorized access by locking accounts after repeated failed attempts.
  3. Strong validation rules should be implemented for password creation.

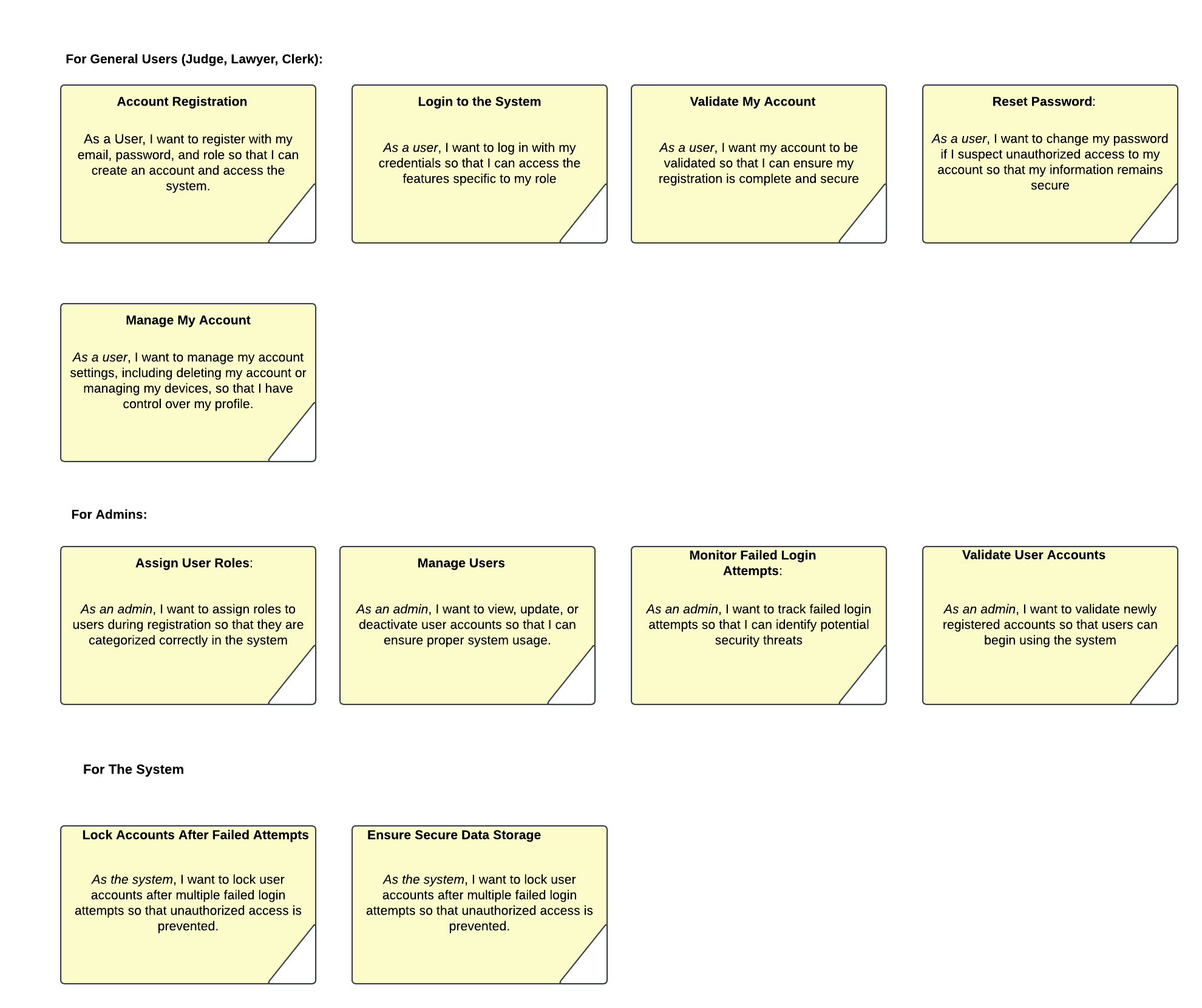
**Usability**:

* 1. The registration and login process should be intuitive and user-friendly.

**Audit Trail**:

* 1. All critical actions (role assignment, account validation) should be logged for auditing purposes.

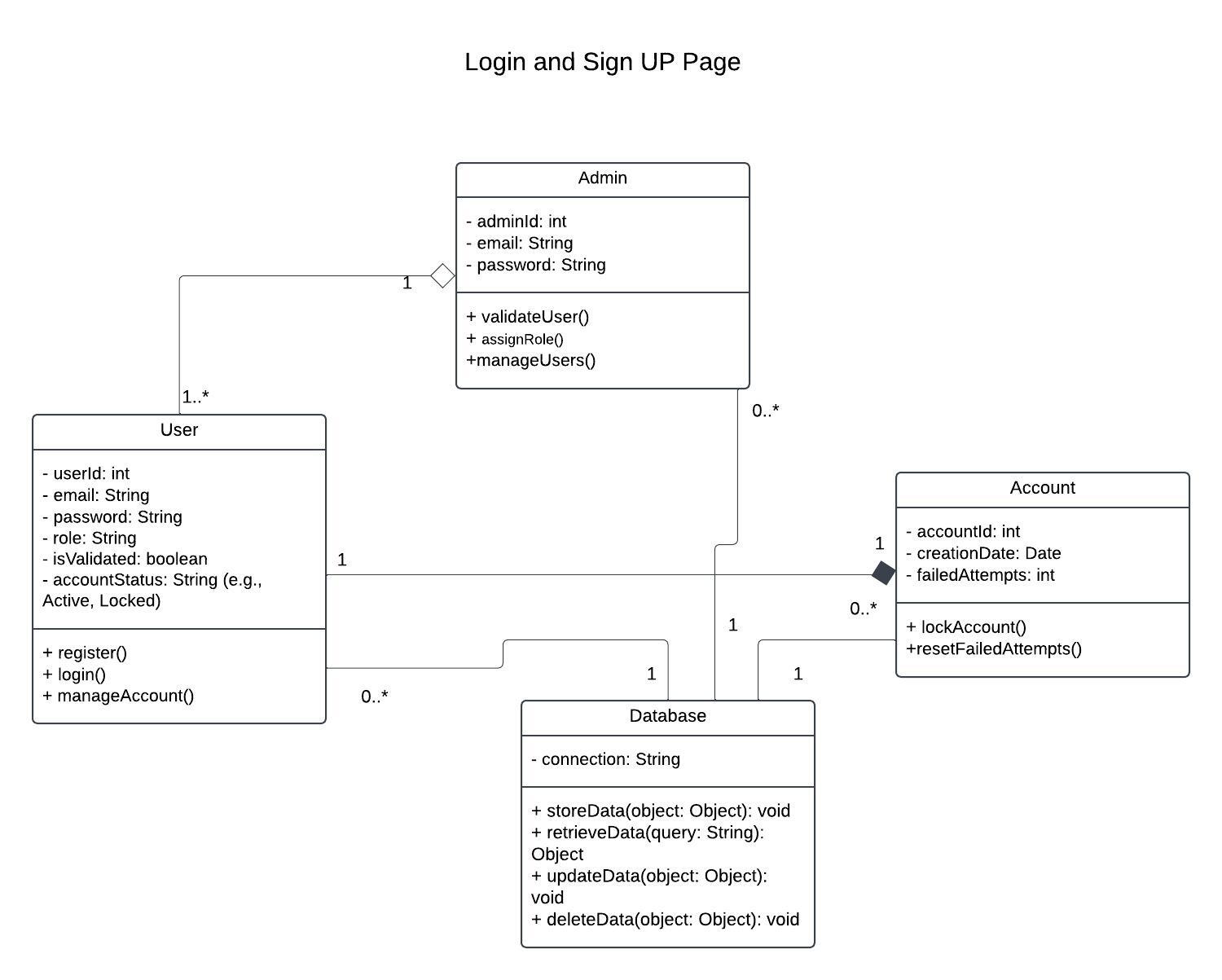
**User Story**



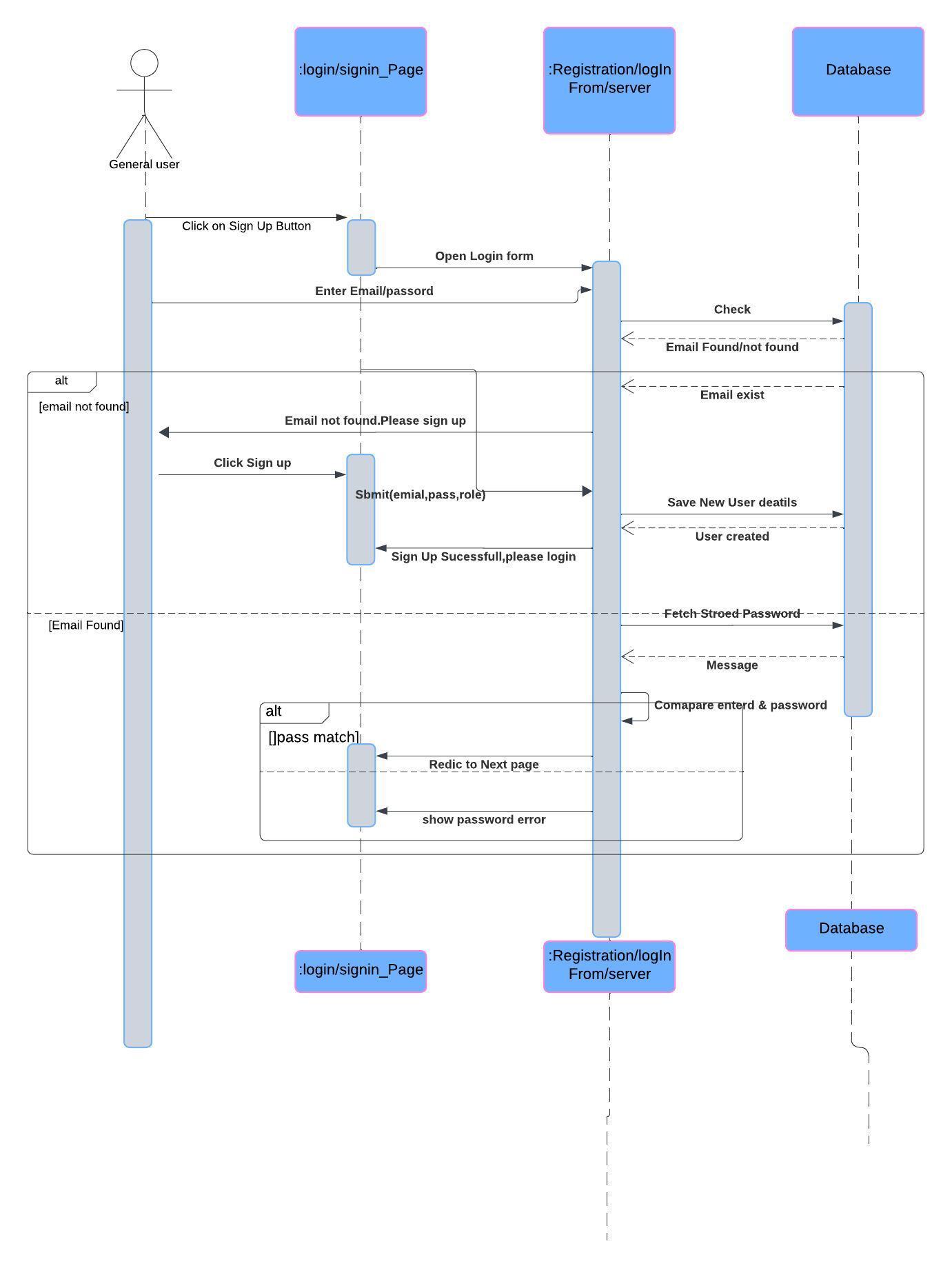
**Use Case Specification**

|  |
| --- |
| **Login and Registration**: Login and Registration |
| **Actors**: General User (Judge, Lawyer, Clerk) Admin System (as an automated actor), Database.  **Description:**  **Registration:**   1. **The user selects the "Sign Up" option.** 2. **The system prompts the user to enter an email, password, and confirm the password.** 3. **The system validates the email and password (password strength and format).** 4. **The user selects a role (Judge, Lawyer, Clerk)** 5. **The system validates the role and stores the account in the database.** 6. **Admin reviews and validates the account for activation.**   **Login**:   1. The user selects the "Login" option. 2. The system prompts the user to enter their email and password. 3. The system checks the credentials against the database. 4. If valid, the user is granted access to their role-specific dashboard**.** 5. If invalid, the system increments the failed attempt counter   **Account Management**:   1. The user selects "Manage Account." 2. Options are provided to:   · Change Password  · Delete Account  · Manage Devices   1. The system executes the requested action and updates the database.   **Exception:**   1. If the system is offline or database connection fails, an error message is displayed, and the operation is terminated. 2. Users can request a password reset link if they cannot remember their password.   **Preconditions**:   1. The system must be online and accessible. 2. A valid database connection must be established. 3. Users must provide unique and valid credentials during registration.   **Postconditions**:   1. Users are successfully registered and validated. 2. Users can log in and access role-specific features. 3. Failed login attempts are tracked, and accounts are locked after multiple failed attempts. |

**Class Diagram**

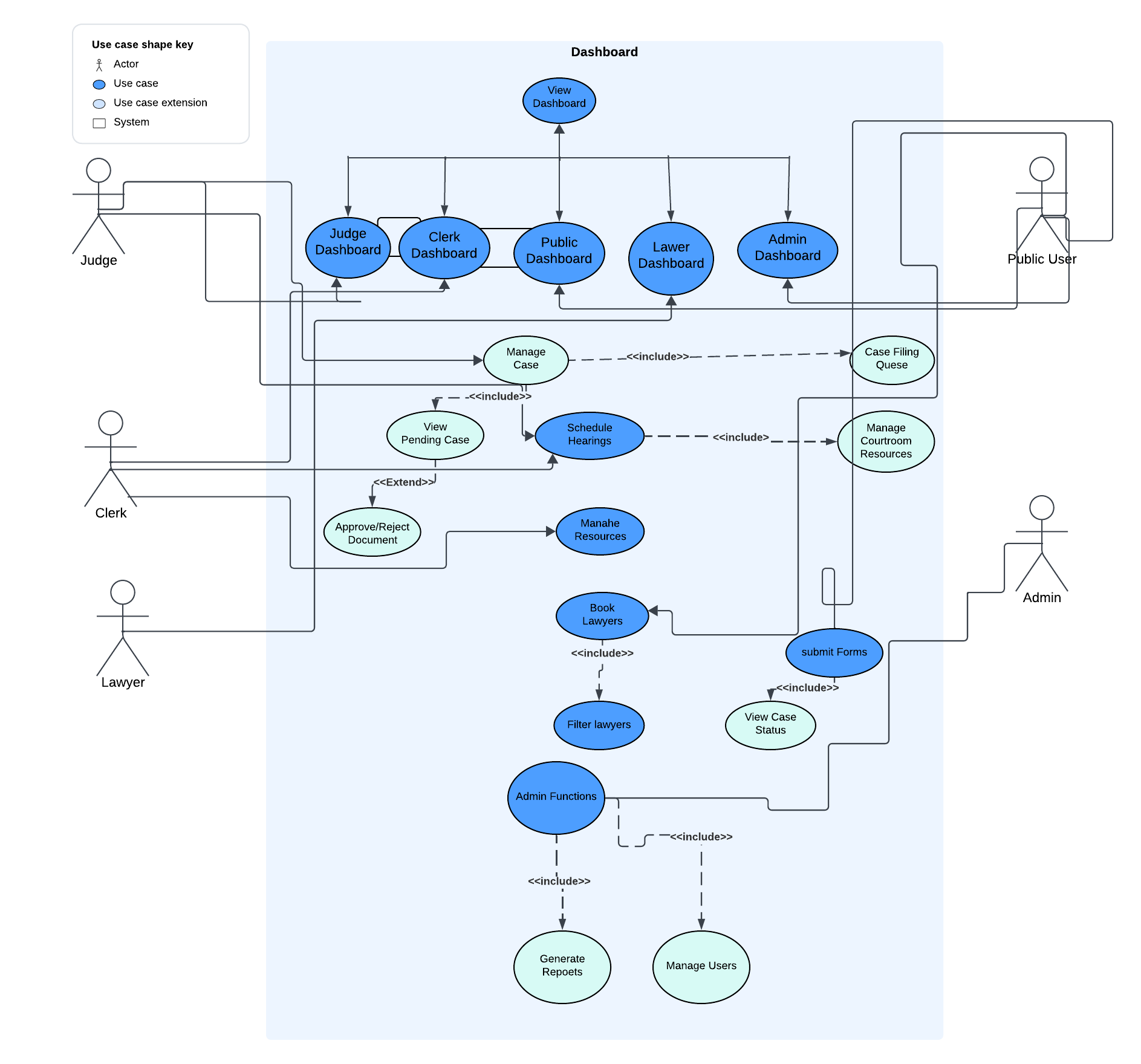
****

**Sequence Diagram**



**Dashboard Page**

**Use Case Diagram:**



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

1. Display tailored dashboard content based on user roles (Judges, Clerks, Lawyers, Public Users, and Admins).
2. Judges: Show pending cases, hearings schedule, document approval tasks, and performance stats.
3. Clerks: Display case filing queue, resource allocation options, and document management tools.
4. Lawyers: Present active cases, hearing schedules, and a communication portal for client interactions.
5. Public Users: Provide options to book lawyers, view case statuses, and submit forms.
6. Admins: Manage users, oversee court schedules, assign courtrooms, and view reports.
7. Judges: Access and update the status of pending cases and manage hearings.
8. Clerks: Organize the filing queue, schedule hearings, and allocate resources to courtrooms.
9. Lawyers: Track case progress and manage their hearing schedules.
10. Include case status and form submission tools for Public Users.
11. Allow Admins to generate reports and manage user accounts.
12. Implement search tools to retrieve specific cases, documents, or hearing schedules.
13. Interactive widgets to display dynamic information (number of pending cases, hearing dates).
14. Notifications for case updates, schedule changes, and task deadlines.
15. Allow Judges to approve/reject case-related documents directly from their dashboards.
16. Enable Clerks to manage courtroom assignments and other resources.

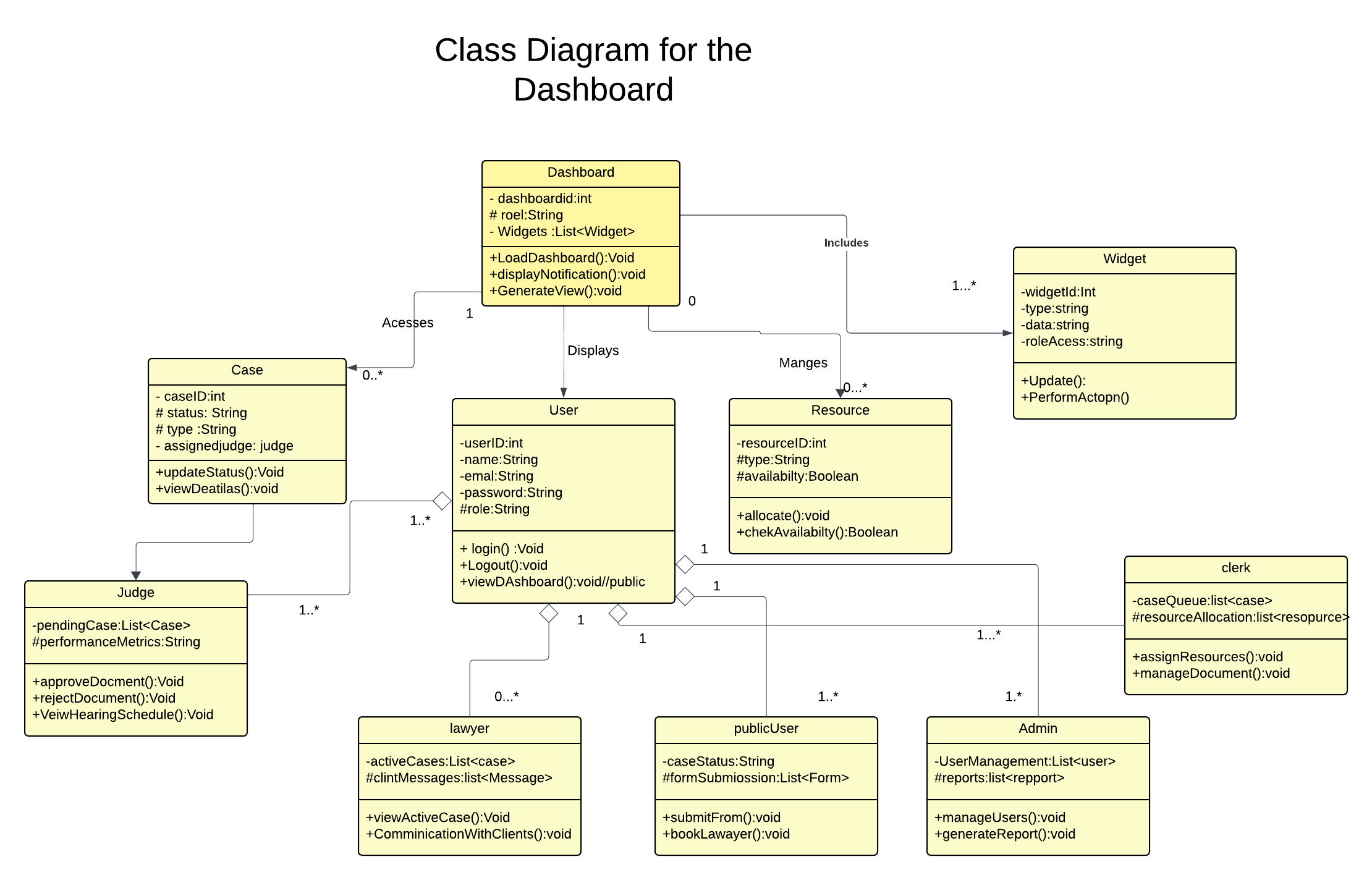
### ****Non-Functional Requirements****

1. Dashboards should load within 2 seconds for all users under normal system load.
2. Efficient handling of large datasets (thousands of pending cases).
3. Support increasing numbers of users (Judges, Lawyers, and Public Users) without performance degradation.
4. Ability to handle future feature expansion without significant architectural changes.
5. Ensure 99.9% uptime for the dashboard, with minimal downtime for maintenance.
6. Implement role-based access control (RBAC) to restrict dashboard features by user role.
7. Encrypt sensitive data displayed on dashboards, such as case details and personal information.
8. Provide an intuitive and user-friendly interface for all roles, with minimal training required.
9. Responsive design to support access from desktops, tablets, and smartphones.
10. Ensure real-time synchronization of data across all dashboards for accurate case and schedule updates.
11. Maintain a log of all dashboard activities, including actions taken by Judges, Clerks, and Admins (document approvals, case status updates).
12. Ensure compatibility with all modern browsers (Chrome, Firefox, Edge).
13. Support integration with external systems for case data synchronization.
14. Provide support for multiple languages and regional date/time formats, if necessary.

**User Story:**

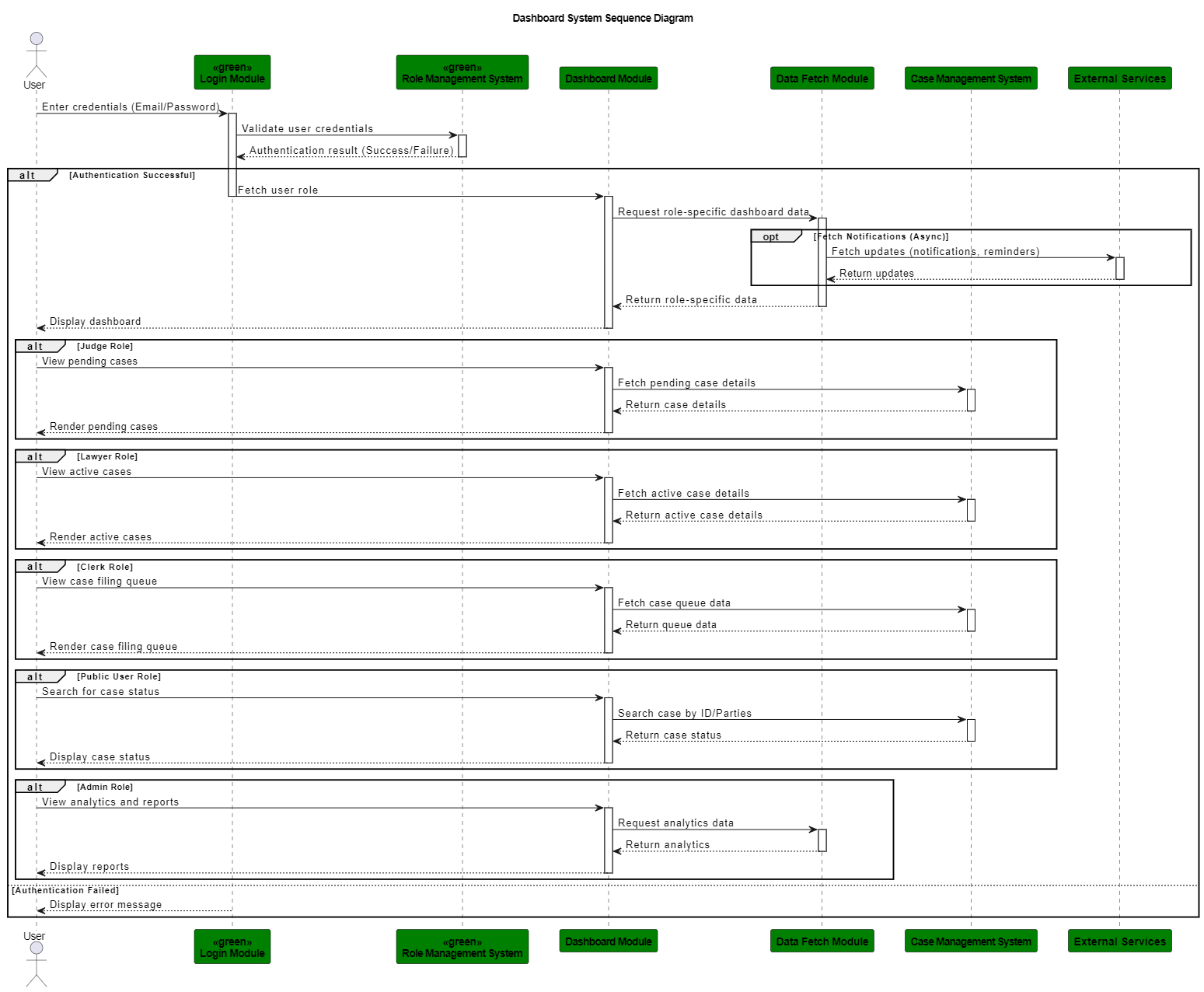
****

**Class Diagram:**

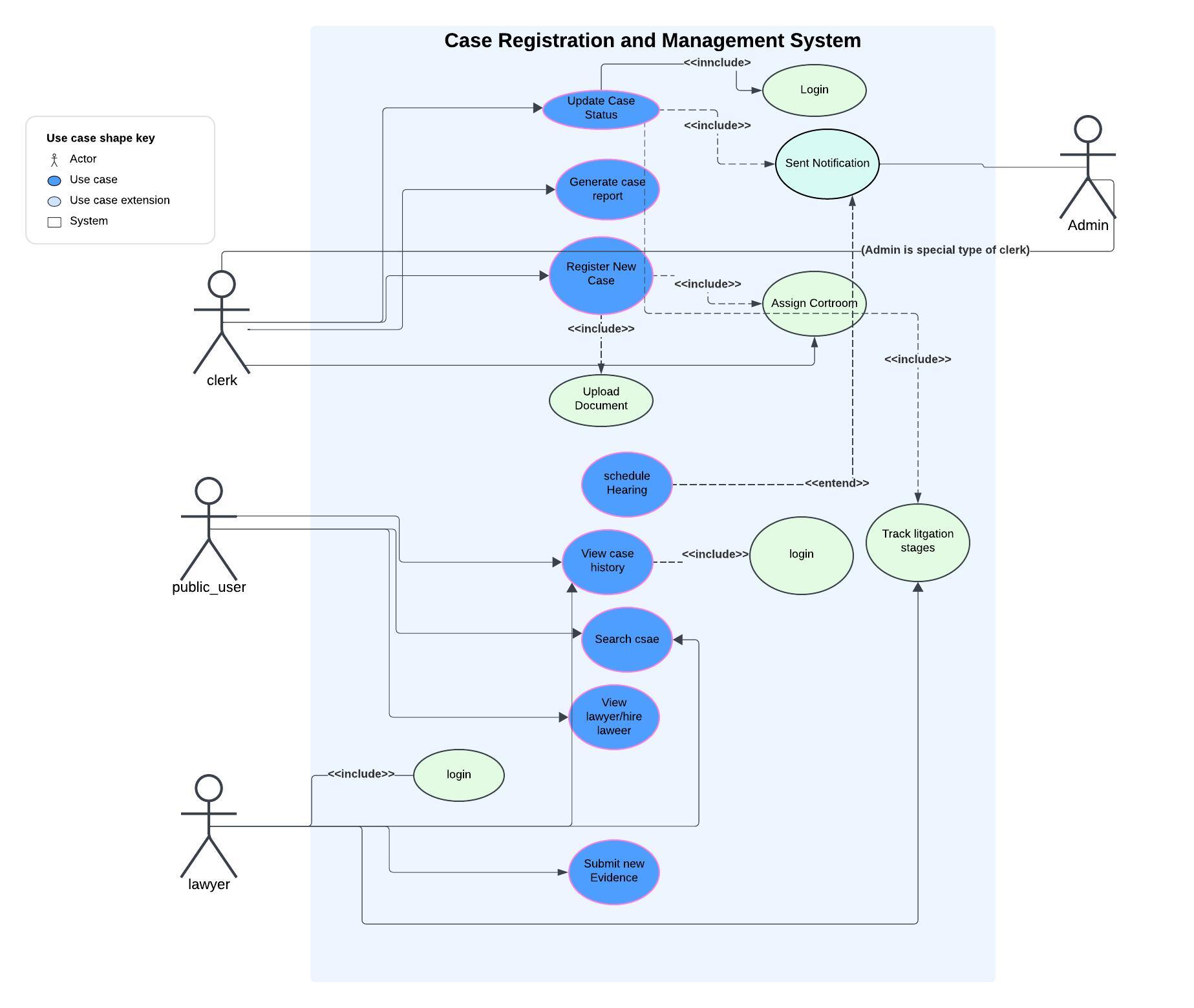
****

**User Specification:**

|  |
| --- |
| **Use Case Name**: Dashboard |
| **Actors**: Judge, Clerk, Lawyer, Public User, Admin, External Systems (for data synchronization)  **Scenario:**   1. The user logs into the CCMS. 2. The system identifies the user's role and loads the corresponding dashboard view. 3. The user interacts with the dashboard based on their role:   Judge:  a. Views pending cases and performance stats.  b. Approves or rejects documents.  c. Manages hearing schedules.  Clerk:  a. Views the case filing queue.  b. Allocates resources to courtrooms.  c. Manages case-related documents.  Lawyer:  a. Views active cases and upcoming hearings.  b. Communicates securely with clients.  Public User:  a. Books a lawyer using the system.  b. Submits forms or documents.  c. Views case statuses.  Admin:  a. Manages users and permissions.  b. Generates reports.  c. Oversees resource allocation.   1. The user logs out or navigates to another module.   **Exception:**   1. If the dashboard fails to load within the specified time, the system displays a "Service Unavailable" error message. 2. No database Connection. 3. Required page not found.   **Precondition:**   1. The user must be authenticated and authorized to access the system. 2. The user role (Judge, Clerk, Lawyer, Public User, Admin) must be defined. 3. All relevant data (case statuses, schedules, resources) must be preload-ed into the system. 4. The users are familiar with basic system navigation. (assumption)   **Post conditions**:  Successfully entered Dashboard:   1. The user can view and interact with role-specific features and information. 2. Actions performed on the dashboard (scheduling hearings, submitting forms) are recorded in the system. 3. Any notifications or tasks are updated in real-time.   Any Error: Stay same pages  Exception: Go to Login page. |

**Sequence Diagram:**

Case Management Page



**Functional and non-functional requirements** for the **Case Registration and Management**

## **Functional Requirements**

### **Case Registration**

1. **Fields**: Case ID (auto-generated), Case Type, Parties Involved (names, roles), Filing Date, and x Description.
2. Include **dropdown menus** for case types and courtroom selection to minimize data entry errors.
3. Enable **upload of supporting documents** ( petitions, evidence) with restrictions on file formats (PDF, JPEG, PNG) and size limits.
4. Validate mandatory fields to prevent incomplete case submissions.

### **2. Case Management**

1. Allow **status updates** for cases:
   * **Status Options**: Filed, Under Review, Hearing Scheduled, Resolved, Dismissed.
   * Automatically track litigation stages based on status changes.
2. Enable case **assignment to specific courtrooms** with an availability checker to avoid conflicts.
3. Provide functionality to schedule or reschedule hearings.
4. Allow **admins to notify relevant parties** (lawyers, judges, clerks) of updates via email or SMS.

### **3. Search and Filtering**

1. Provide a **basic search** by Case ID or Party Name for public users.
2. Offer **advanced search filters** for authorized roles.Filters: Case Type, Filing Date, Status, Parties Involved, Courtroom.
3. Return search results with summary information (Case ID, Type, Status, Filing Date).

### **4. Document Management**

1. Allow **upload of multiple documents** for each case and associate them with specific litigation stages.
2. Include a **document retrieval system** for authorized roles to search and download case documents by metadata ( Case ID, upload date).

### **5. Role-Based Access**

* Ensure features are accessible only to authorized roles:

1. **Clerks/Admins**: Full access to register, update, assign, and schedule cases.
2. **Lawyers**: View assigned cases, submit evidence, and track litigation stages.
3. **Public Users**: Limited access to search cases and view status (no editing).

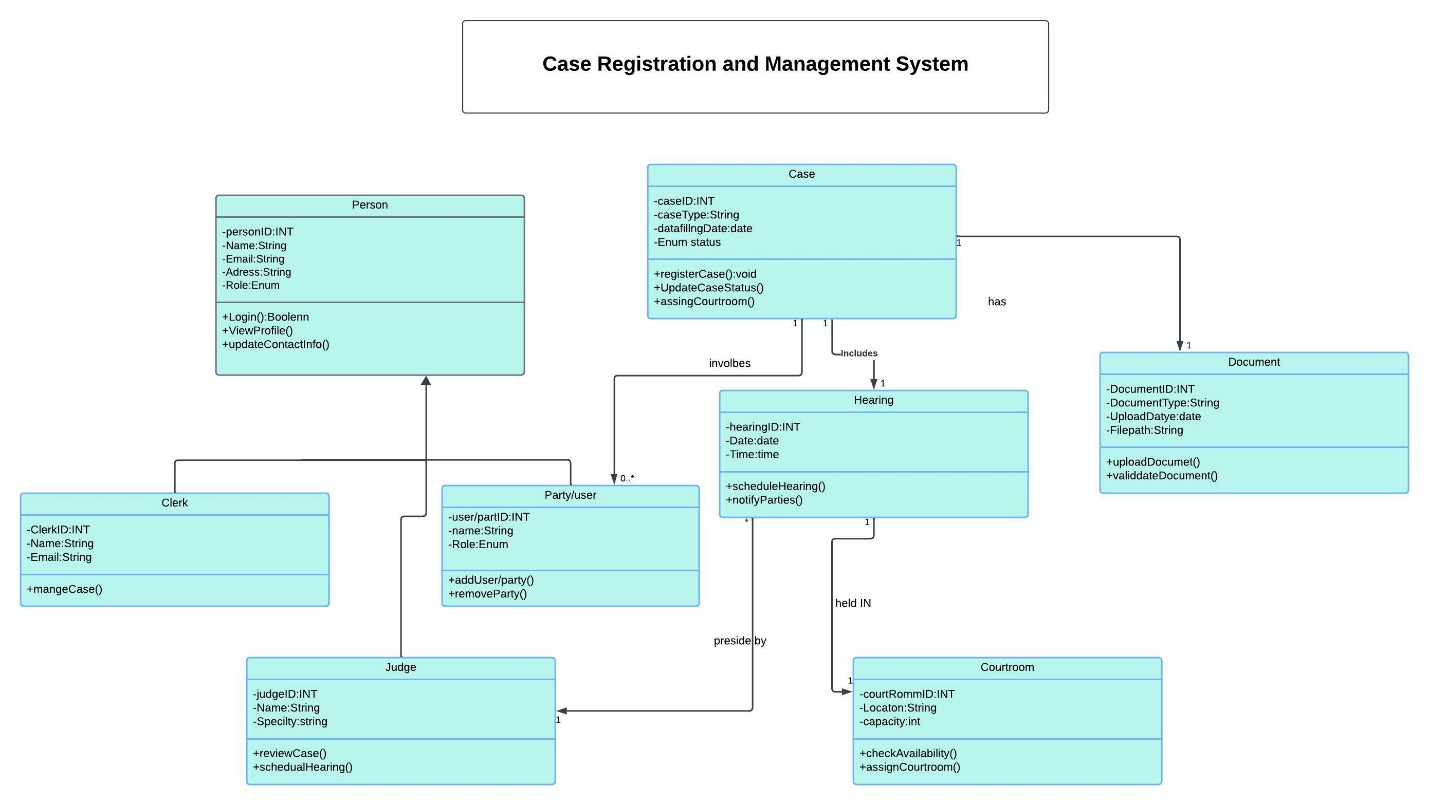
## **Non-Functional Requirements**

1. Ensure the page is **user-friendly** with an intuitive layout, form validation, and tooltips for complex fields.
2. Include clear labels and instructions to minimize errors by users unfamiliar with the system.
3. Use secure login mechanisms (multi-factor authentication) to restrict access to authorized users.
4. Encrypt sensitive data (party details, case records) during transmission and storage.
5. Prevent unauthorized document access using **dynamic session tokens** and strict user permissions.
6. Ensure the system can handle **high traffic loads** during peak hours without delays or downtime.
7. Ensure the system has **99.9% uptime**, with robust backup and recovery mechanisms.
8. Implement automated **database backups** every 30 minutes to prevent data loss

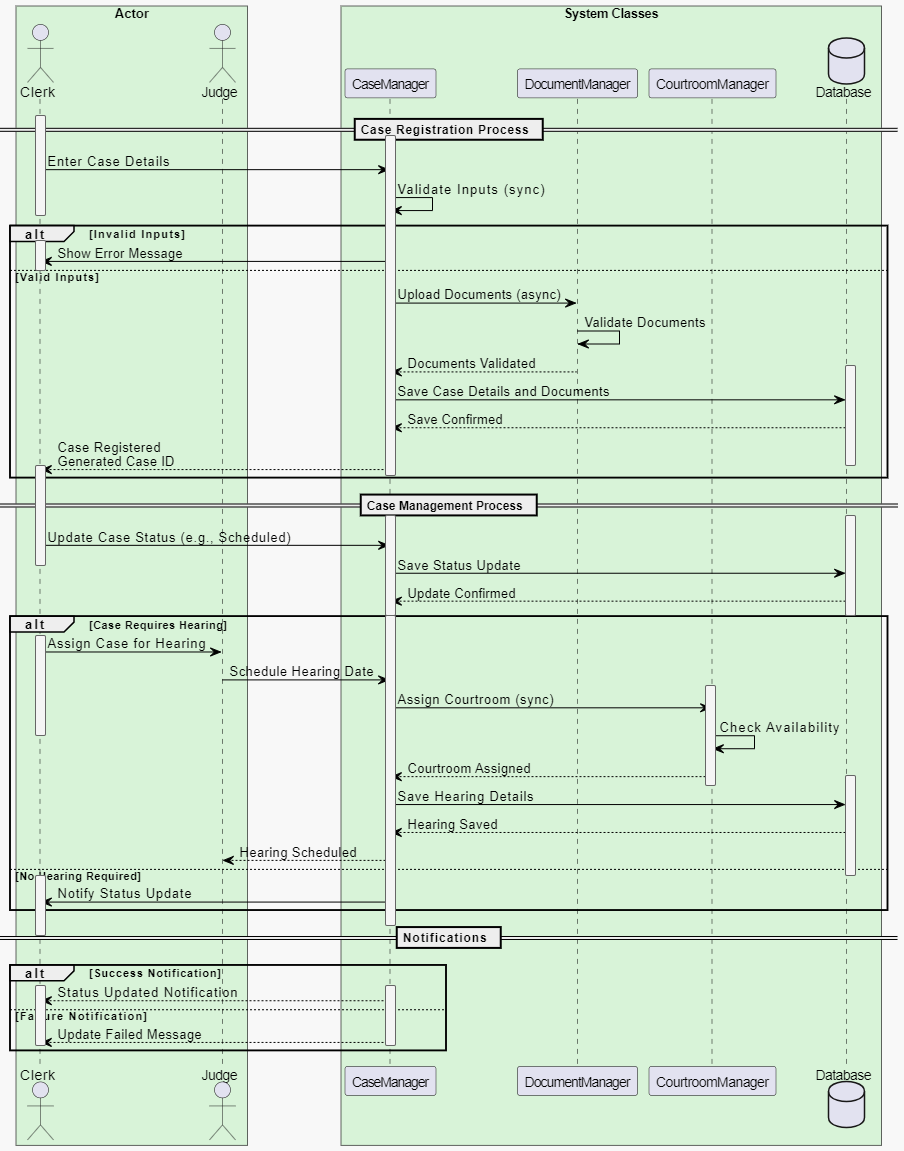
User Story For Case Management

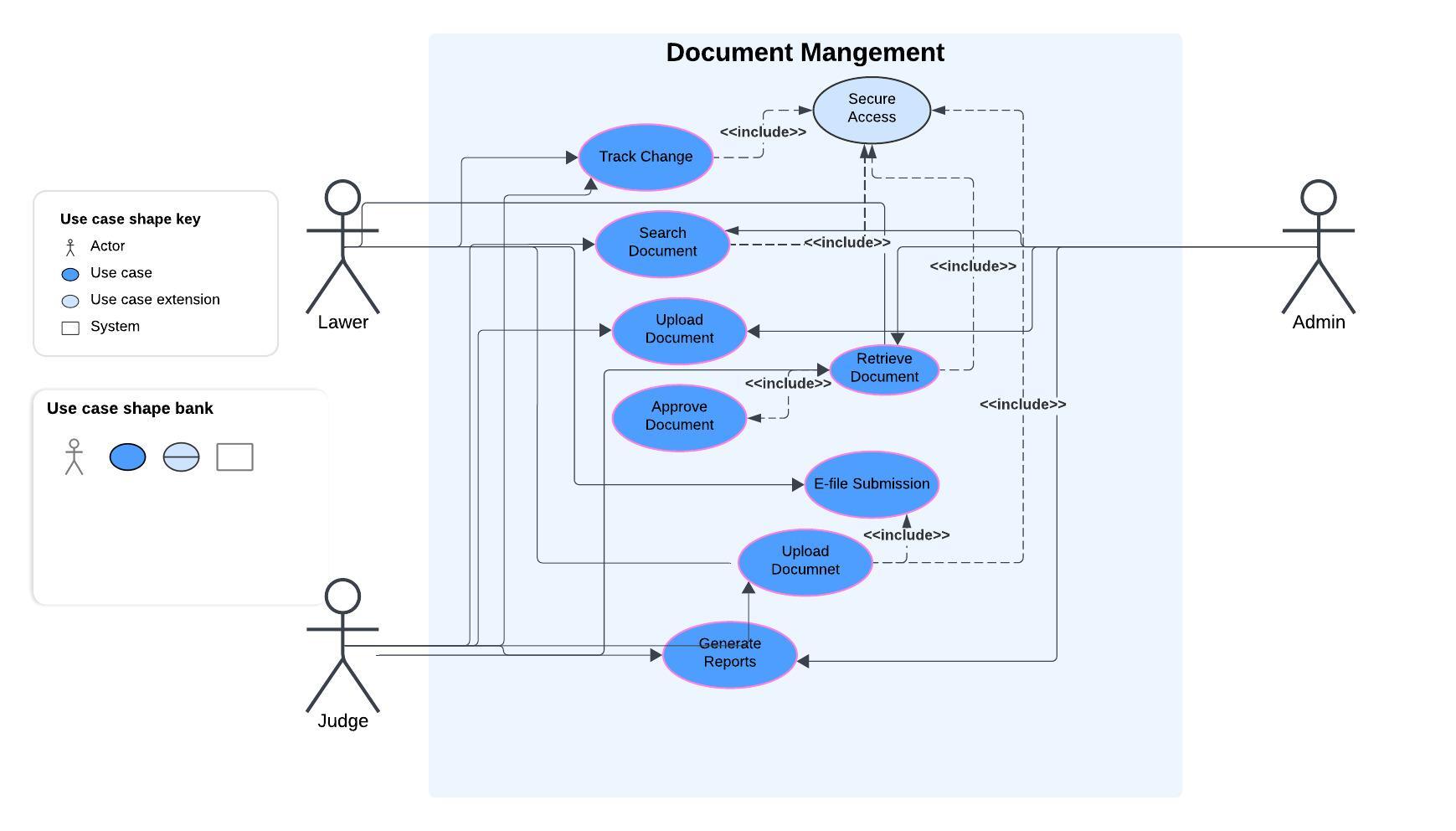


|  |
| --- |
| **Use Case Name:** Case Registration and Management |
| **Actors:**Clerk,Admin,**Lawyer,**Public User.  **Scenario:**   1. The Clerk/Admin logs into the system using valid credentials. 2. The system verifies the credentials and redirects to the dashboard. 3. The Clerk navigates to the "Register New Case" page. 4. The system displays a form with required fields (Case Type, Parties Involved, Filing Date). 5. The Clerk fills out the form and uploads supporting documents. 6. The system validates the inputs and uploads. 7. A The Clerk submits the case. 8. The system generates a unique Case ID, saves the details in the database, and redirects the Clerk to the "Pending Cases" list. 9. User view their case history and find lawyer 10. For the lawyer same way clark find case details,submit evidence.   **Exceptions:**  1.Unauthorized Access   * Condition: A user without appropriate permissions attempts to register or update a case. * System Response: The system denies access and logs the incident.   2: Document Upload Failure   * Condition: A document upload fails due to size or format restrictions. * System Response: The system prompts the user to upload a valid file.   **Preconditions:**   1. The user must have a valid account and appropriate role-based access permissions. 2. The system must be connected to the database to fetch or save case-related data. 3. Required metadata (e.g., list of case types, courtroom availability) must be preloaded into the system.   **Postconditions:**   1. The case is successfully registered, and a unique Case ID is generated. 2. Case details are saved in the database and are accessible to authorized users. 3. Notifications (if enabled) are sent to relevant parties. 4. Came some error go to log in page. |

Class Diagram For Case Registration and Management

Sequence Diagram Case Registration and Management



Class Diagrame For Document Management:

**Functional Requirement:**

1. Allow users to upload case-related documents (petitions, evidence, court orders).

2. Support multiple file types (PDF, DOCX, JPEG, etc.).

3. Ensure successful storage of documents with metadata (Case ID, upload date, upload-er role).

4. Automatically assign unique identifiers to each uploaded document.

5. Capture and manage metadata (e.g., file type, size, related case ID, upload er information).

6. Judges and Clerks: Full access to view and approve case documents.

7. Lawyers: Access only their case documents.

8. Public Users: Restricted access to view approved case-related public records.

9. Search by case ID, parties involved, document type, or upload date.

10. Use filters to narrow down search results (e.g., document status, upload er role).

11. Enable judges or authorized clerks to review and approve uploaded documents.

12. Notify users (lawyers) when their submitted documents are approved or rejected.

13. Allow lawyers and authorized users to submit documents electronically for review.

14. Provide a confirmation receipt for each successful submission.

15. Maintain a version history for updated documents.

16. Allow users to view or revert to previous versions if needed.

17. Allow users (especially clerks) to upload multiple documents at once.

18. Provide bulk download options for case bundles (e.g., all documents related to a specific case).

19. Maintain a log of all document activities (e.g., upload, review, approval, download).

20. Ensure logs include timestamp, user ID, and activity type.

21. Notify users of pending document approvals or new uploads relevant to their cases.

22. Alert users about nearing deadlines for document submission.

**Non-Functional Requirements:**

1. Handle uploads of large files (up to 100 MB) with minimal delay.

2. Enable document search to return results within 2 seconds, even with large data sets.

3. Accommodate increasing volumes of documents and concurrent users as the system grows.

4. Ensure smooth operation with tens of thousands of documents in the repository.

5. Encrypt documents during upload, storage, and retrieval to prevent unauthorized access.

6. Implement strict role-based access controls to ensure only authorized users can view or modify documents.

7. Ensure regular backups of the document repository to prevent data loss.

8. Provide quick recovery mechanisms in case of system failures.

9. Ensure the interface for uploading and retrieving documents is intuitive and user-friendly.

10. Offer tool-tips and error messages to guide users in case of upload or search issues.

11. Adhere to legal and regulatory standards for data protection ( GDPR, HIPAA, or country-specific regulations).

12. Ensure proper handling of sensitive and confidential documents.

13. Integrate with external systems (e.g., third-party storage solutions or court databases) for seamless data exchange.

14. Support API connections for document retrieval and updates from other systems.

15. Ensure 99.9% uptime for the document management system.

16. Provide consistent performance across all user roles and locations.

17. Support multiple languages for document metadata and user interface.

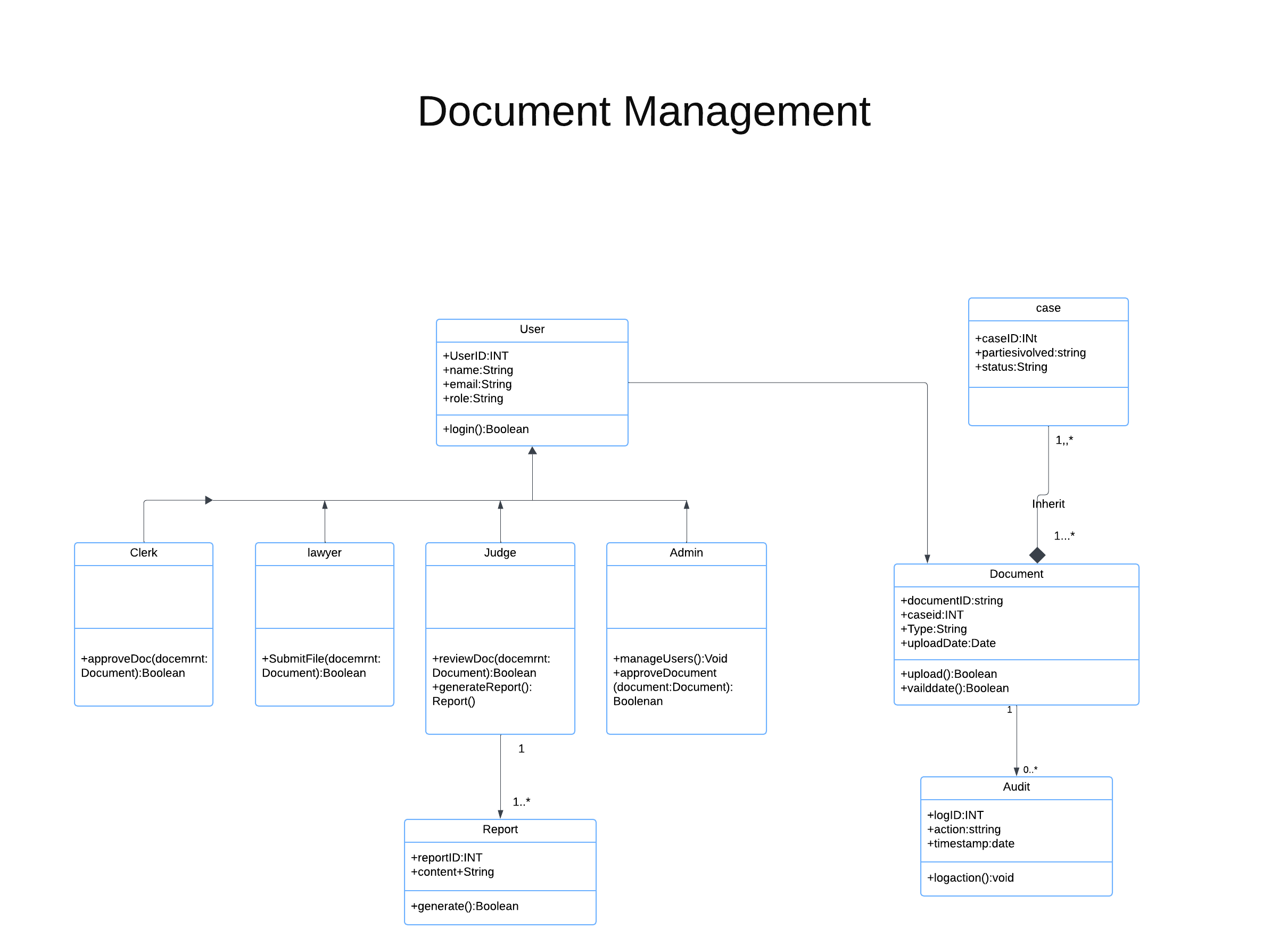
18. Provide detailed error messages for upload failures (e.g., file too large, unsupported format).

19. Implement retry mechanisms for failed uploads.

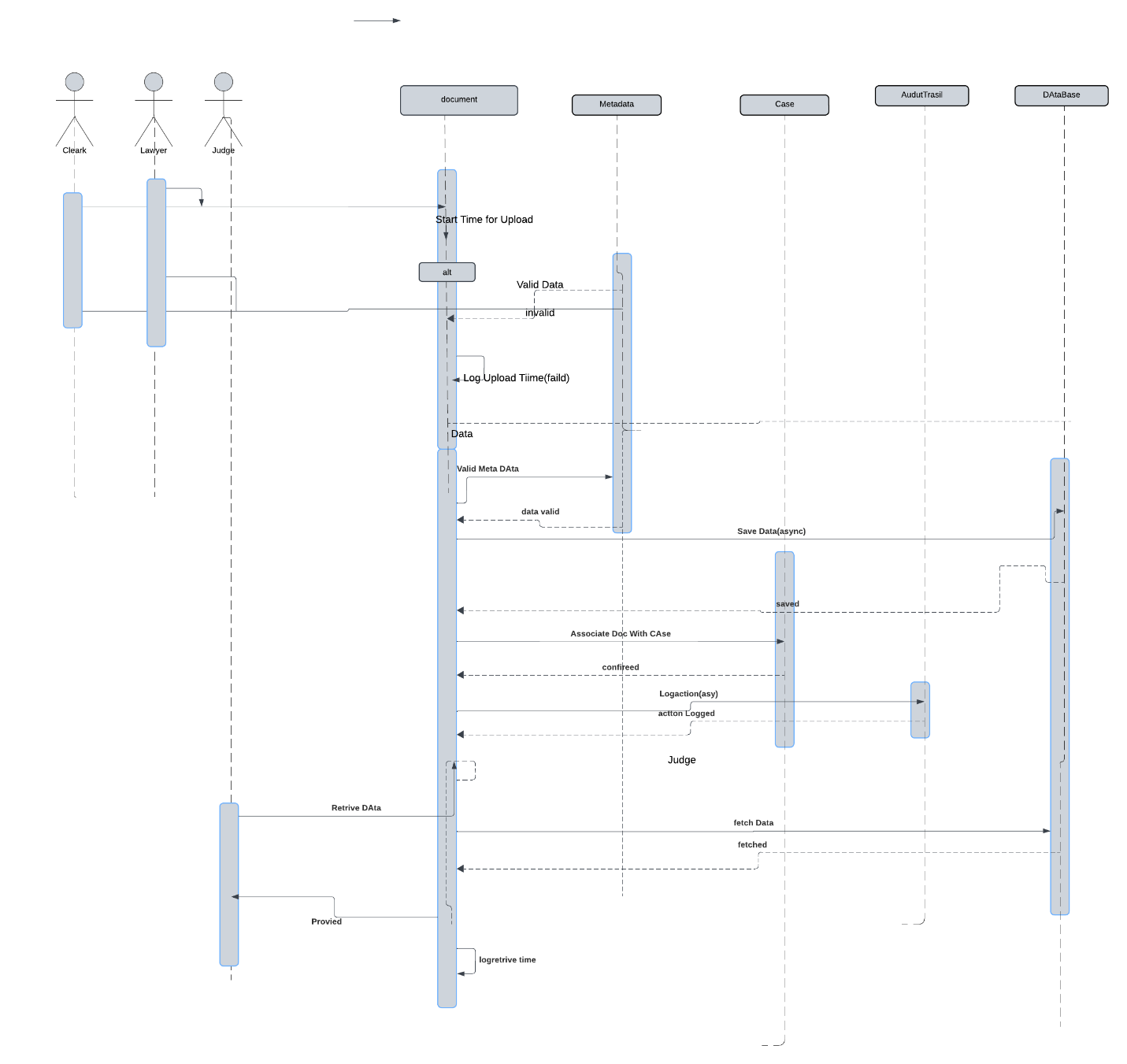


**Use Case Specification: Document Management – Upload Documen**t

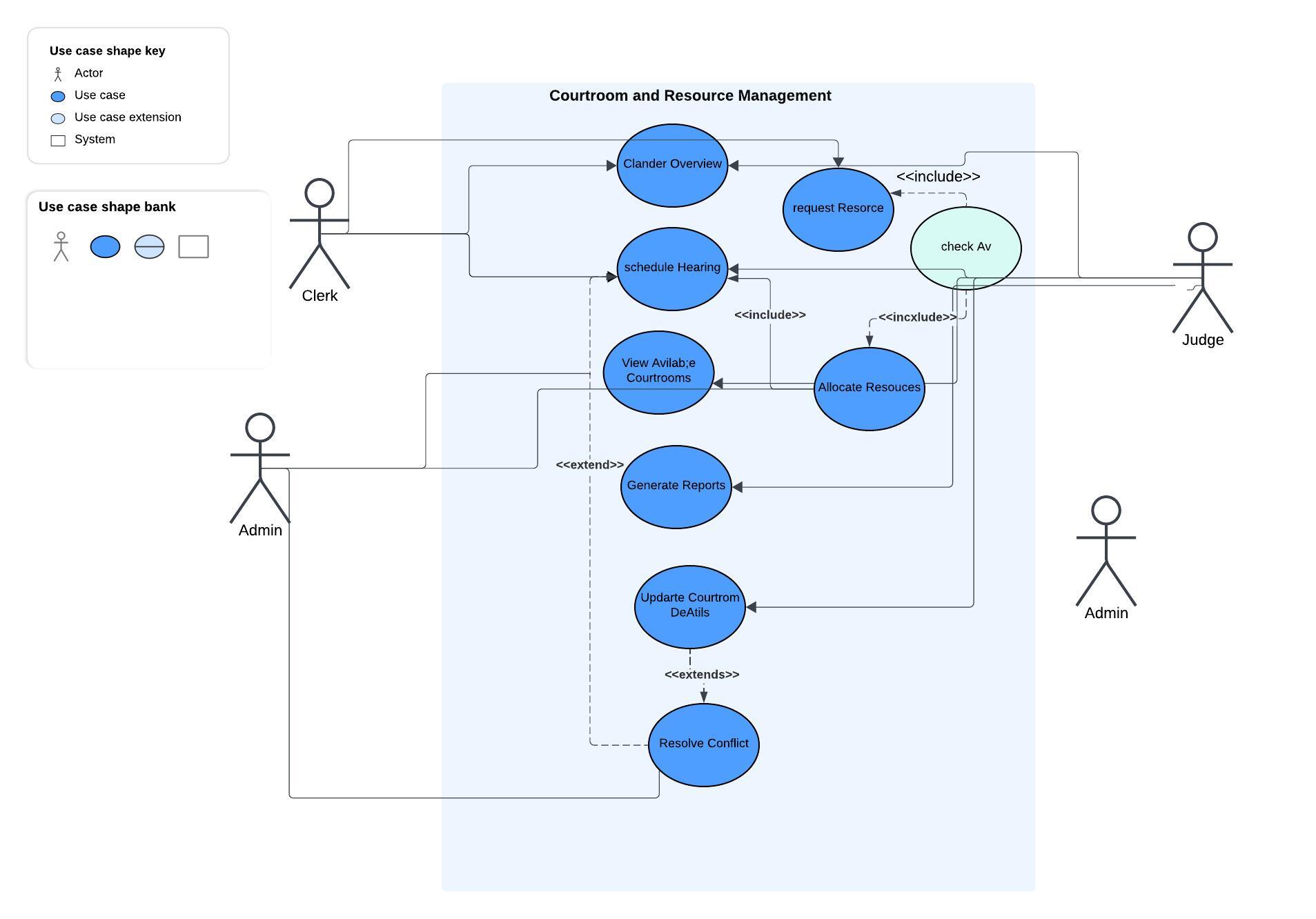
|  |
| --- |
| **Use Case Specification: Document Management – Upload Documen**t |
| **Actor:**Clerk,Lawyer,Admin  **Description/Scenario :**   1. The actor logs into the system with valid credentials. 2. The actor navigates to the "Document Management" module. 3. The actor selects the "Upload Document" option. 4. The system displays a form for uploading documents with mandatory metadata fields (Case ID, Document Type). 5. The actor fills in the required metadata and selects the file(s) to upload. 6. The system validates the file format, size, and completeness of metadata. 7. If validation passes, the document is stored in the system with a unique identifier. 8. The system updates the case record with the new document status. 9. The system sends a confirmation notification to the actor. 10. The system logs the upload activity in the audit trail.   **Exceptions:**  **Invalid File Format**: Error message for unsupported file formats.  **File Size Exceeded:** Error message for exceeding maximum file size.  **Missing Metadata:** Error message for incomplete metadata.  **Authentication Failure:** Redirect to the login page or error message for unauthorized access.  **Precondition:**   1. The actor must be logged into the system with valid credentials. 2. The actor must have appropriate upload permissions. 3. The metadata for the related case must exist in the system. 4. The system must be operational and connected to the storage server.   **Post-condition:**   1. The document is securely uploaded and indexed with a unique identifier. 2. The system updates the case record with the document status. 3. A confirmation notification is sent to the actor. 4. The document is available for search and retrieval by authorized users.   The upload activity is logged in the audit trail |



Sequence Diagrame For Document Mangement



**Courtroom and Resource Management system**Page



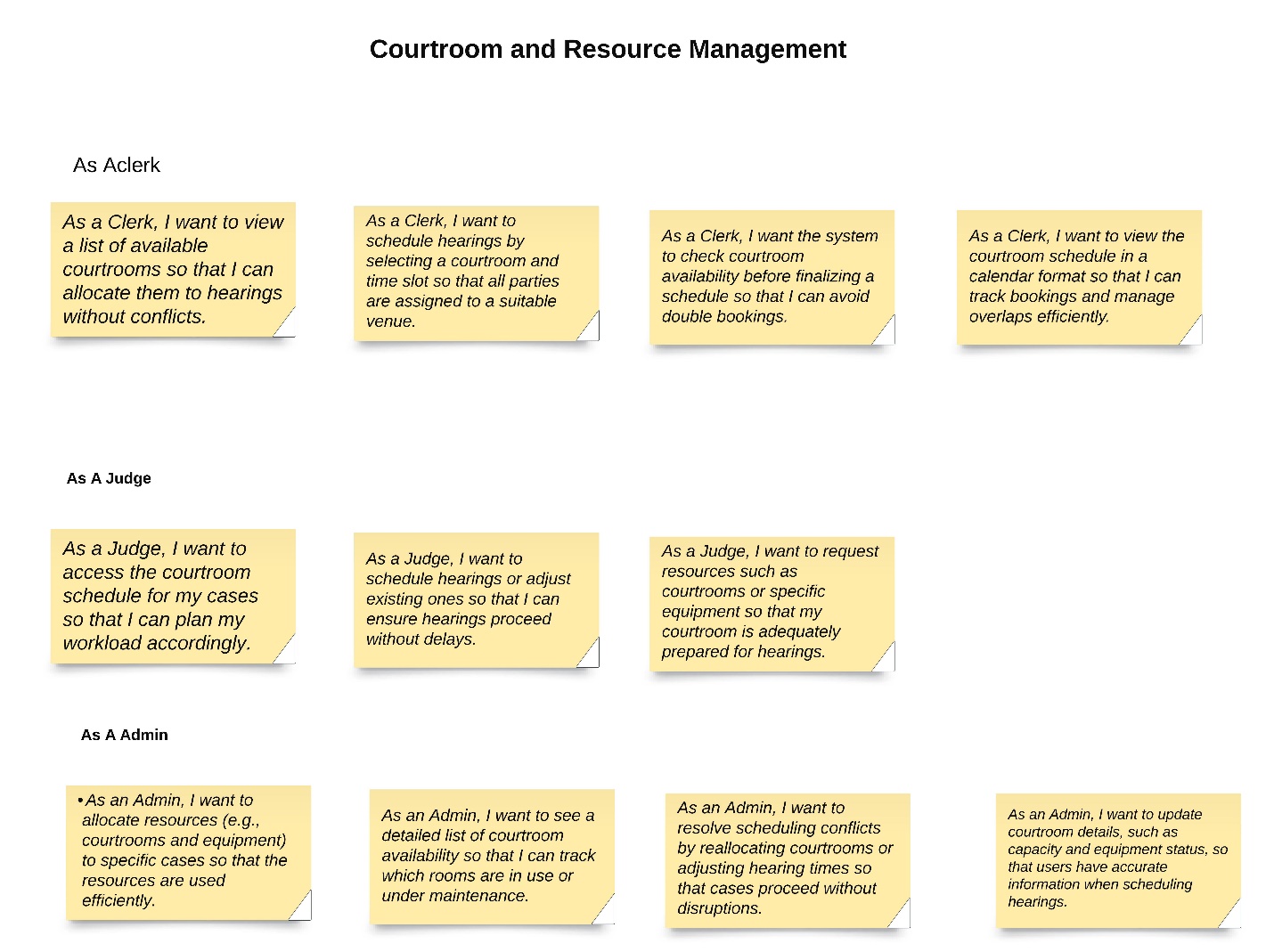
**Functional and non-functional requirements for the Courtroom and Resource Management system:**

**Functional Requirements**

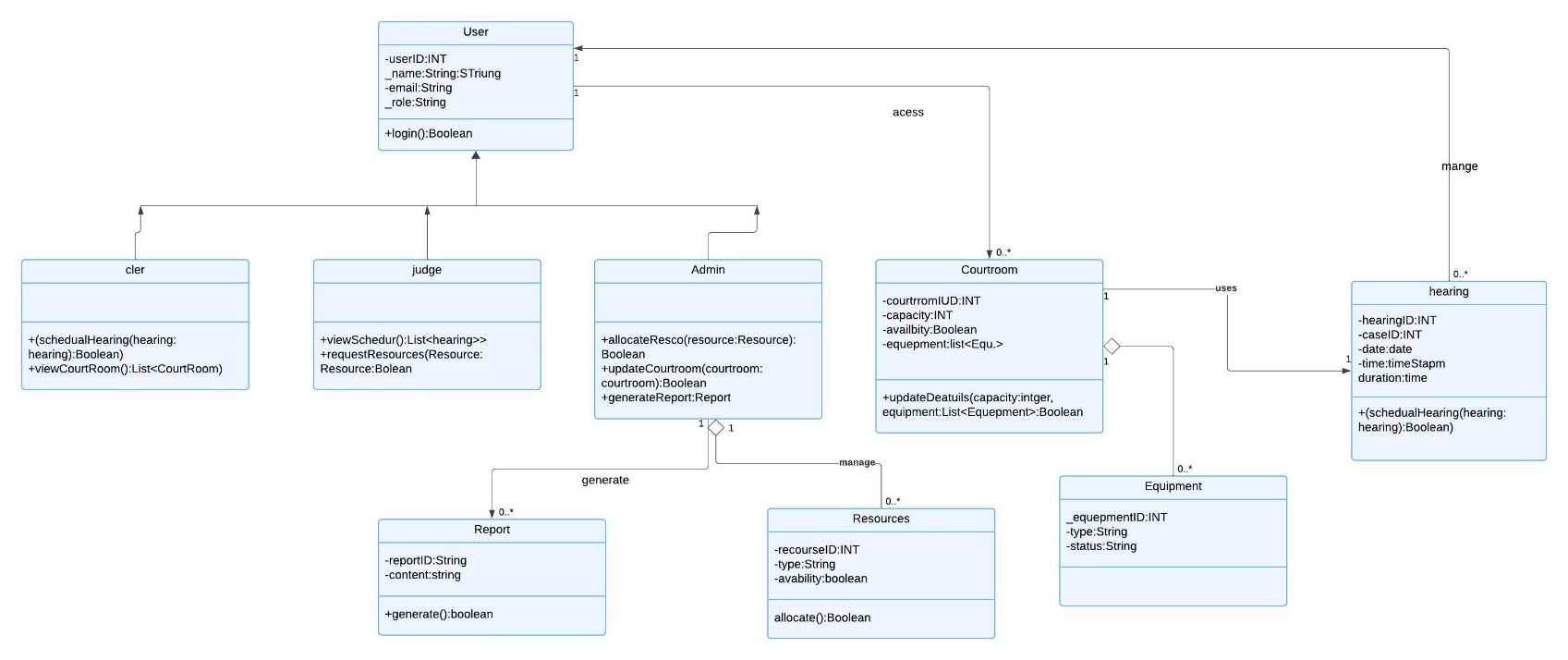
1. The system must allow users (Clerks, Judges, Admins) to view a list of available courtrooms in real-time.
2. Availability must include detailed attributes such as room capacity, technology (video conferencing equipment), and accessibility.
3. Users must be able to schedule hearings by selecting available courtrooms and specifying the date, time, and duration.
4. The system should check for scheduling conflicts before confirming the hearing.
5. The system must validate the availability of courtrooms, resources, and required equipment before scheduling or allocation.
6. This process should be included in both scheduling hearings and resource allocation workflows.
7. The system must provide a calendar interface for users to view the complete schedule of courtrooms, highlighting booked and available time slots.
8. The calendar should allow filtering by date, courtroom, or judge.
9. Admins must be able to allocate resources (courtroom, equipment, clerical support) for specific hearings or events.
10. Resource allocation must consider dependencies such as courtroom availability and required equipment.
11. The system must notify the user when a scheduling or resource conflict arises.
12. Admins must have tools to resolve conflicts by rescheduling hearings, reallocating resources, or marking priority cases.
13. Admins must be able to update courtroom configurations, such as capacity, equipment, or availability status ("Under Maintenance").
14. Updates should trigger automatic recalculations of courtroom availability.
15. Admins must be able to generate reports on courtroom usage, resource allocation, and scheduling conflicts.
16. Reports should include metrics such as utilization rates, resolved conflicts, and hearing statistics.
17. Judges and Clerks must be able to request additional resources (e.g., courtrooms, special equipment) for specific hearings or cases.
18. Requests should be logged for approval by Admins.
19. The system must enforce role-based access:
20. Clerk: Schedule hearings, view courtrooms, request resources.
21. Judge: View schedules, request resources.
22. Admin: Full access, including resolving conflicts, updating courtroom details, and generating reports.

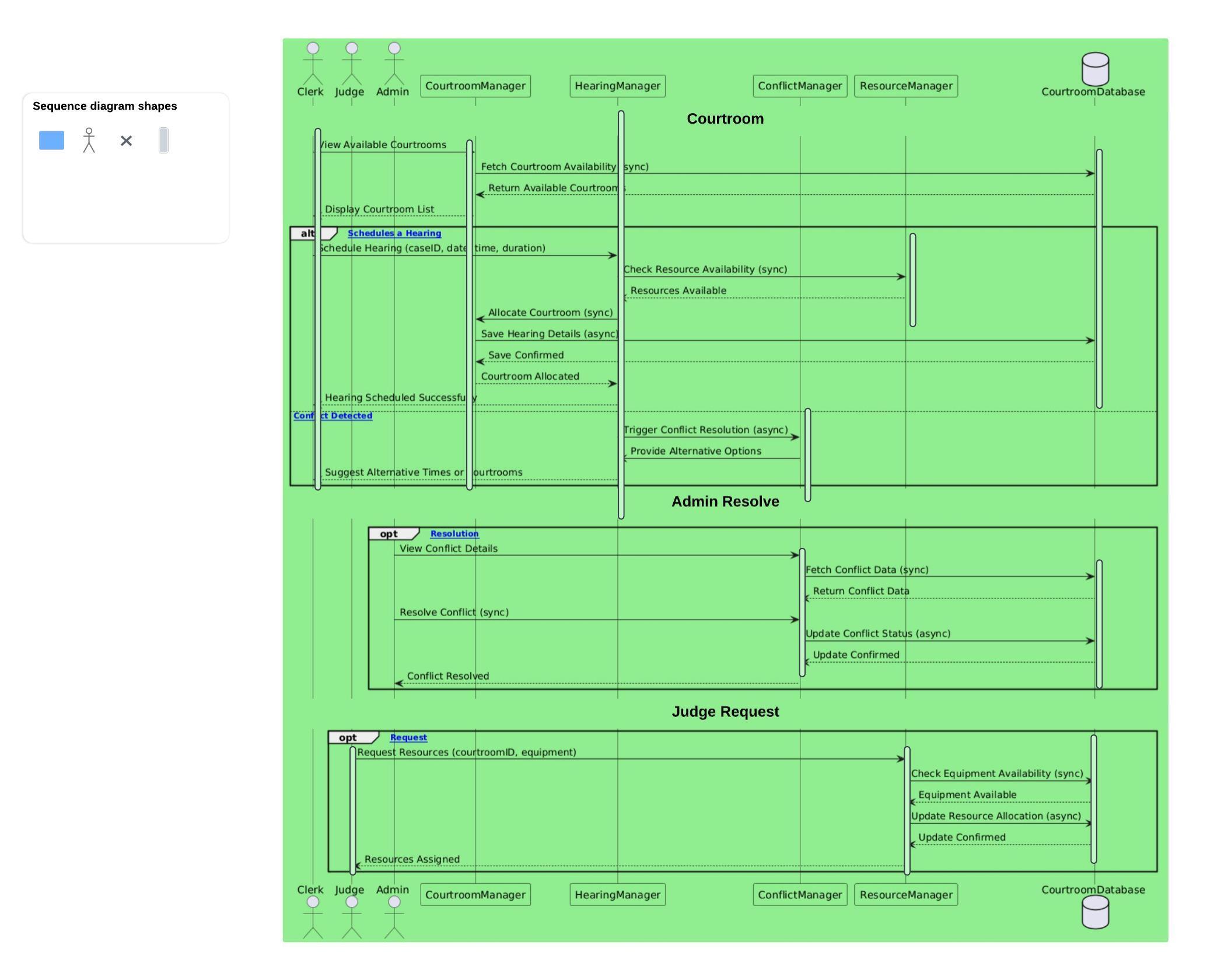
**Non-Functional Requirements**

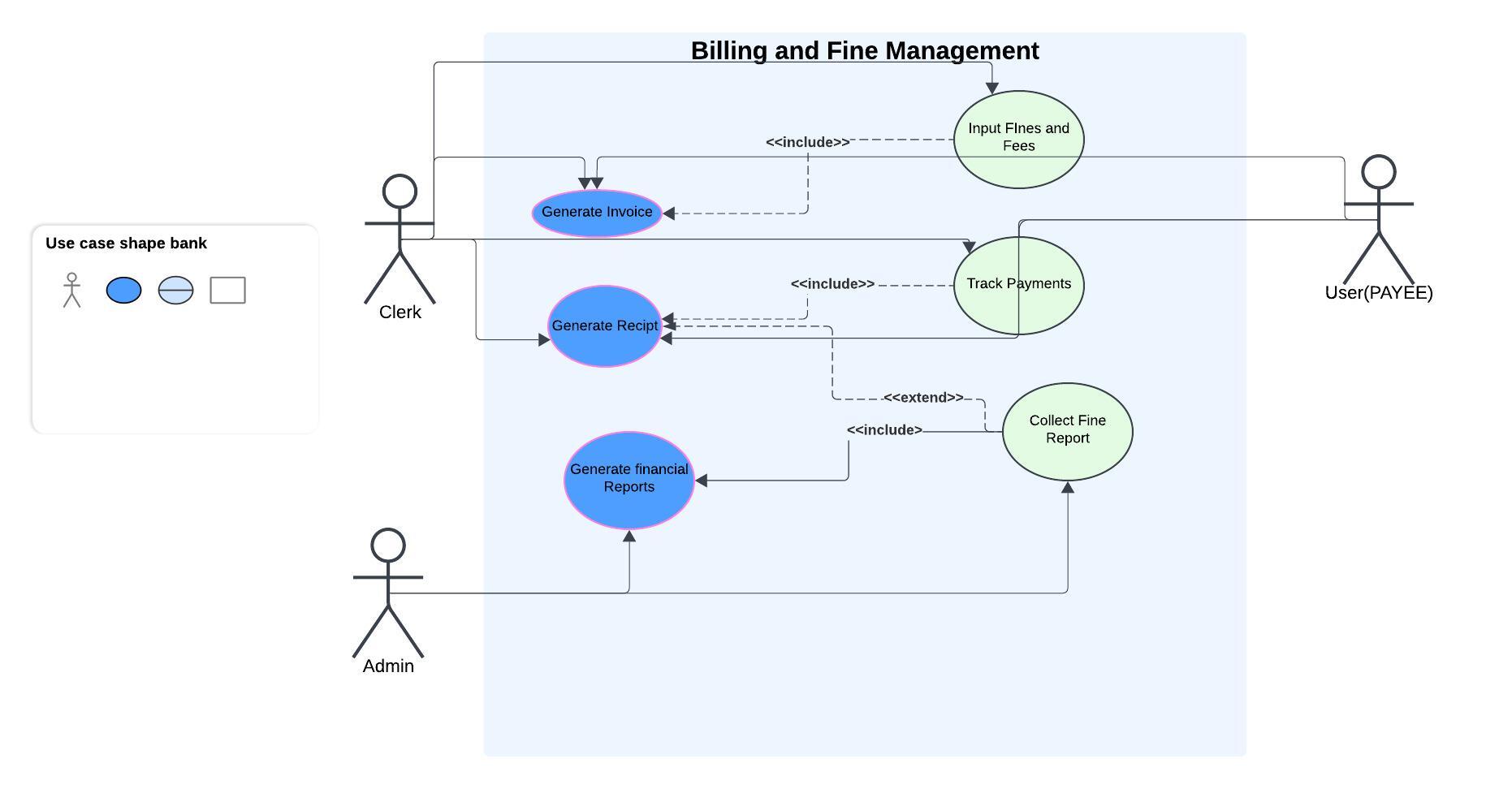
1. The system must handle up to 100 simultaneous users with minimal latency (2 seconds for viewing courtrooms or calendars).
2. The system should process scheduling operations within 5 seconds.
3. The system should support the addition of new courtrooms, resources, and users without performance degradation.
4. The calendar view should scale to support multiple courtrooms across multiple dates without slowing down.
5. The system must ensure 99.9% uptime to prevent disruptions to courtroom management.
6. Scheduled hearings and resource allocations must not be lost due to system failures.
7. The system must have an intuitive interface, with clear navigation for Clerks, Judges, and Admins.
8. The calendar interface must be user-friendly, supporting drag-and-drop for rescheduling hearings.
9. Role-based access control (RBAC) must be implemented to restrict access to certain functionalities based on user roles.
10. Data (courtroom schedules, hearing details) must be encrypted during transmission and storage.



|  |
| --- |
| **Use Case Name:**Courtroom and Resource Management |
| Actor:Clerk, Judge, Admin  **Description/Scenario:**   1. The Clerk/judge logs into the system with appropriate credentials and navigates to the "Courtroom and Resource Management" module. 2. The actor selects an option:   Clerk or Judge: Views courtroom availability or schedules hearings.  Admin: Allocates resources, resolves conflicts, or updates courtroom details.   1. The Clerk inputs the case details (case ID, date, duration). 2. The system checks for conflicts and verifies resource availability. 3. If no conflicts are found, the system schedules the hearing and updates the calendar. 4. If conflicts are detected, the system suggests alternative slots or rooms. 5. The actor selects the required resources (courtrooms, equipment). 6. The system validates the selection and updates resource status accordingly. 7. The system flags conflicts, such as overlapping bookings or unavailable equipment. 8. The Admin reallocates resources or reschedules hearings to resolve the issue. 9. The Admin modifies room configurations (capacity, equipment status). 10. The system updates all dependent modules, ensuring no conflicts arise. 11. The system logs all actions (scheduling, resource allocation, updates) for auditing purposes. 12. The Clerk/judge exits the module or continues with additional tasks.   **Exceptions:**   1. If no suitable courtroom is available for the requested time slot, the system displays an error message: "No available courtroom for the selected time. Please choose another time or room." 2. If a scheduling conflict occurs, the system blocks the action and suggests alternative slots: "Selected courtroom is already booked. Try a different time or room." 3. If required fields (e.g., case ID, date) are missing, the system displays an error message: "Please complete all mandatory fields. |

****



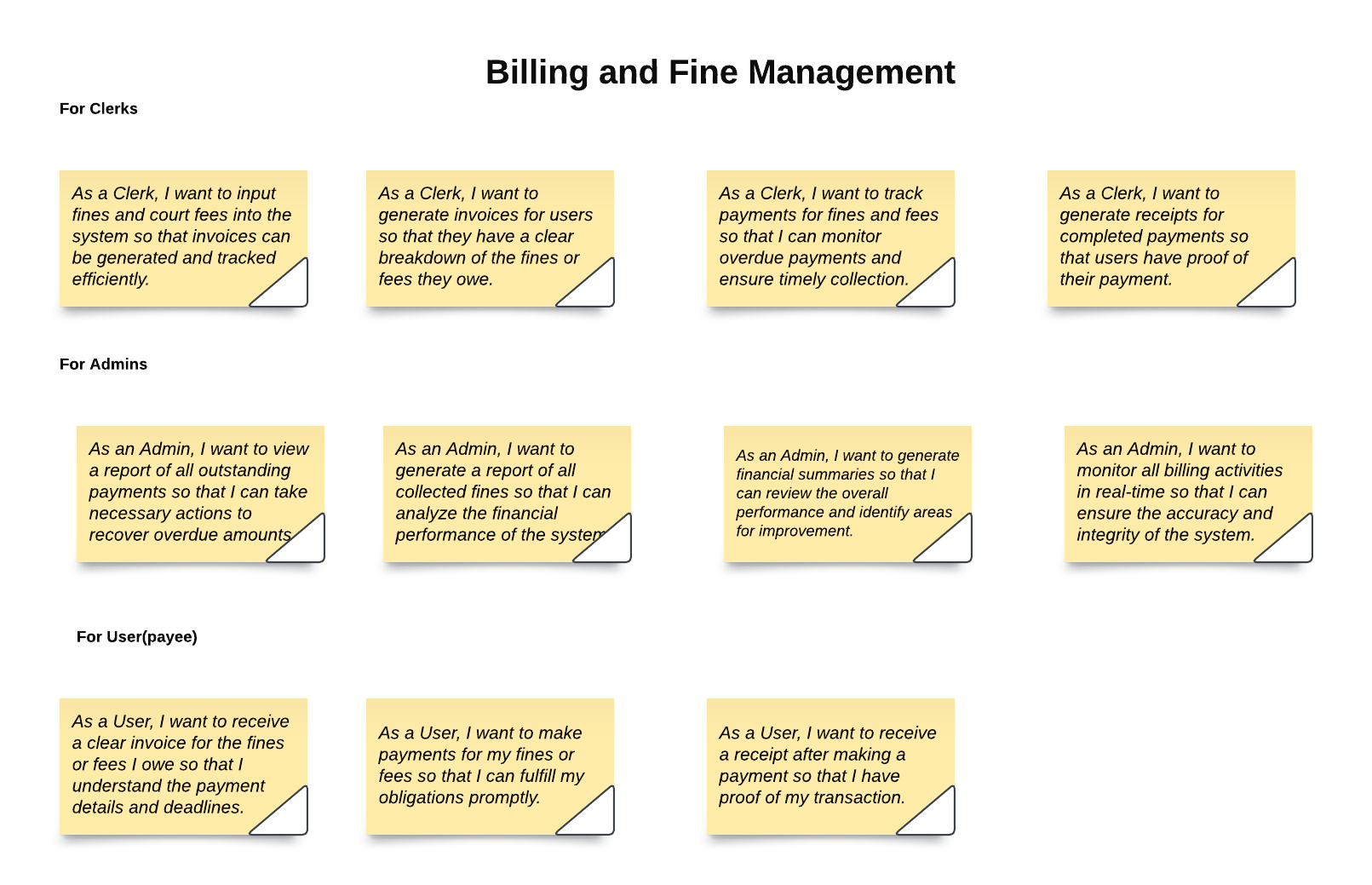
Page: Billing and Fine Management

Functional Requirements:

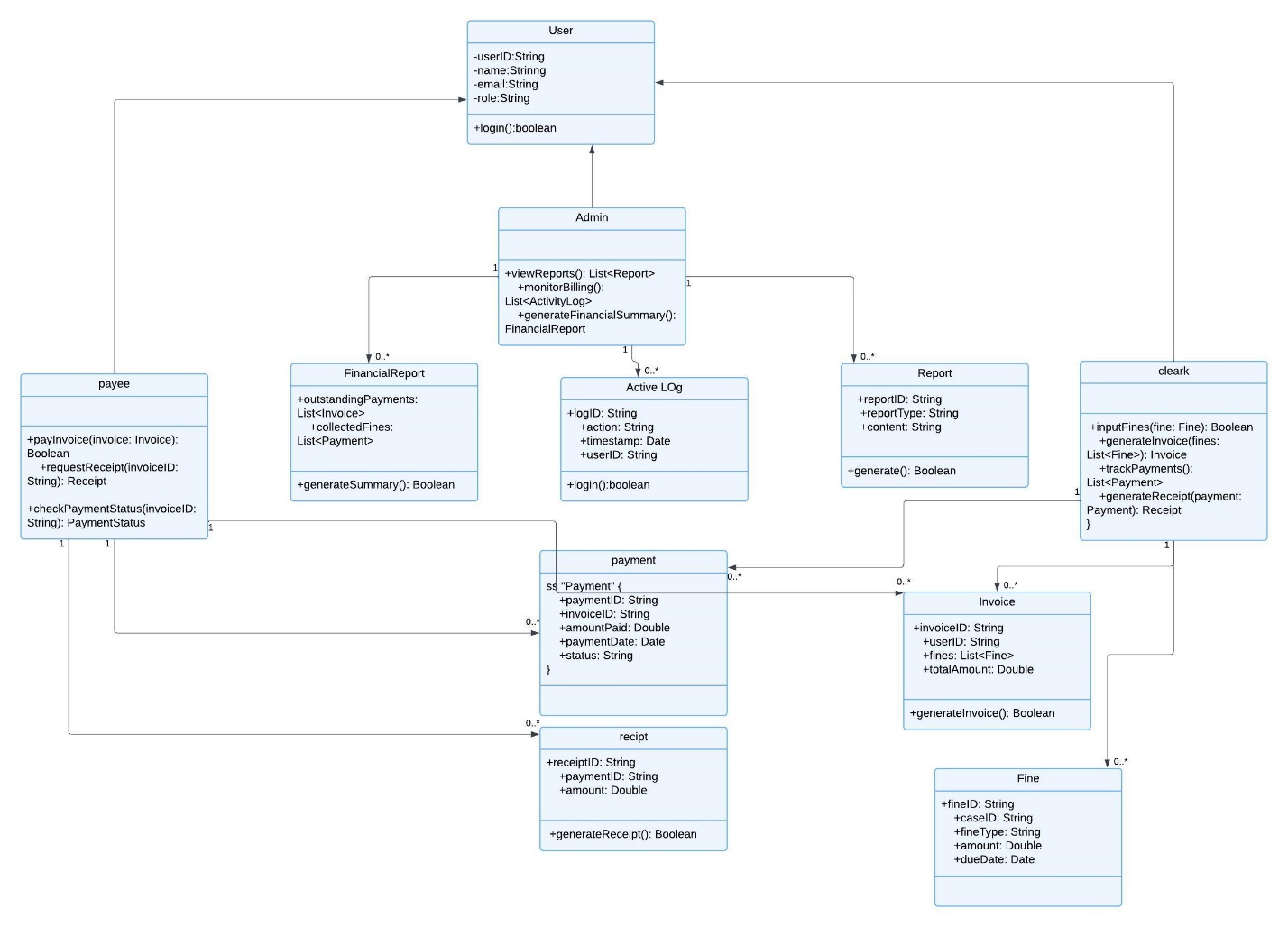
1. The system must allow clerks to input fines and fees for specific cases and users.
2. Users must be able to view and pay their invoices online.
3. The system must generate detailed invoices including fine details, amounts, and due dates.
4. Payment tracking must update statuses in real-time and generate receipts after successful payments.
5. The system must allow admins to generate financial reports, including outstanding payments and collected fines.
6. Invoices, payments, and receipts must be securely stored in the database.
7. The system must send notifications to users for overdue fines and payment confirmations.
8. The system must validate all inputs, such as fine amounts and payment details, before processing.
9. Admins must be able to monitor billing activity logs for auditing purposes.

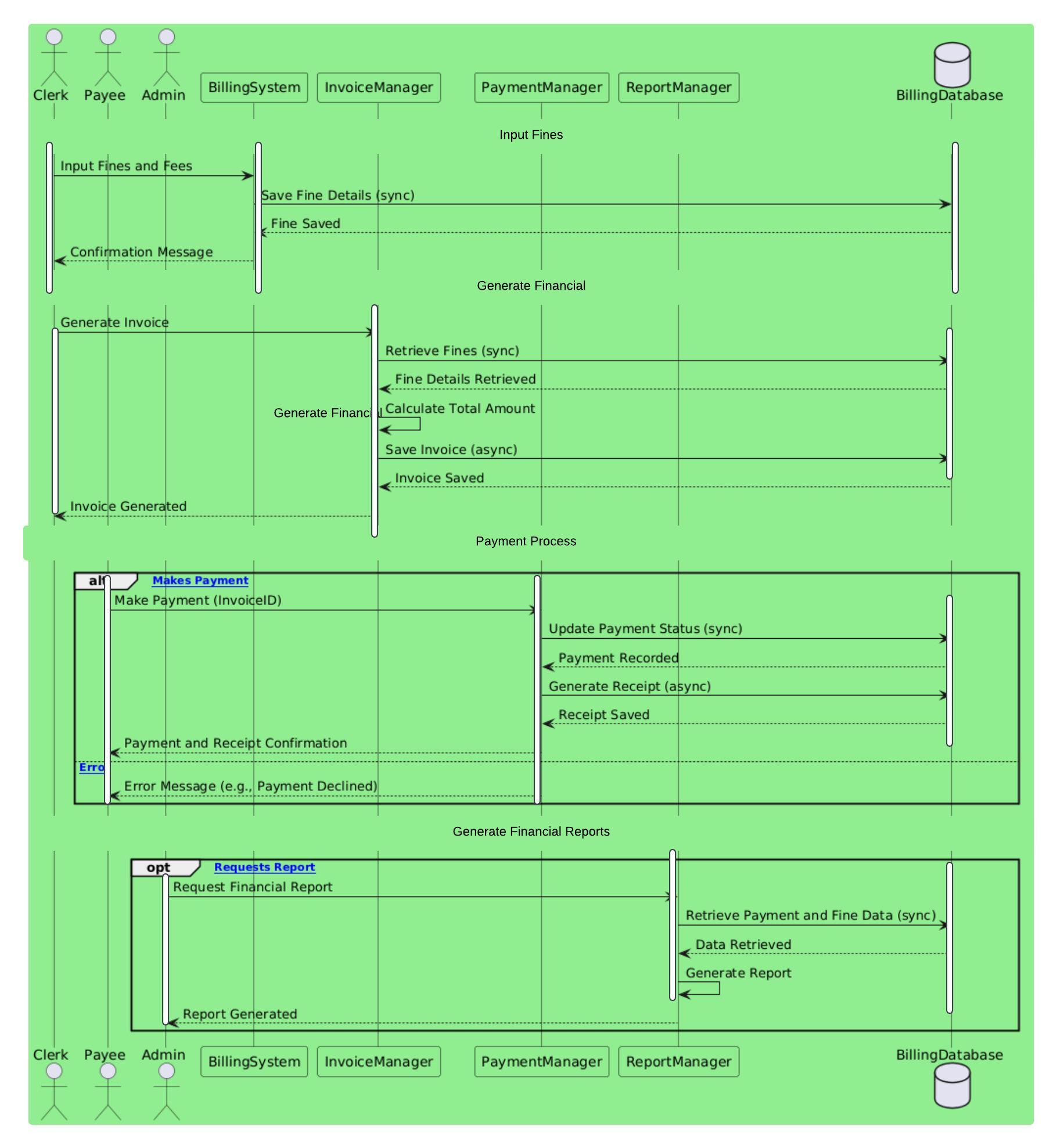
Non-Functional Requirements:

1. The system must respond to user actions, such as invoice generation and payment processing, within 3 seconds.
2. The system must handle up to 100 concurrent users without performance degradation.
3. All sensitive data, such as payment and user details, must be encrypted during storage and transmission.
4. The interface must be intuitive and accessible to users with varying technical expertise.
5. The system must maintain 99.9% uptime to ensure availability for billing and payment operations.
6. All payment transactions must comply with relevant financial and data protection regulations.
7. Reports must be exportable in PDF and Excel formats for easy sharing and offline use.
8. Audit logs must be retained for a minimum of 5 years to ensure traceability of billing activities.
9. The system must support multi-language interfaces for a diverse user base.



|  |
| --- |
| **Use Case Name**:Billing and Fine Management |
| **Actor:**Clerk, Payee (User), Admin  **Description/Scenario:**   1. The Clerk inputs fines and fees into the system using the case or user details. 2. The system validates the input and stores the fines in the database. 3. The Clerk generates an invoice based on the entered fines, which is sent to the Payee. 4. The Payee views the invoice and makes a payment using available payment methods. 5. The system processes the payment, updates the payment status, and generates a receipt. 6. The Admin generates financial reports, such as outstanding payments or collected fines, for auditing and review. 7. The system logs all actions, such as invoice creation, payment processing, and report generation, for auditing purposes.   **Exceptions:**   1. If required input fields are incomplete, the system shows an error message and blocks submission. 2. If payment processing fails, the system notifies the Payee with an error message. 3. If database connectivity is lost, the system displays a notification and retries the operation. 4. If an Admin attempts to generate a report for a nonexistent date range, the system displays a No Data Found message.   Precondition:   1. The user must be logged into the system with the appropriate role and permissions. 2. Fines and fees must be entered into the system before invoices can be generated. 3. Payment methods must be pre-configured and operational for successful transactions.   Postcondition:   1. Fines and fees are successfully recorded in the system. 2. Invoices are generated and sent to the Payee for payment. 3. Payment statuses are updated, and receipts are generated for completed transactions. 4. Financial reports are generated and saved for review and auditing purposes.   The system logs all operations for traceability and compliance |





Database:

Database Tables

1. Users

Table Name: users

Columns:

user\_id (PK): Unique identifier for the user.

name: Full name of the user.

email: Email address of the user.

password: Encrypted password.

role: Role of the user (Clerk, Admin, Judge, Payee).

created\_at: Timestamp of user creation.

updated\_at: Timestamp of last update.

2. Courtrooms

Table Name: courtrooms

Columns:

courtroom\_id (PK): Unique identifier for the courtroom.

capacity: Maximum capacity of the courtroom.

availability: Availability status (Available, Booked, Under Maintenance).

equipment: List of available equipment (JSON format or relational).

created\_at: Timestamp of courtroom creation.

updated\_at: Timestamp of last update.

3. Cases

Table Name: cases

Columns:

case\_id (PK): Unique identifier for the case.

case\_type: Type of case (e.g., civil, criminal).

parties\_involved: List of involved parties (JSON or relational).

status: Current case status (Filed, Hearing Scheduled, Resolved).

created\_at: Timestamp of case creation.

updated\_at: Timestamp of last update.

4. Hearings

Table Name: hearings

Columns:

hearing\_id (PK): Unique identifier for the hearing.

case\_id (FK): References cases(case\_id).

courtroom\_id (FK): References courtrooms(courtroom\_id).

date: Scheduled date of the hearing.

time: Scheduled time of the hearing.

duration: Duration of the hearing (in minutes).

status: Hearing status (Scheduled, Completed, Cancelled).

5. Fines

Table Name: fines

Columns:

fine\_id (PK): Unique identifier for the fine.

case\_id (FK): References cases(case\_id).

user\_id (FK): References users(user\_id).

fine\_type: Type of fine (e.g., late filing, contempt).

amount: Fine amount.

due\_date: Deadline for fine payment.

status: Payment status (Pending, Paid).

6. Invoices

Table Name: invoices

Columns:

invoice\_id (PK): Unique identifier for the invoice.

user\_id (FK): References users(user\_id).

fines: List of fine IDs (JSON or relational).

total\_amount: Total amount due on the invoice.

status: Invoice status (Pending, Paid, Cancelled).

generated\_at: Timestamp when the invoice was generated.

7. Payments

Table Name: payments

Columns:

payment\_id (PK): Unique identifier for the payment.

invoice\_id (FK): References invoices(invoice\_id).

amount\_paid: Amount paid in the transaction.

payment\_date: Date and time of the payment.

status: Payment status (Successful, Failed).

8. Receipts

Table Name: receipts

Columns:

receipt\_id (PK): Unique identifier for the receipt.

payment\_id (FK): References payments(payment\_id).

amount: Total amount on the receipt.

generated\_at: Timestamp when the receipt was generated.

9. Reports

Table Name: reports

Columns:

report\_id (PK): Unique identifier for the report.

report\_type: Type of report (e.g., financial summary, outstanding payments).

content: Report content (JSON or text format).

generated\_at: Timestamp when the report was generated.

10. Equipment

Table Name: equipment

Columns:

equipment\_id (PK): Unique identifier for the equipment.

courtroom\_id (FK): References courtrooms(courtroom\_id).

type: Type of equipment (e.g., projector, microphone).

status: Equipment status (Available, In Use, Under Maintenance).

11. Activity Logs

Table Name: activity\_logs

Columns:

log\_id (PK): Unique identifier for the log.

user\_id (FK): References users(user\_id).

action: Action performed (e.g., Generated Invoice, Scheduled Hearing).

timestamp: Time of the action.