B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Object-Oriented Modeling – 23CS5PCOOM

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering in
Computer Science and Engineering

Submitted by:

Pranav Easwar

1BM23CS365

Department of Computer Science and Engineering B.M.S.
College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
August 2025-December 2025

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Modeling(23CS5PCOOM) laboratory has been carried out by)Pranav Easwar(1BM23CS365) during the 5th Semester August 2025-December 2025

.

Signature of the Faculty Incharge:

Dr. Adarsha Sagar H V

Assistant ProfessoR

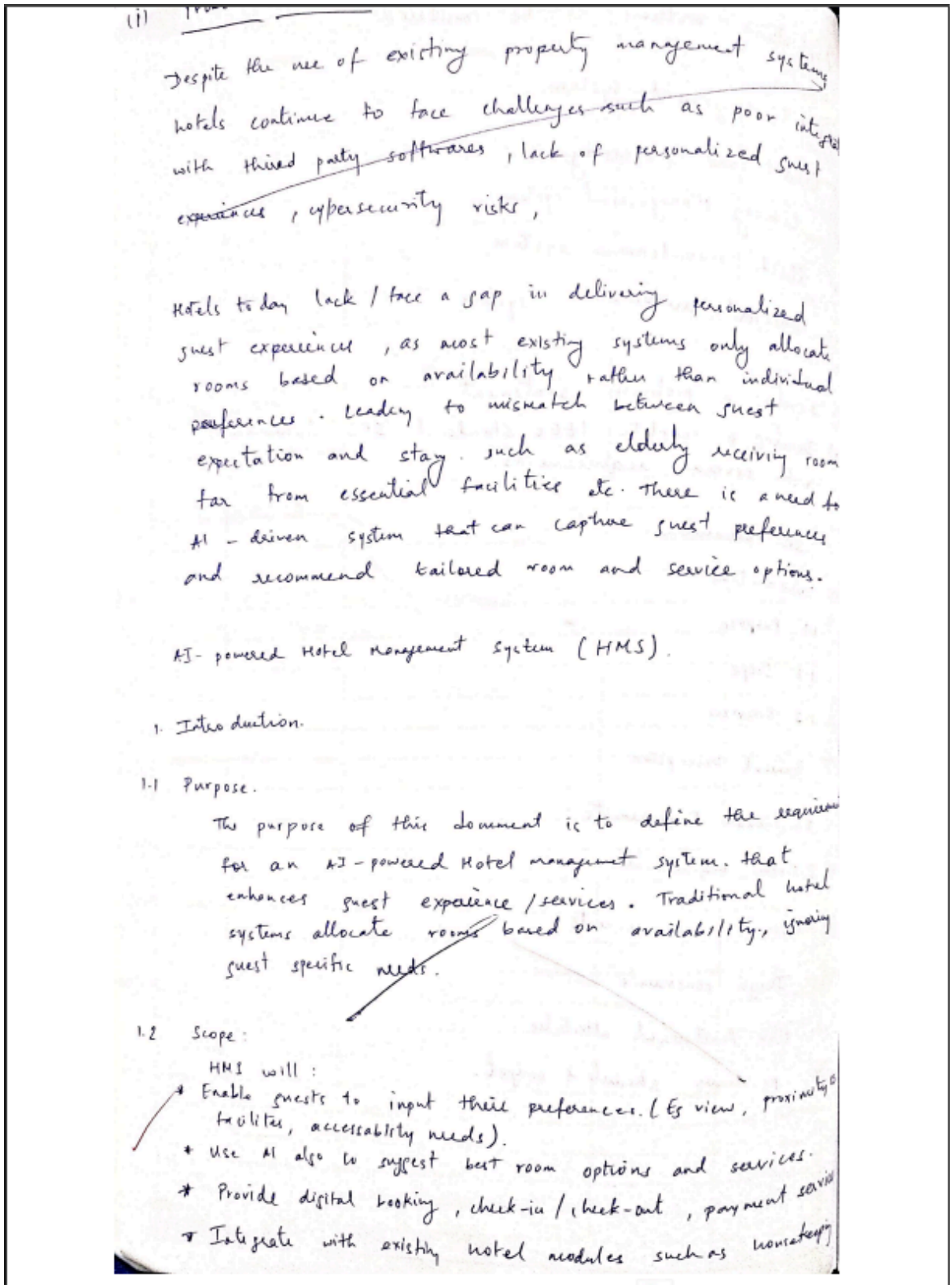
Department of Computer Science and Engineering B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System
2. Credit Card Processing
3. Library Management System
4. Stock Maintenance System
5. Passport Automation System

1. Hotel Management System

SRS:



(i)

Despite the use of existing property management systems hotels continue to face challenges such as poor integration with third party softwares, lack of personalized guest experiences, cybersecurity risks,

Hotels today lack / face a gap in delivering personalized guest experiences, as most existing systems only allocate rooms based on availability rather than individual preferences. Leading to mismatch between guest expectation and stay. such as elderly receiving room far from essential facilities etc. There is a need for AI - driven system that can capture guest preferences and recommend tailored room and service options.

AI- powered Hotel management system (HMS).

1. Introduction.

1.1 Purpose.

The purpose of this document is to define the requirements for an AI - powered Hotel management system. that enhances guest experience / services. Traditional hotel systems allocate rooms based on availability, ignoring guest specific needs.

1.2 Scope:

HMI will:

* Enable guests to input their preferences. (Eg view, proximity to facilities, accessability needs).
* Use AI also to suggest best room options and services.
* Provide digital booking, check-in / check-out, payment services
* Integrate with existing hotel modules such as housekeeping

restaurant reservations etc.

* Securely store guest preferences to enhance future stays with personalized recommendation.

System will be used by guests, front-desk, admins via mobile, web & dashboard interface.

1.3 Overview

HMS will act as an intelligent property management platform with strong emphasis on personalized and guest-centric service.

2. General Description

Users : Guests, hotel staff, administrators.

System interface : Web-based application, mobile guest app, staff dashboard.

Assumptions : Guests are willing to provide preference details. Hotel has infra to support AI-based suggestions.

Dependencies : Integrate with payment gateways, OTA platforms, AI recommendation engine etc.

3. Functional Requirements.

The system shall :

1. Allow guests to input personal preferences (Eg: quiet room, near elevator, scenic view, wheelchair access).
2. Store guest profile to learn from past stays.
3. Use AI algo to recommend suitable rooms.
4. Allow digital checkout / checkin with mobile keys.
5. Service requests via mobile app.

7. Generate reports
8. Notify staff about special guest requirement.

4. Interface requirements.

- Guest Interface :
  - Mobile app/web portal for booking, personalized recommendations
- Staff Interface :
  - To monitor guest needs, handle alerts on special requests
- Admin Interface :
  - tools, system config, staff scheduling & service monitor
- External Interface :
  API integration with OTA platform, payment gateway.

5. Performance Requirements.

- AI recommendation engine should generate room/service suggestion within 3 seconds.
- System should handle upto 300 concurrent bookings
- Reservation updates should sync within 5 seconds

6. Design Constraints

- Must comply with data protection Laws.
- Cloud based support
- Both Android & IOS platform

7. Non-functional Req

- Reliability : 99.9% uptime
- Scalability : Support multiple properties
- Security : End-to-end encryption, secure auth
- Usability : Intuitive UI.
- Maintainability : Modular design to enable AI update modules.

8. Preliminary Schedule and Budget:

Schedule:

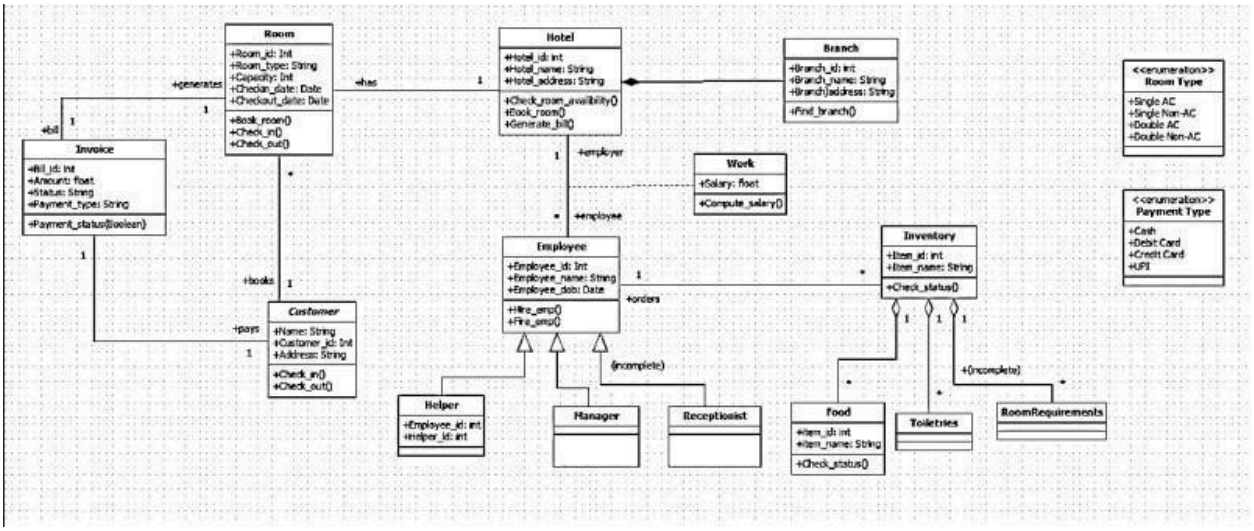
Month 1 : Req analysis + AI model selection
month 2 : System Design
month 3-4 Develop core modules
month 5 : Integrate with payment gateway
month 6 : App development
month 7 : Testing

Budget:

Development and AI-integration : $ 50,000
cloud Hosting : 12,000/ year
maintain + AI model : $ 7000/ year
Train & Documentati : : $4,000.

Total initial Budget $66,000

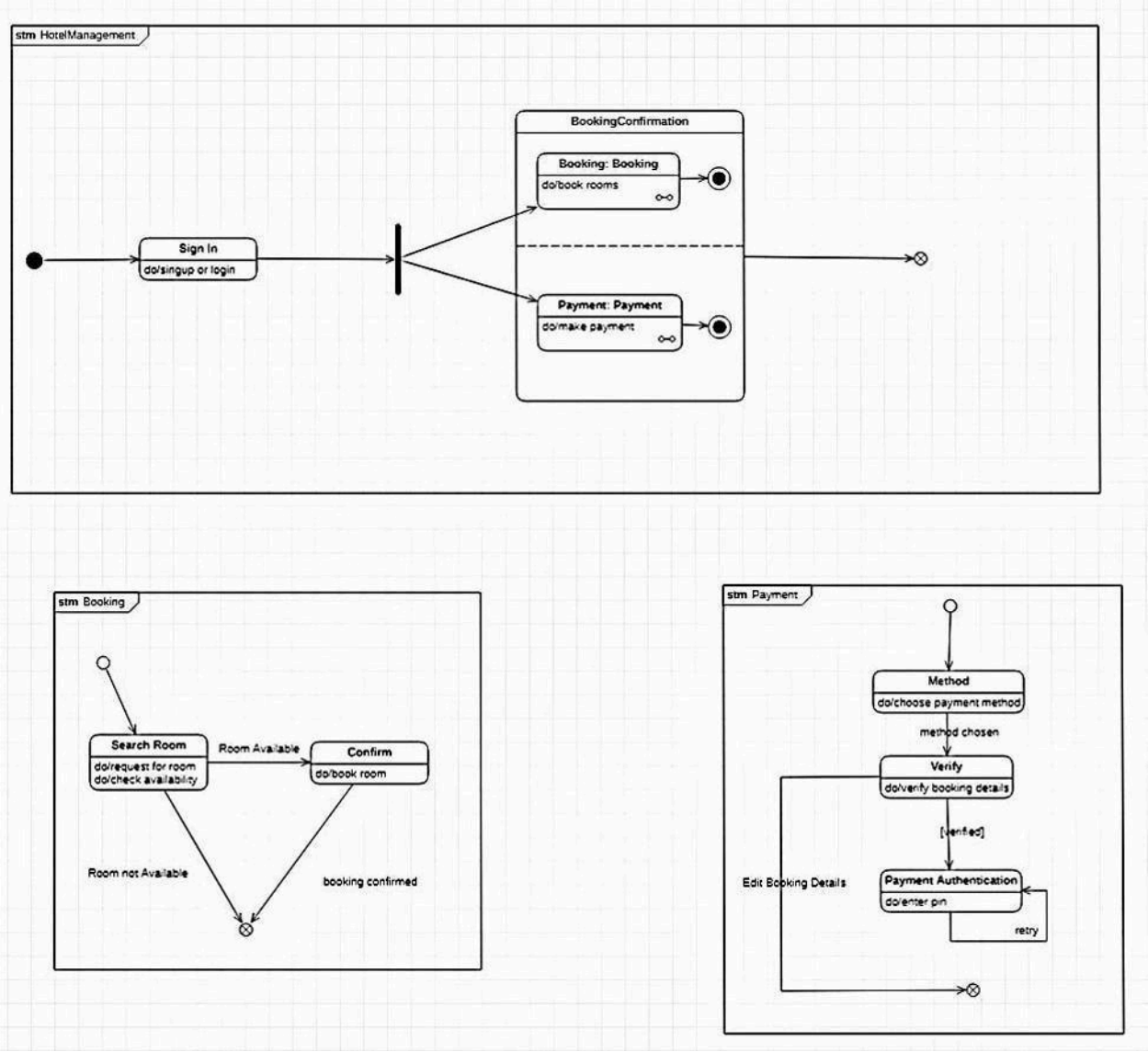
Class Diagram:



Explanation:

The class diagram for a Hotel Management System provides a detailed structure of the system by defining core classes—Hotel, Branch, Customer, Room, Employee, Payment, FoodItem, and Stock—alongside their attributes, methods, and intricate relationships. A hotel manages multiple branches, each with its own specifics, while customers can book various types of rooms, such as bedrooms or suites, and make secure payments. Employees are organized by roles like chef, manager, or receptionist, streamlining hotel operations, customer service, and food preparation. Additional components track food items and stock, ensuring an integrated approach to inventory and service management. Enumerations for room types and payment modes help standardize and automate bookings and transactions. By capturing both inheritance (e.g., specialized rooms and employee roles) and composition relationships, the diagram ensures a robust and adaptable design for managing hotel workflows efficiently.

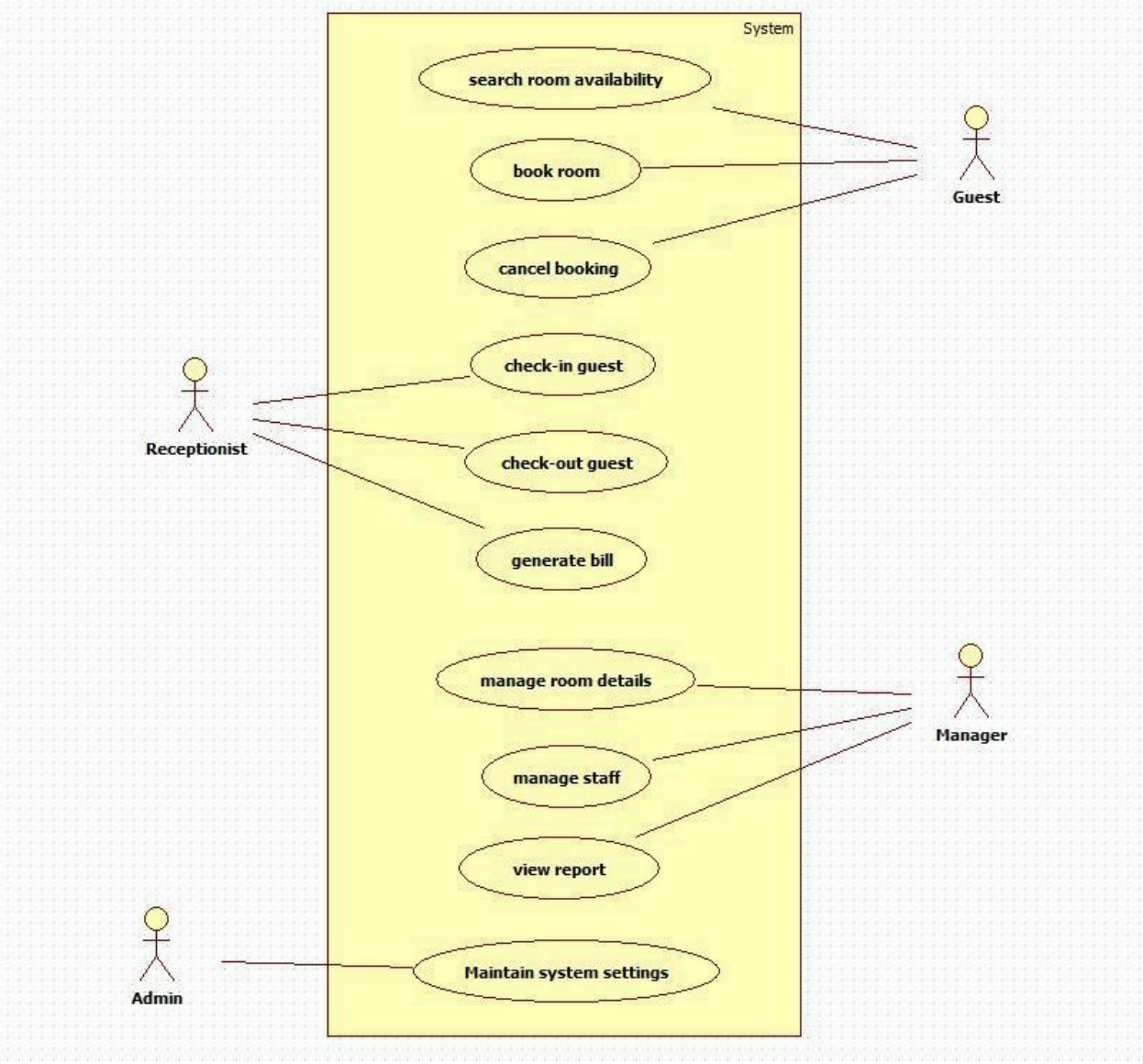
State Diagram



Explanation:

The advanced state diagram explains the entire hotel room management process. It begins with the system waiting for the user. When a room is booked, the system handles reservation details and then moves to the check-in process, where customer information is recorded and a room is assigned. During the guest's stay, the system enters the Room Occupied state, which includes billing and guest services like food delivery or housekeeping. After the guest checks out, the room goes through cleaning and, if needed, maintenance. Once completed, the room becomes available again. This diagram shows the full cycle of a room—from booking to check-in, stay, check-out, cleaning, and maintenance—providing a deeper view of hotel operations.

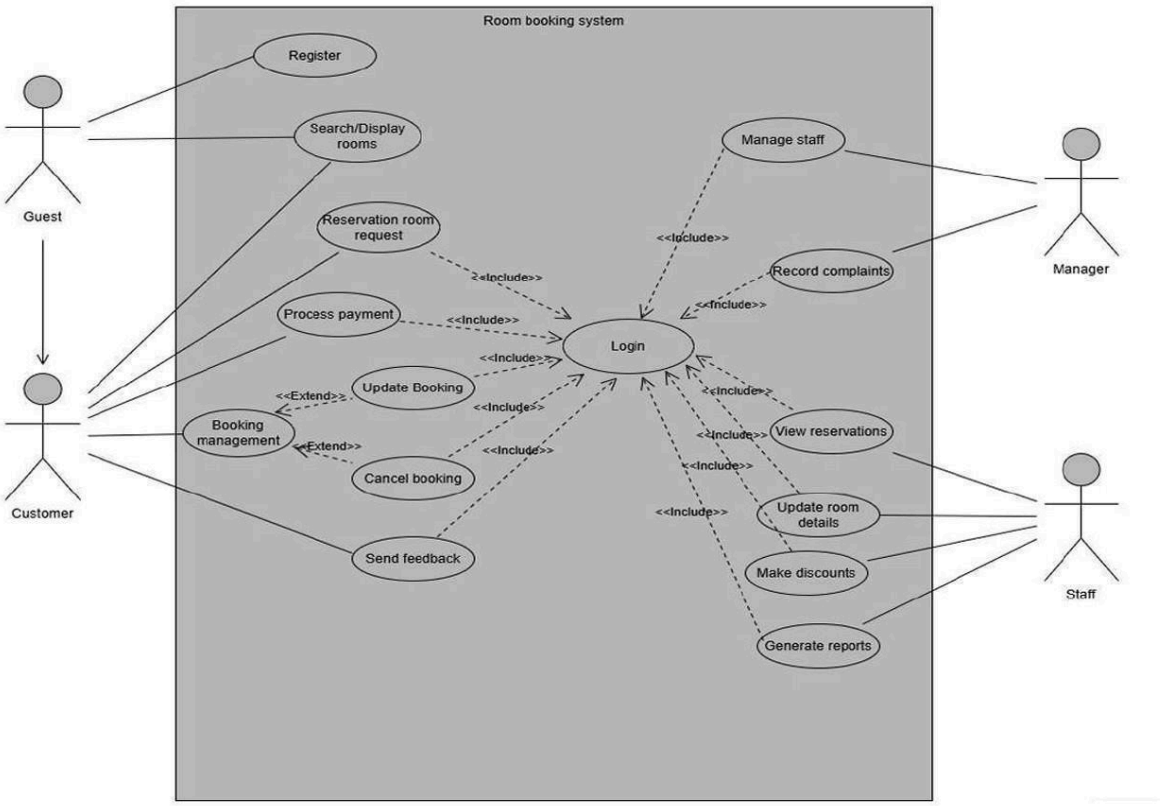
Use Case Diagram(Simple):



Explanation:

The simple use case diagram shows the basic everyday functions of the Hotel Management System. It involves four main actors: Customer, Receptionist, Manager, and the external Payment System. The Customer can perform common tasks like booking a room, checking in, ordering food, making payments, and checking out. The Receptionist handles bookings, assigns rooms, and prepares bills. The Manager has administrative responsibilities such as managing employees, handling rooms, and viewing reports. The Payment System helps process customer payments. Overall, this diagram focuses only on the essential hotel operations and is easy to understand because it highlights basic interactions between users and the system.

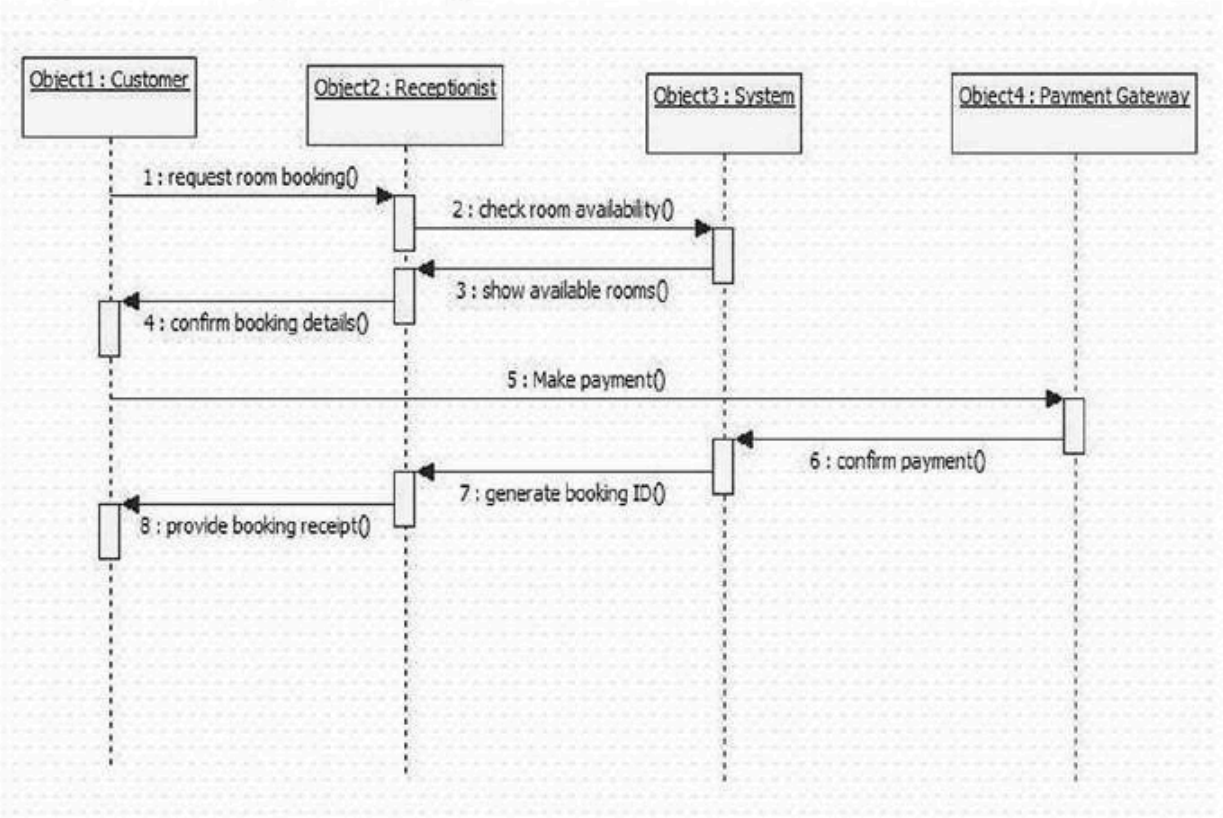
Use Case Diagram(Advanced):



Explanation:

The advanced use case diagram shows a more detailed view of how hotel staff and users interact with the system. It includes both Admin and User roles along with all supporting processes.The Admin can make reservations, view bookings, handle check-ins and check-outs, calculate bills, and process payments. These actions may involve additional steps such as choosing room types, selecting customers, filtering available rooms, or finalizing a reservation.The User can create customer profiles, check available rooms, view customer lists, and select reservations. These use cases support the admin's work and help the system run smoothly.This diagram gives a clearer picture of the system's internal workflow, showing how different tasks are connected and how the hotel manages the entire reservation and payment process.

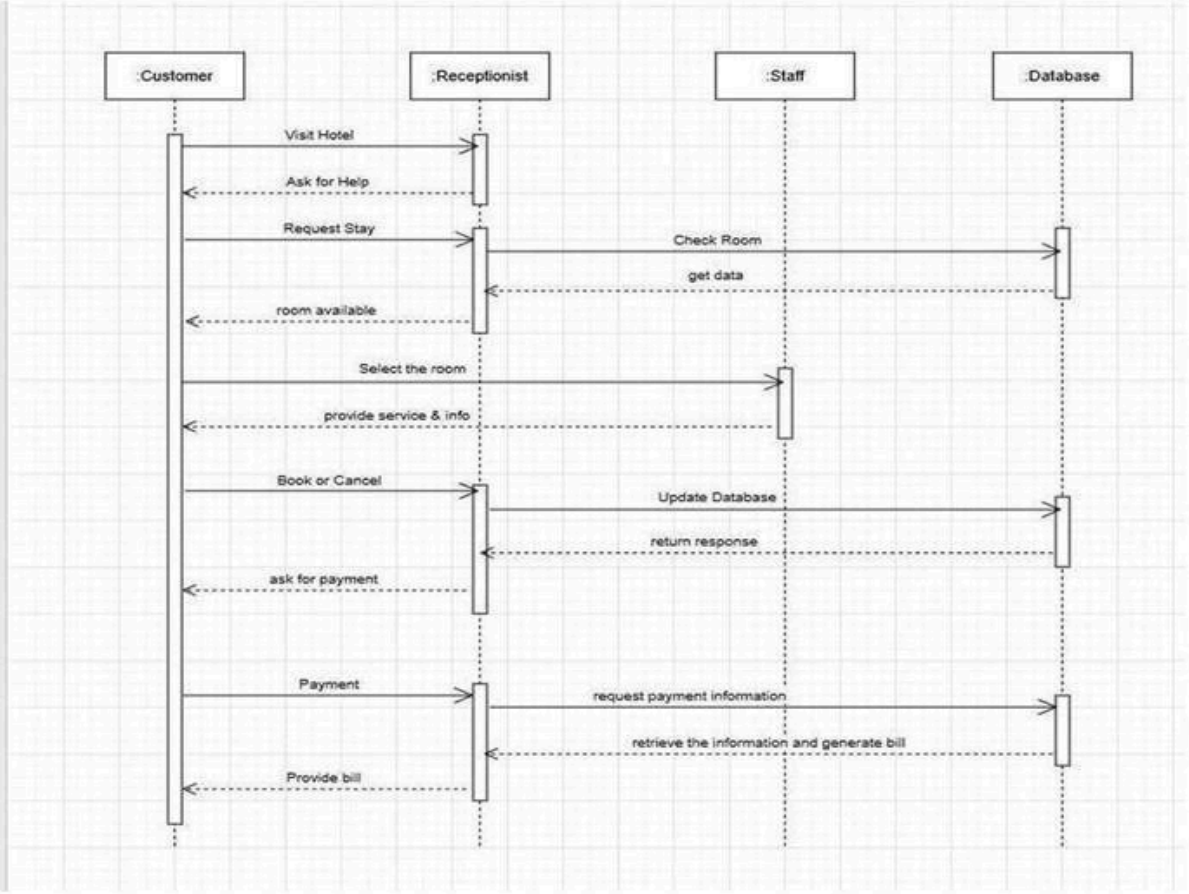
Sequence Diagram(Simple):



Explanation:

The simple sequence diagram explains the basic steps involved in booking a room. The process begins when the Customer asks the Receptionist to book a room. The receptionist checks availability by sending a request to the System, which returns a list of available rooms. After seeing the options, the customer confirms the booking.Next, the customer proceeds to make the payment. The receptionist sends the payment request to the Payment Gateway, which verifies the transaction and returns the result. If the payment is successful, the system creates a booking ID, and the receptionist gives the customer a booking receipt.This diagram focuses only on the key tasks—checking rooms, confirming booking, and processing payment—without showing extra steps or internal system details.

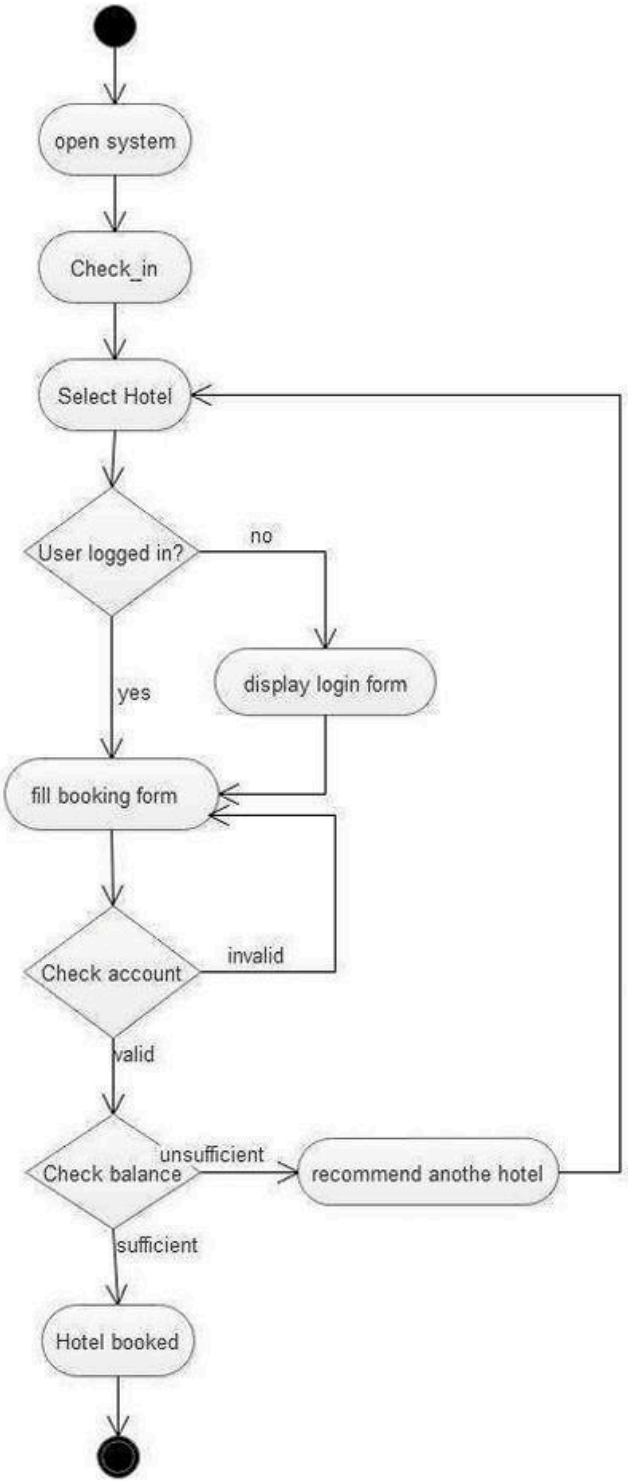
Sequence Diagram(Advanced):



Explanation:

The advanced sequence diagram provides a more complete picture of the hotel booking process. It starts when the Customer approaches the Receptionist for help. The customer requests a room, and the receptionist forwards this request to the Staff, who checks room availability by interacting with the Database. The database returns the room information, and the receptionist shares it with the customer. The customer selects a room, and the receptionist provides the required details or services. If the customer decides to book or cancel, the receptionist updates the Database, which stores the new booking status and returns confirmation. After the booking is confirmed, the receptionist asks the customer for payment. The receptionist retrieves payment details from the database and prepares the final bill for the customer. This advanced diagram shows the full workflow—from request to billing—highlighting the roles of staff and database operations behind the scenes.

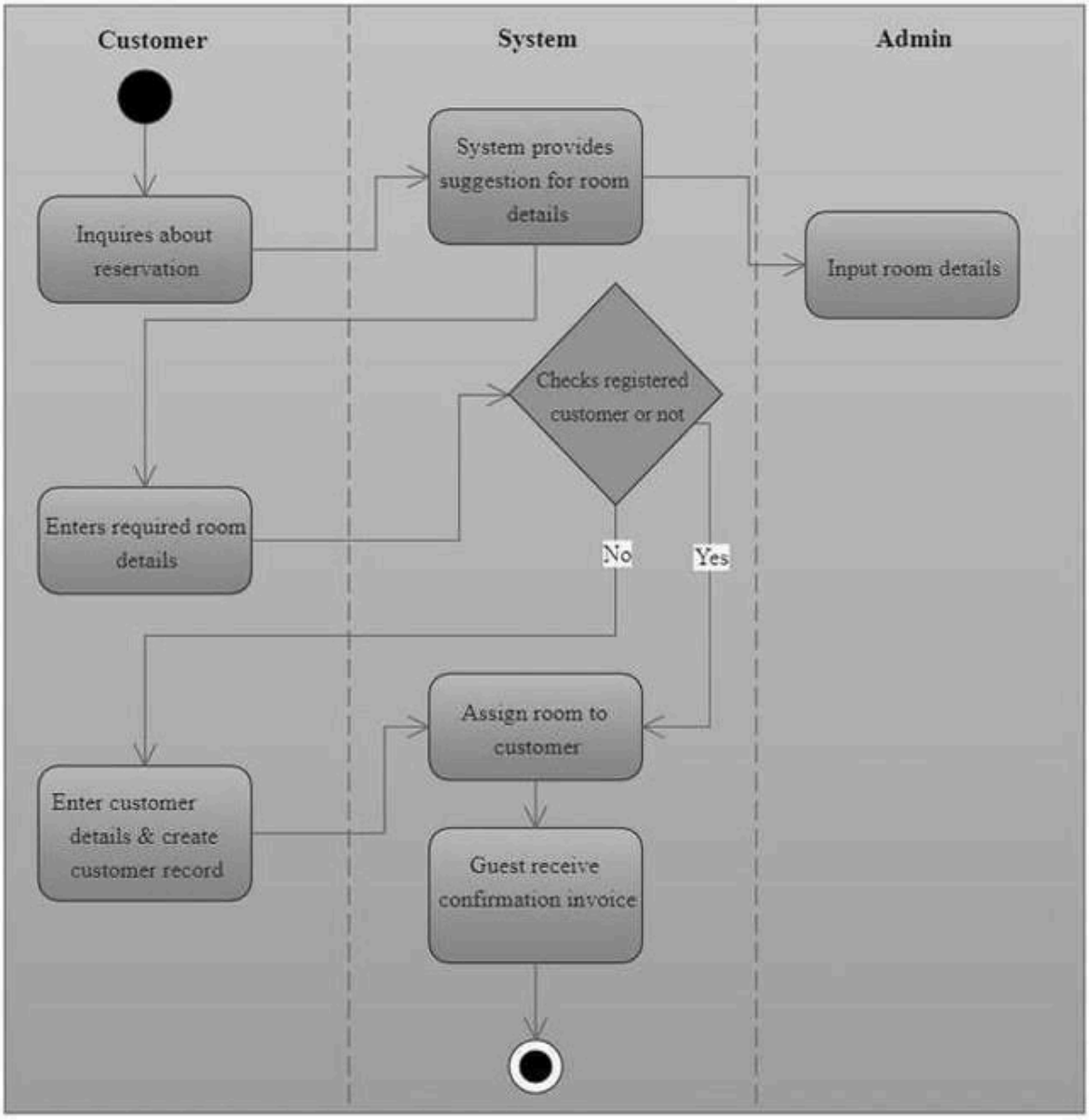
Activity Diagram(Simple):



Explanation:

The advanced sequence diagram provides a more complete picture of the hotel booking process. It starts when the Customer approaches the Receptionist for help. The customer requests a room, and the receptionist forwards this request to the Staff, who checks room availability by interacting with the Database. The database returns the room information, and the receptionist shares it with the customer. The customer selects a room, and the receptionist provides the required details or services. If the customer decides to book or cancel, the receptionist updates the Database, which stores the new booking status and returns confirmation. After the booking is confirmed, the receptionist asks the customer for payment. The receptionist retrieves payment details from the database and prepares the final bill for the customer. This advanced diagram shows the full workflow—from request to billing—highlighting the roles of staff and database operations behind the scenes.

Activity Diagram(Advanced):



Explanation:

The advanced activity diagram shows the same process in more detail and separates the actions into three swimlanes: Customer, Database, and Payment.

- In the Customer section, the user logs in, books a room, and makes the payment.
- In the Database section, the system checks if rooms are available and reserves the room when possible.
- In the Payment section, the system calculates the total amount, processes the payment, and generates the bill.

2. Credit Card Processing

SRS:

1. Introduction

Problem Statement :

A secure, real time fraud detection system is needed to identify and prevent unauthorized credit card transactions with high accuracy

1. Introduction :

1.1 Purpose

The purpose of the document is to define the requirements for developing a credit card fraud detection system that detects & prevents fraudulent transactions in real-time.

1.2 Scope :

The system will analyze incoming credit card transactions and classify them as legitimate or potentially fraudulent using ML and rule based detection methods.

Key features :

- Real-time fraud detection during transaction auth
- self-learning capability to adapt to fraud patterns
- Alerts customers & banks for suspicious activity

1.3 Overview

The FDS acts as middleware between the transaction processing engine and authorization system, uses historical transaction data, customer behavior patterns & external risk signal to determine fraud likelihood

2. [illegible] detection

- Continuous monitor of transaction in real-time
- Hybrid technique (rule based + AI/ML models)
- Suspicious transactions will be flagged for review or declined.
- Customer will be notified via SMS / Email / App notification.

3. Functional requirements.

- System shall analyze all credit card transactions before authorization.
- System calculates fraud risk score for each transaction
- System block / flag transactions above a configurable risk threshold.
- System shall notify customer if a transaction is blocked.
- System shall log all detected fraud attempts.
- System supports ML model updates without downtime.

4. Interface requirements

User Interface.

- Admin dashboard (fraud analyst view).
- Customer notification interface (mobile / web).

API Interface

- REST API for integration with payment gateway
- API to fetch transaction data for model retrain.

5. Performance requirements:

- System must process 10,000+ transaction per second. <100ms latency
- Fraud detection accuracy must be >95%.
- False positive should not exceed 2%
- The system must support 24/7 uptime.

6. Design Constraints:

must use end to end encryption (AES-256) for data transmission.

- support cross-platform integration

7. Non-function requirements

- Security: strong encryption, secure API.
- Scalability: scale horizontally to handle spikes in transaction volume
- Maintainability: Easy model retraining and rule updates.

8. Preliminary schedule and Budget:

Schedule:

month 1: Req. gathering and design.

month 2-3: Development of fraud detection engine.

month 4: API Integration.

month 5: Testing

month 6: Deployment

Budget :

Dev team : $120,000

Cloud infra : $20,000

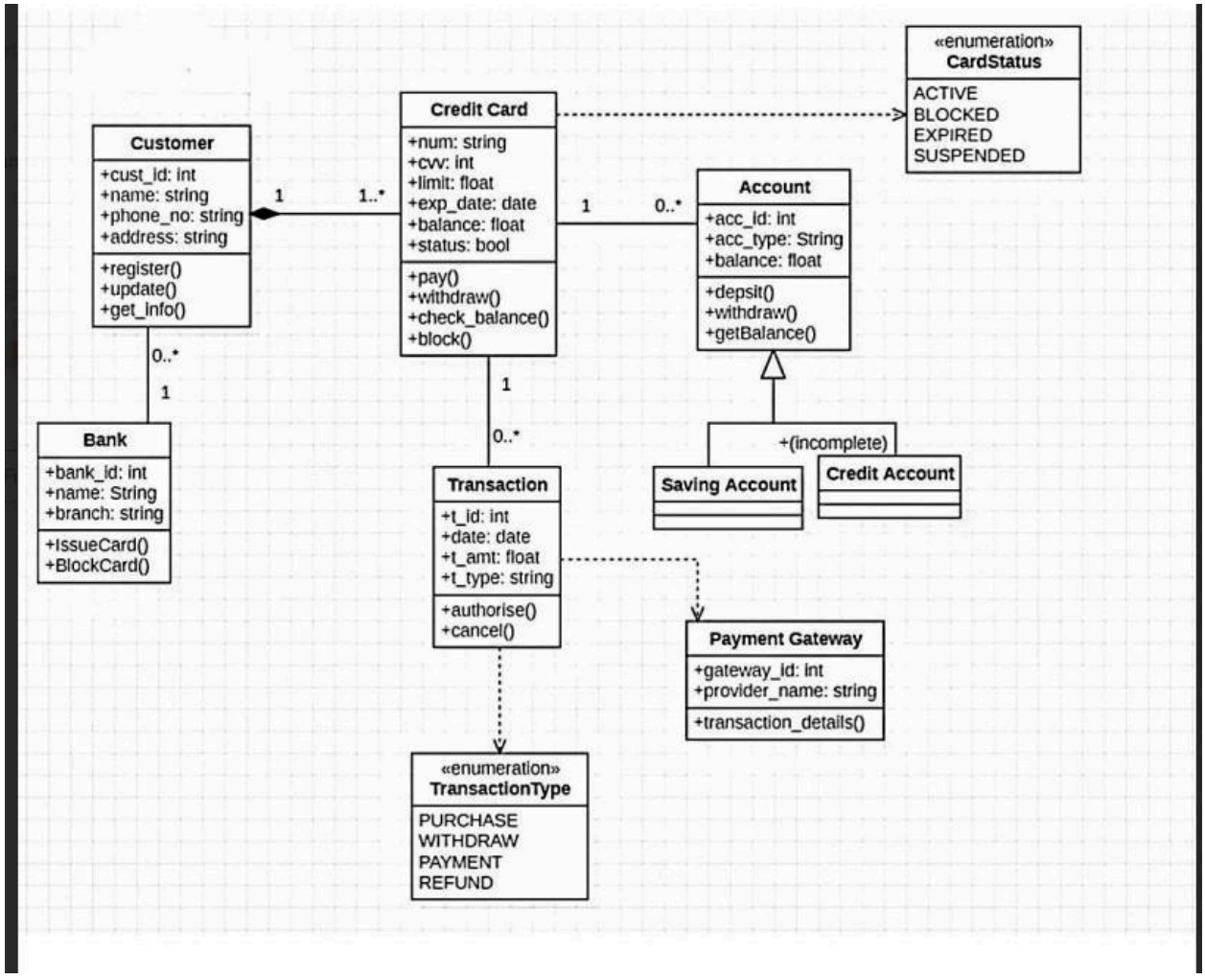
Licensing + Compliance : $10,000

Testing + QA : $15,000

Contingency : $10,000

Total : $175,000

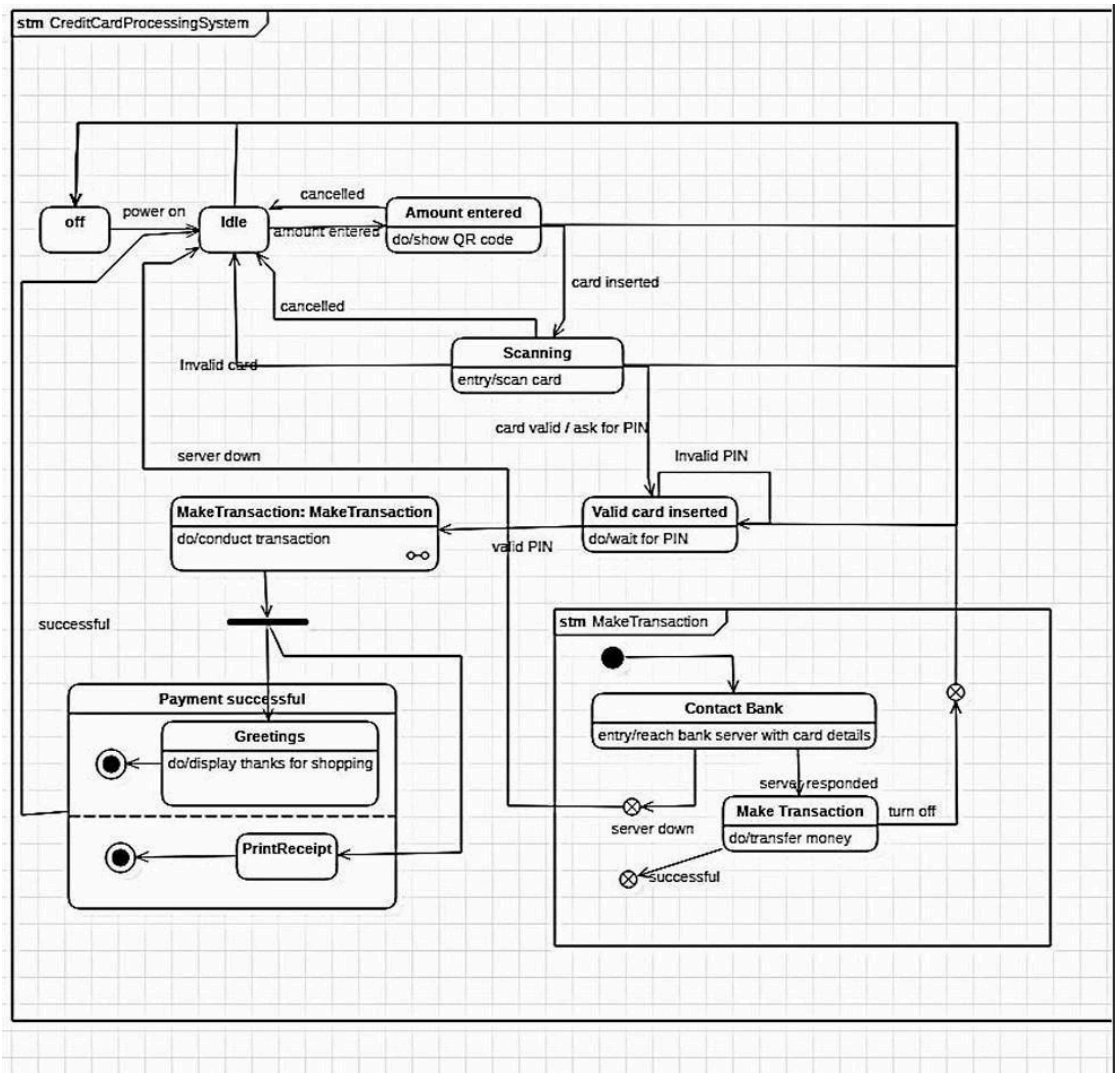
Class Diagram:



Explanation

The class diagram represents an online payment processing system that manages customers, merchants, banks, and transactions. The Customer initiates payments using their card, which is linked to a Bank Account. A Transaction is created for each payment, containing details such as amount, date, status, and authorization code. The Bank authorizes and settles payments, while the PaymentGateway facilitates communication between the customer's bank and the merchant. The Merchant receives payments, issues refunds, and handles chargebacks. Additional classes such as Authorization and Refund manage fraud checks, verification, and money returns. Supporting entities like Person, Payment, and the status enumeration define basic attributes and types used throughout the system. Overall, the diagram shows how different components work together to process, validate, and complete electronic payment transactions securely.

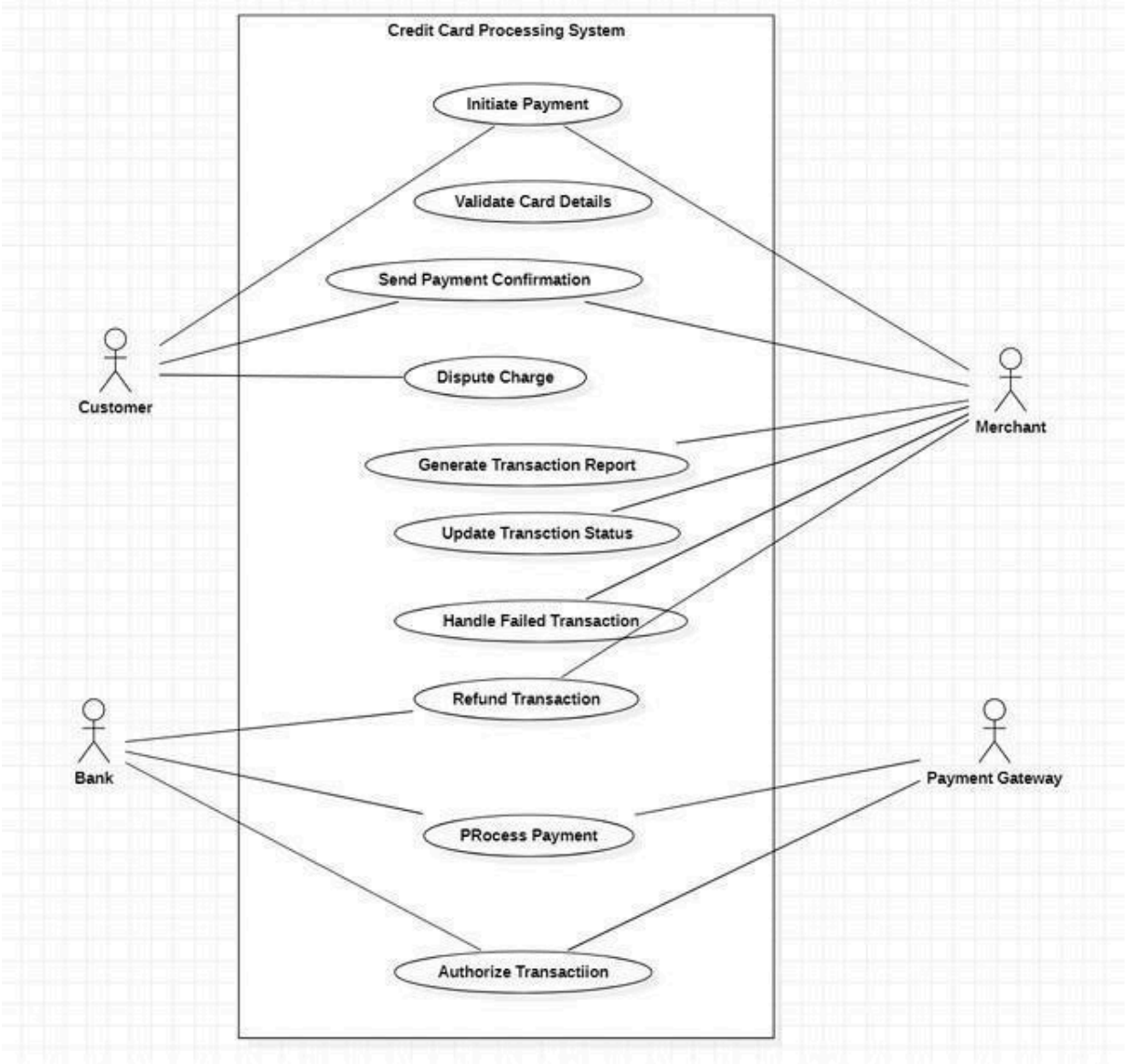
State Diagram:



Explanation:

The advanced state diagram describes a more detailed and realistic payment or ATM terminal process. It begins in an Idle state, where the terminal waits for input. A user either enters an amount (for QR payment) or inserts a card. The system validates the card through a scanning process. If the card is valid, the terminal asks for the PIN and moves to the Valid Card Inserted state. After the correct PIN is entered, the process moves to Make Transaction, where the transaction is conducted and sent to the bank for approval. The Control Bank state checks card details and verifies the transaction. If the bank responds successfully, the terminal processes the payment and shows a greeting message, followed by printing a receipt. The diagram also includes multiple exceptional paths such as invalid PIN, invalid card, server down, or cancelled actions. This advanced diagram shows a complete, realistic workflow including scanning, PIN verification, bank communication, success/failure handling, and receipt printing.

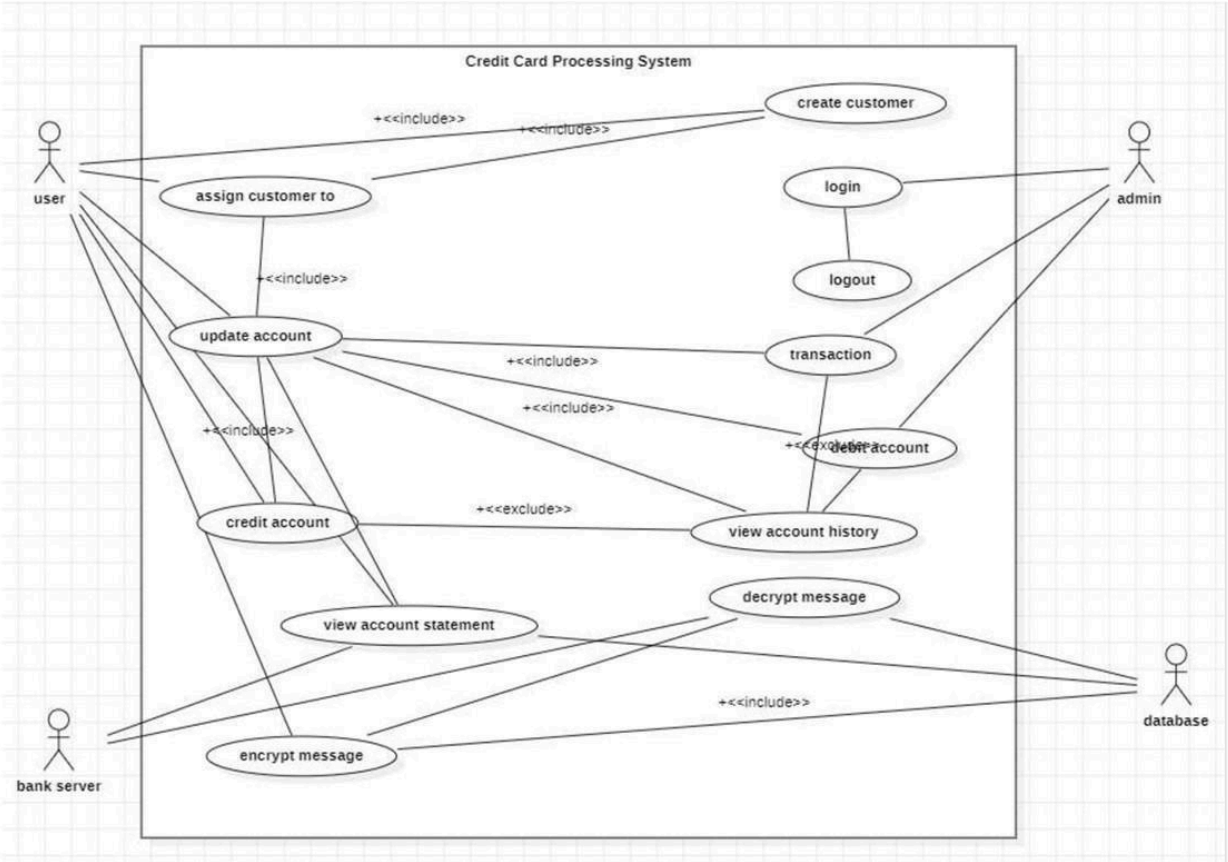
Use Case Diagram(Simple):



Explanation:

The simple use case diagram focuses on the main functions involved in credit card processing between a merchant and the banks. The primary actor is the Merchant's Credit Card Processing System, which communicates with both the Merchant's Bank and the Customer's Credit Card Bank. The system performs core actions such as Authorizing, Capturing, Crediting, Voiding, and Verifying transactions. These actions represent the essential steps required to approve a payment, capture funds, issue credit, reverse a payment, or validate card details. Since the diagram only shows the main operations without detailing internal workflows, it provides an easy-to-understand overview of how basic credit card transactions are handled.

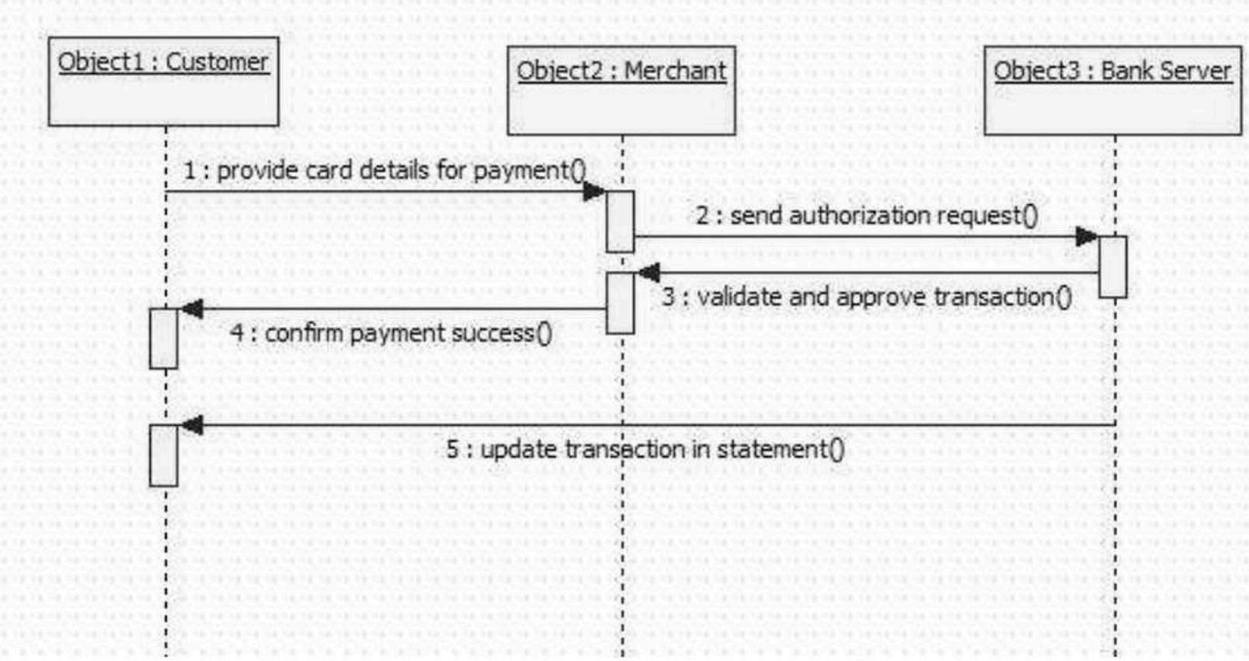
Use Case Diagram(Advanced):



Explanation:

The advanced use case diagram presents a much more detailed view of how users and administrators interact with a banking system. Multiple actors participate: the Admin, Customer (User), Bank Server, and Database. The system supports a wider set of processes such as Creating Customer, Assigning Customer ID, Updating Account, Crediting and Debiting Accounts, Viewing Statements and Account History, Transactions, Login/Logout, and secure processes like Encrypting and Decrypting Messages. Many relationships use *include* and *extend*, showing how smaller actions support larger processes. This diagram provides a complete picture of how customer accounts are managed, how transactions flow through the system, and how security is maintained. It shows internal system behavior more clearly than the simple diagram.

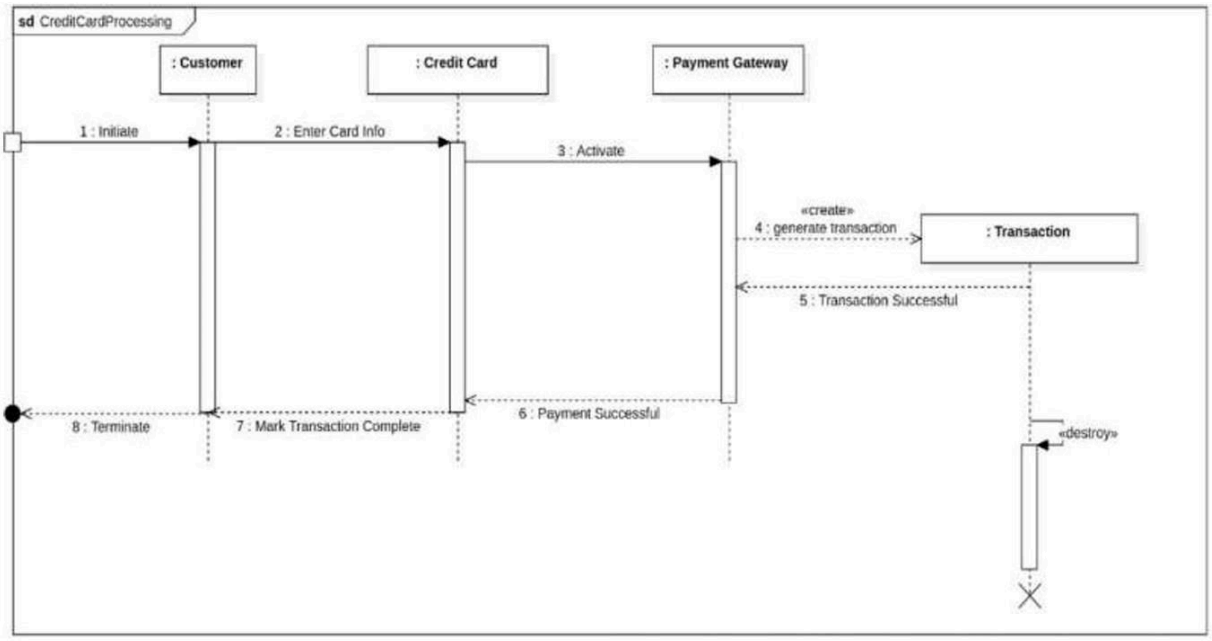
Sequence Diagram(Simple):



Explanation:

The simple sequence diagram shows the basic steps involved when a customer makes a payment using a credit card. The process starts when the Customer enters card information, which is then sent to the Credit Card system. The credit card activates the request and sends it to the Payment Gateway. The payment gateway generates the transaction and forwards it to the internal processing object. Once the transaction is approved, a Transaction Successful message is sent back through the payment gateway to the credit card system. The credit card then informs the customer that the payment is successful, and the transaction is marked as complete. This diagram only focuses on the essential steps: entering card data, verifying payment, and completing the transaction.

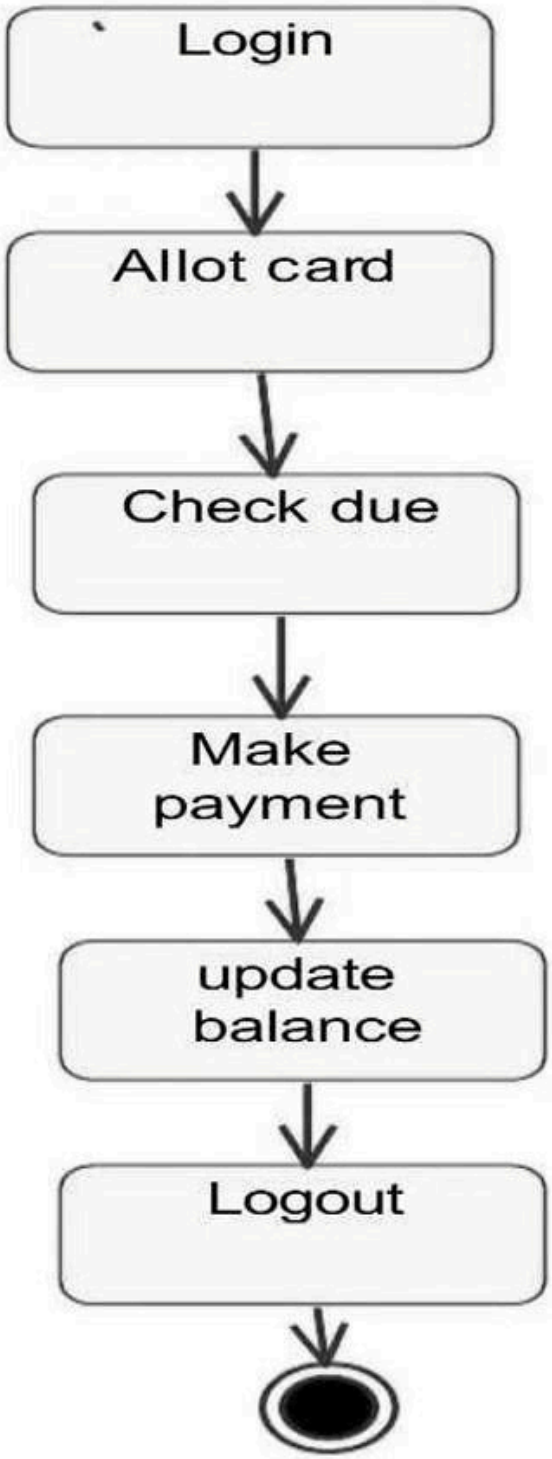
Sequence Diagram(Advanced):



Explanation:

The advanced sequence diagram provides a more detailed view of how a credit card payment is handled behind the scenes. It begins when the Customer provides card details to the Merchant for payment. The merchant then sends an authorization request to the Bank Server. The bank server validates the card information, checks the account status, and approves or denies the transaction. If approved, the bank sends back a confirmation to the merchant, who then informs the customer that the payment was successful. Finally, the merchant updates the customer's statement or transaction history. This advanced diagram shows a more complete workflow, including authorization, validation, approval, and transaction recording.

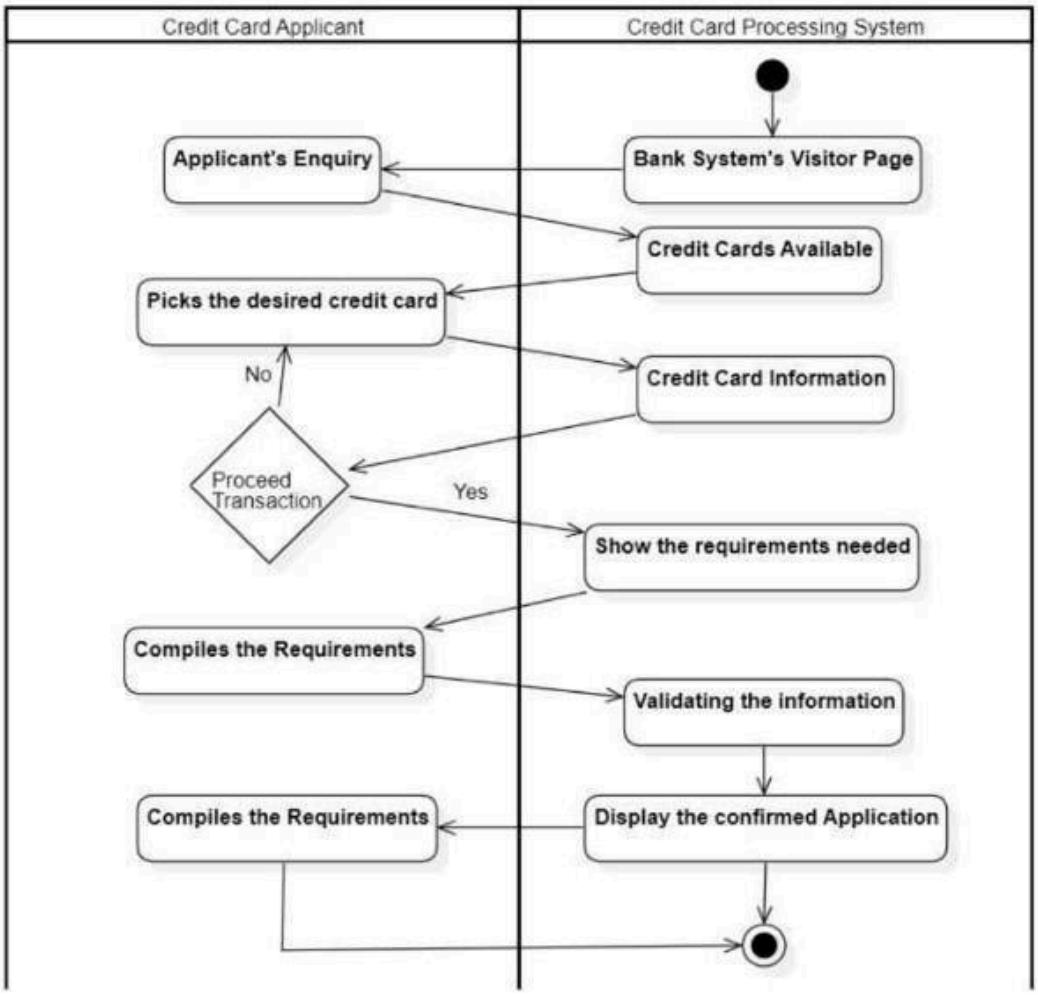
Activity Diagram(Simple):



Explanation:

The simple activity diagram explains the basic steps in completing a credit card payment. The process starts when the customer enters card details and submits a payment request. The system then validates the details. If the card information is invalid, the system immediately displays a payment failure message. If the details are valid, the request is sent to the issuer bank, where the system checks the customer's available credit. If the bank does not authorize the payment, failure is shown again. If everything is correct, the bank authorizes the transaction, and the amount is transferred to the merchant. Finally, the user is shown a payment successful message. This diagram focuses only on the main steps—entering details, validating, checking credit, and approving or rejecting the payment.

Activity Diagram(Advanced):



Explanation:

The advanced activity diagram presents a more detailed workflow of how a credit card application is handled. It begins when an applicant makes an inquiry about credit cards. The credit card processing system responds by displaying available card options. The applicant then selects a card and decides whether to proceed. If they continue, the system shows the required documents and information needed for application. The applicant gathers and submits these requirements. The system then validates the submitted information, checking eligibility criteria. Once everything is confirmed, the system displays the approved or completed application. This diagram covers more steps and interactions compared to the simple one and offers a clearer view of how credit card applications are processed from inquiry to validation.

3. Library Management System

SRS:

Problem statement

Manual library operations cause errors in tracking books, borrowing and returns, An automated LMS is needed to ensure effective book management.

1. Introduction :

1.1 Purpose

purpose of this document is to specify requirements for a library management system that automates cataloging, borrowing and returning of books.

1.2 Scope :

The system will allow librarians to manage book inventories and users to search, borrow & return books efficiently. Handle late fee calculations automatically.

1.3 overview :

System provides

- Book cataloging and search
- Borrow / return
- Late fee calculation
- User & librarian access portal.

2. General Description.

System replaces manual record-keeping with digital database. Users can view available books, borrow them & track due dates.

3. Functional requirements.

- System shall allow users to search books
- System shall allow borrowing and returning of books
- System shall calculate late fee automatically
- System shall maintain user & book records

4. Performance Interface requirement:

- Must support atleast 1000 users concurrently
- ~~Database should be light weight~~.
- Search results must be displayed in <2 seconds.

5. Interface ~~Performance~~ requirements.

- User Interface - Simple web/app interface for students / librarians.
- External Interface : Printer support for receipts.

6. Design constraints:

- must run on standard pc's with internet access.
- Database should be lightweight

7. Non-functional requirements.

- Security : User login authentication required
- Usability : Intuitive interface for non-technical users

Reliability : 99% uptime for operations.

8. Preliminary Schedule + Budget.

Schedule :

month 1: Requirement + design

month 2: Development

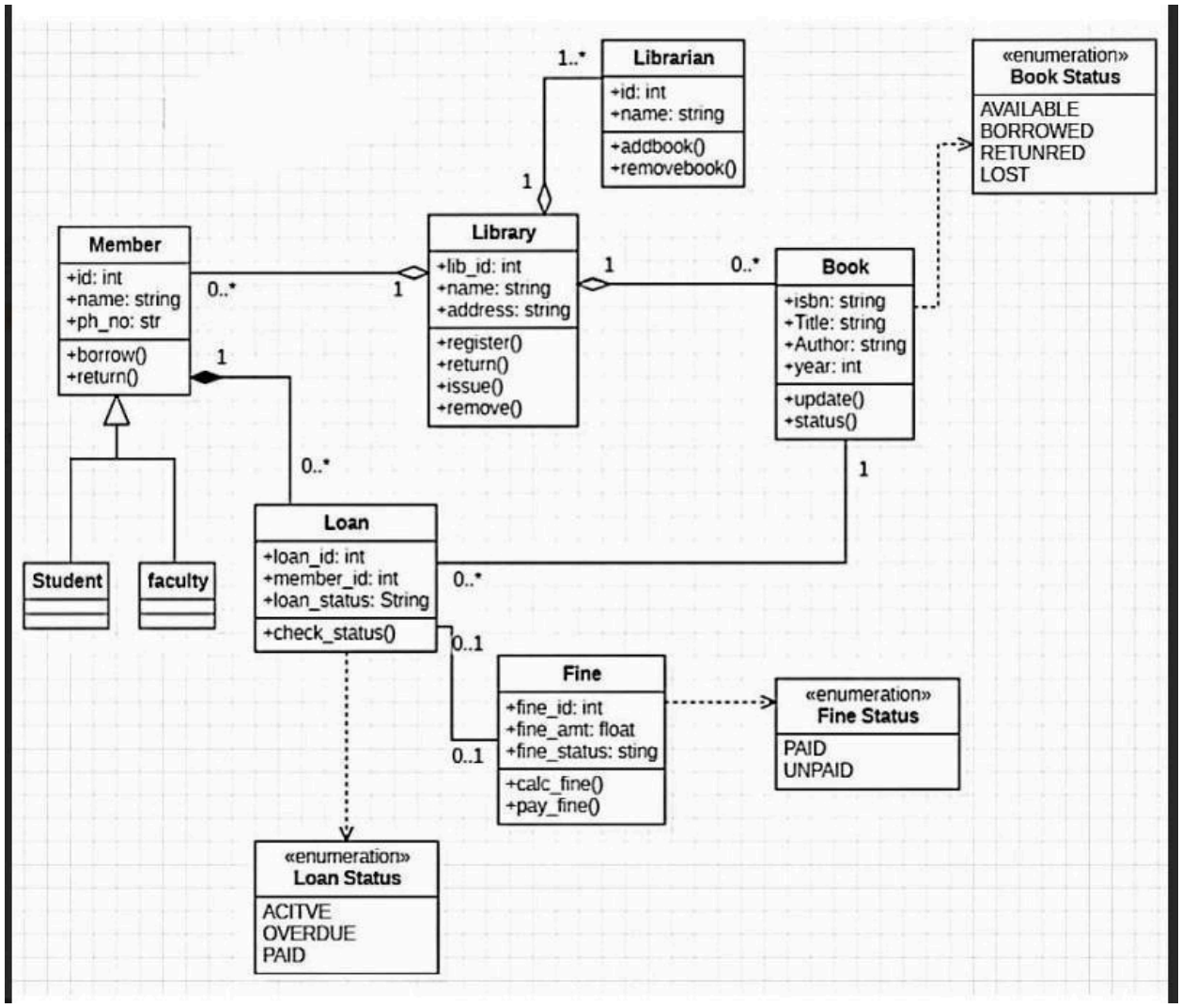
month 3: Testing + deployment.

Budget :

Development + setup : $5,000

Maintenance : $1,000/ year.

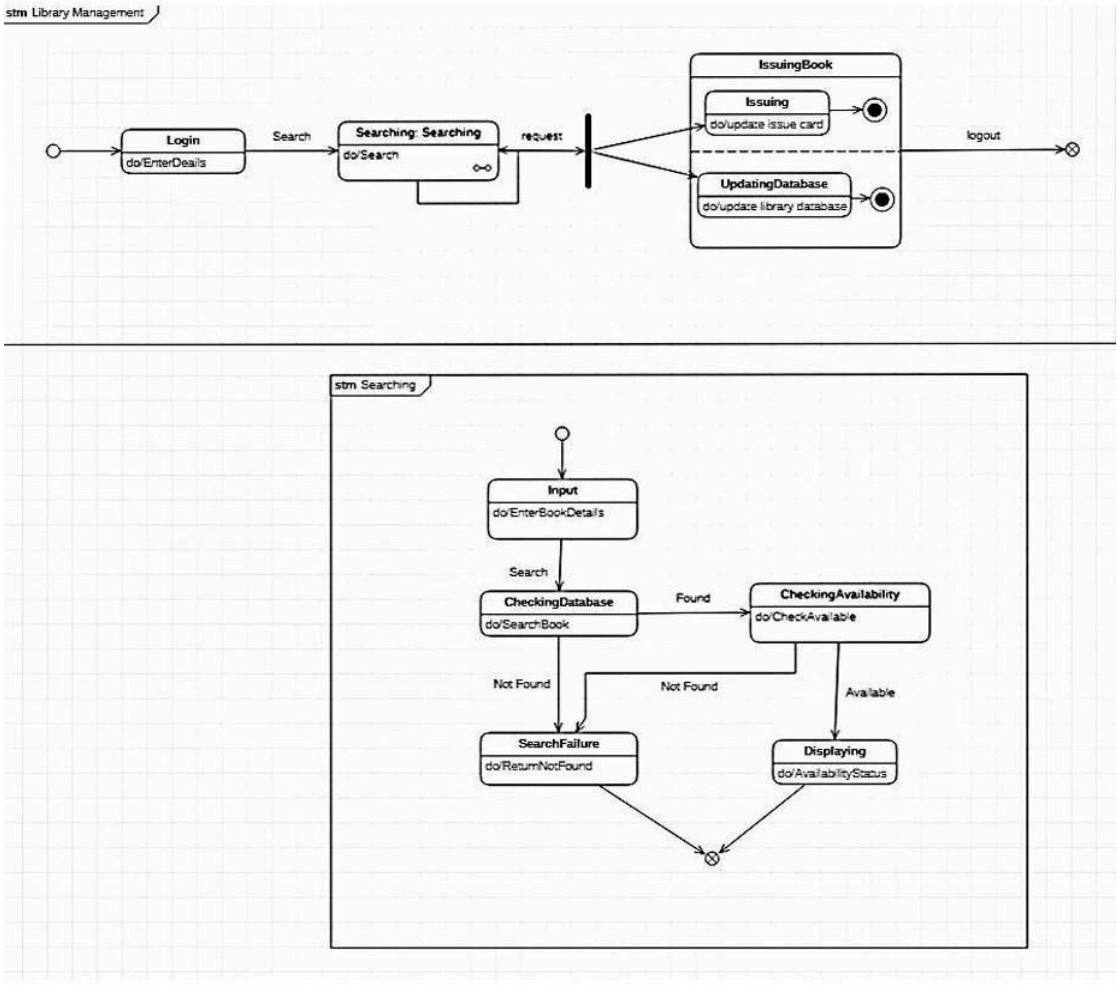
Class Diagram:



Explanation:

The class diagram represents a Library Management System where books, members, librarians, and transactions interact to manage library operations. The Book class stores details such as author, edition, price, and purchase date, and it is specialized into different types like Journals, Magazines, and Subject Text through inheritance. Members of the library—categorized as Students and Faculty—send book requests and borrow books. The Librarian plays a central role by searching books, verifying members, issuing books, calculating fines, and handling book returns. The librarian also updates the status of books and ensures that library policies like issuing limits are followed.

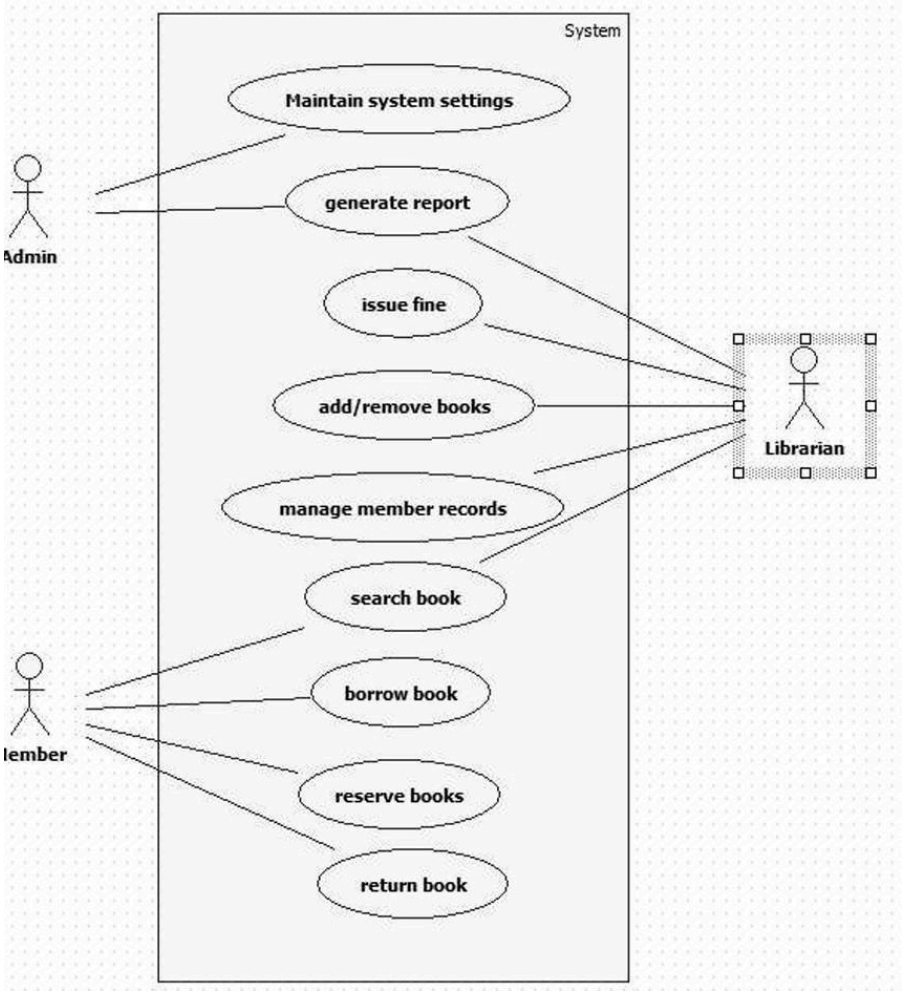
State Diagram:



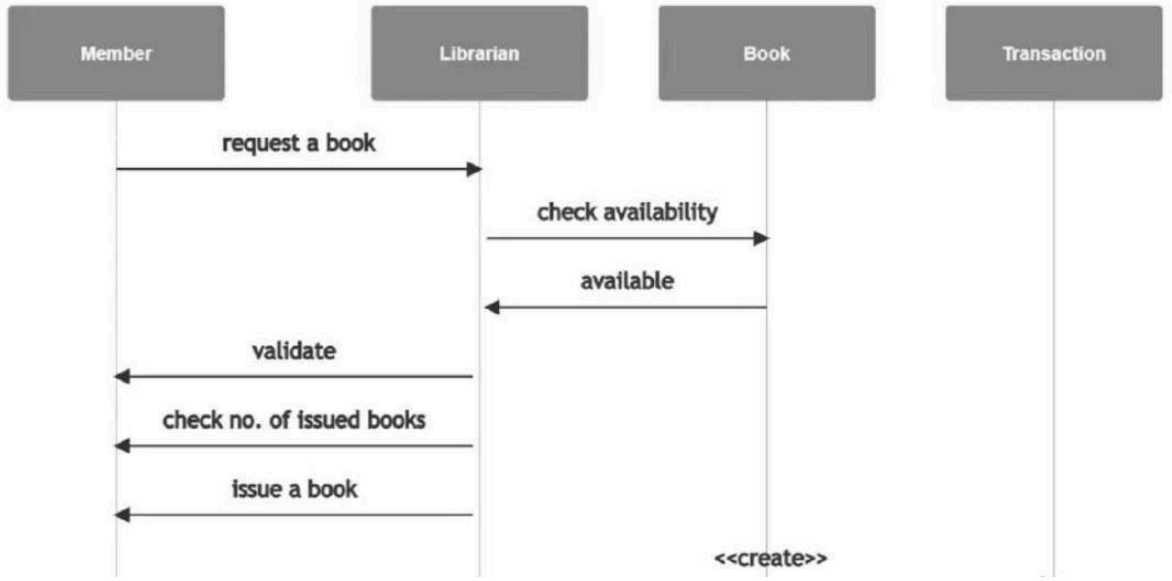
Explanation:

The advanced state diagram provides a more complete picture of the library's book issuing and returning process. It starts with the user searching for a book. If the book is available, the system verifies stock and moves to the issue state, where member and library records are updated. If the book is unavailable, the system places the user in a waitlist and notifies them when the book becomes available. Once issued, the system moves into the Borrowed state, where the due date is monitored. From here, the user may renew the book or return it. Upon returning, the system checks for damage or late return and computes fines if applicable. If a fine exists, the user proceeds to Fine Payment; otherwise, the process ends. This diagram captures the full lifecycle of a borrowed book—from search and waitlist to issuing, borrowing, returning, renewal, and fine

Use case diagram(Simple):

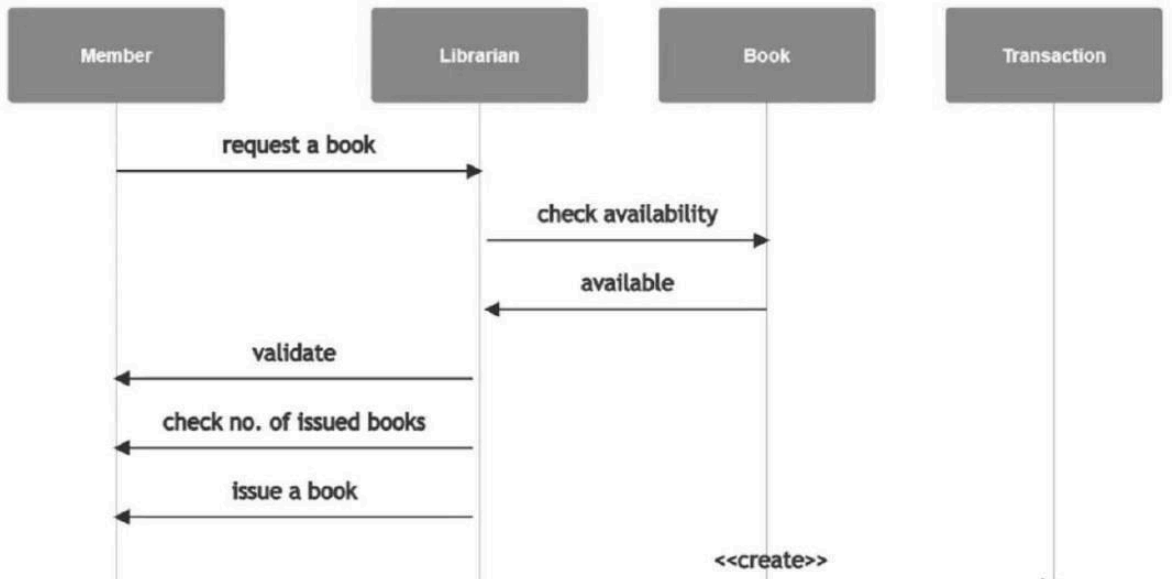


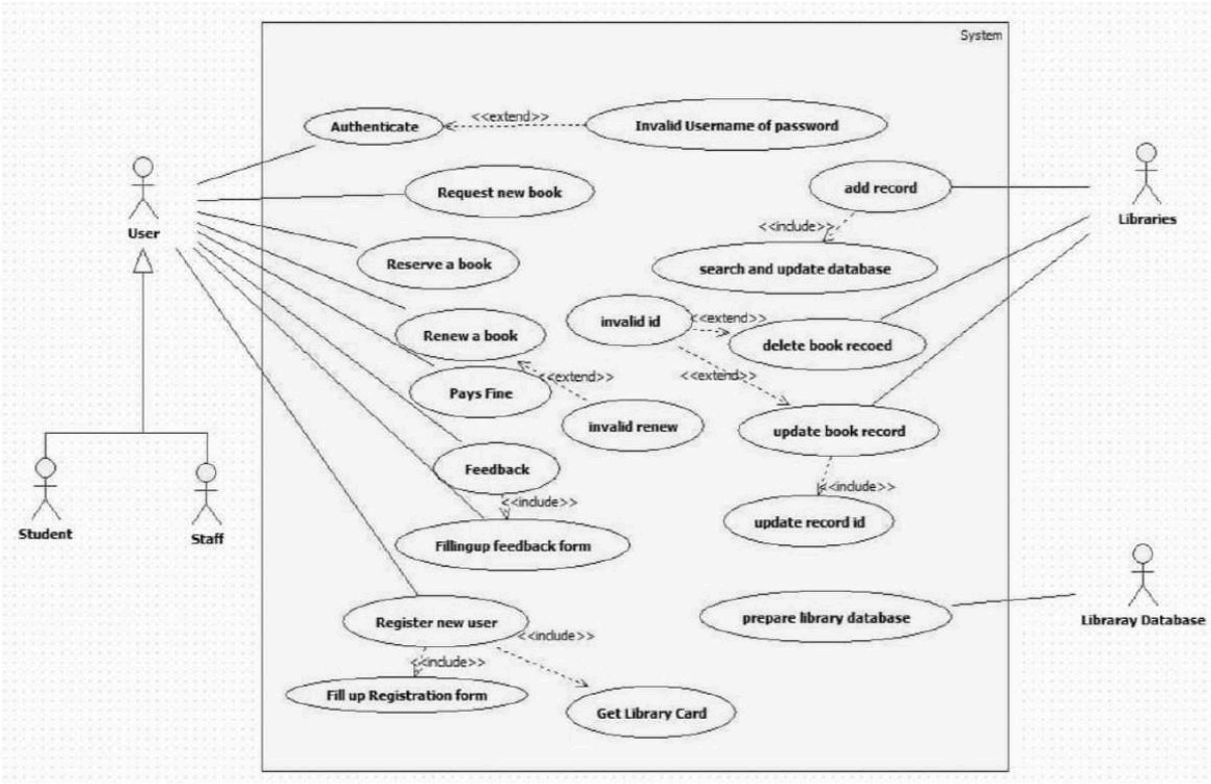
Explanation:



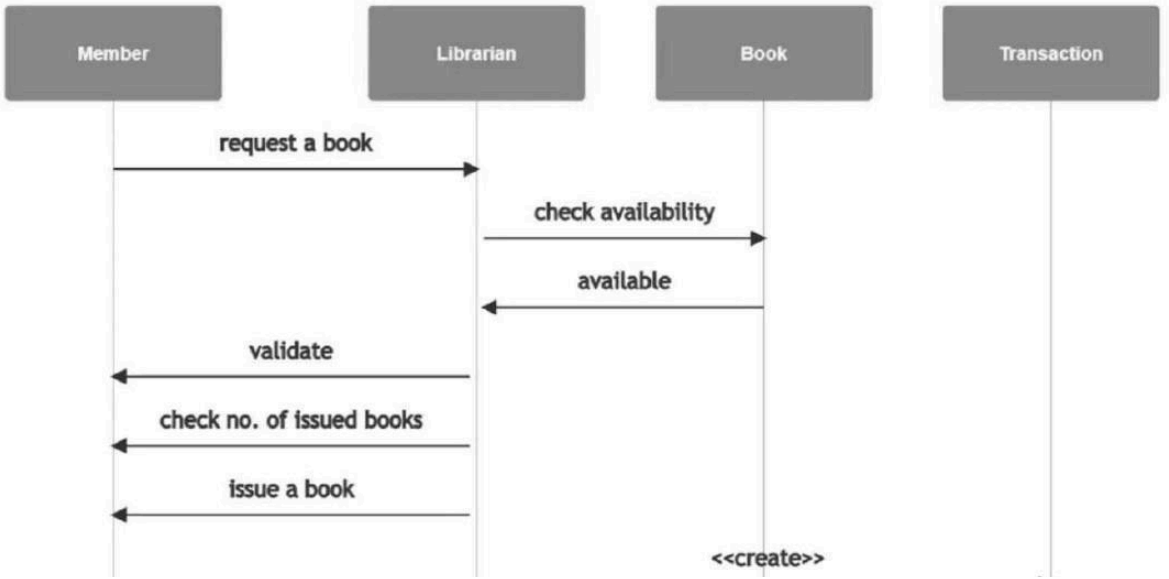
Use Case Diagram(Advanced):

Explanation:





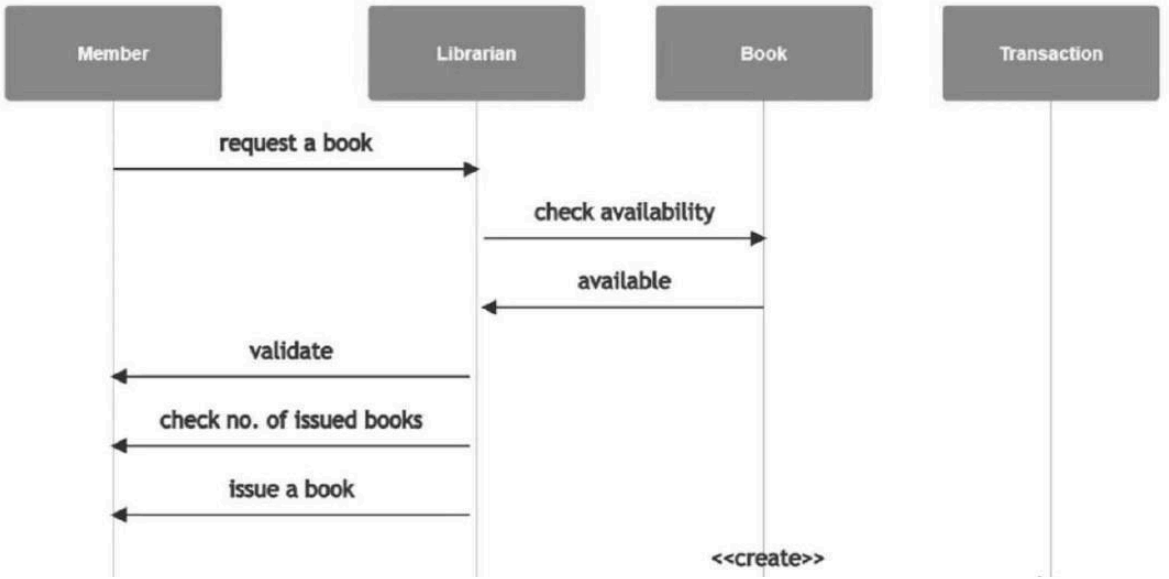
Explanation:



The advanced use case diagram presents a more detailed and comprehensive view of the library system by involving multiple actors—User (Student/Staff), Libraries, and the Library Database. It includes a wider range of use cases such as Authentication, Requesting New Books, Reserving Books, Renewing Books, Paying Fines, Providing Feedback, Registering New Users, and Filling Forms. The system also manages book records using operations like Adding, Updating, and Deleting Records, as well as preparing the library database. Several use cases use *include* and *extend* relationships to show dependencies and optional behaviors such as invalid login, invalid ID, or invalid renewal. This advanced diagram gives a full picture of how users interact with the system, how library data is maintained, and how operations extend beyond simple issuing and returning of books.

Sequence Diagram(Simple):

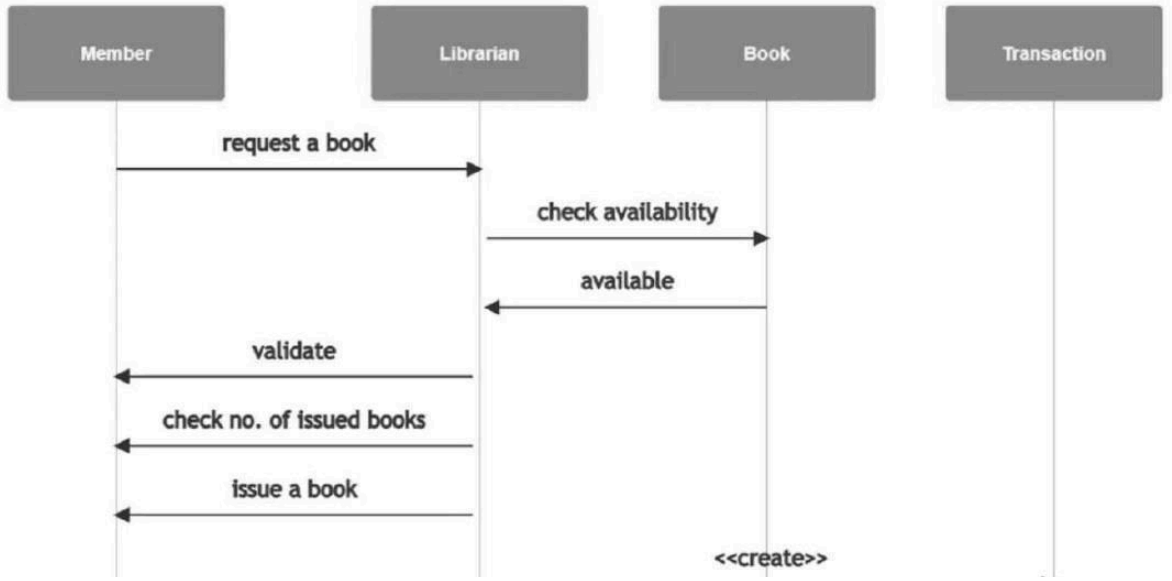
Explanation:

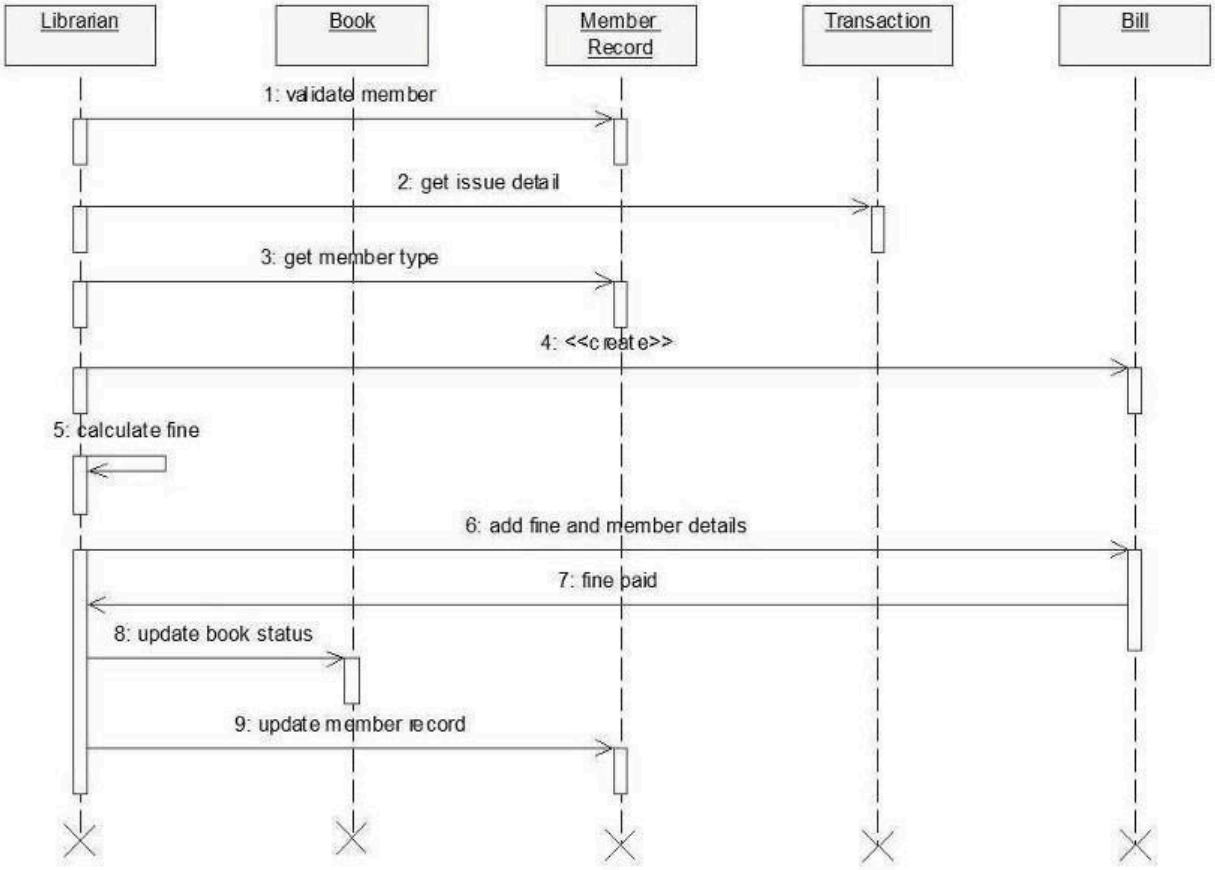


The simple sequence diagram shows the basic process of borrowing a book in the library. The interaction begins when the Member requests to borrow a book. The Librarian receives this request and sends a command to the System to search for the book. The system returns the book details, after which the librarian proceeds to issue the book to the member. Once the book is issued, the librarian updates the library database, and finally the member receives the issued book. This diagram focuses only on the essential steps—requesting, searching, issuing, and updating—making it easy to understand how a book is borrowed in a simple workflow.

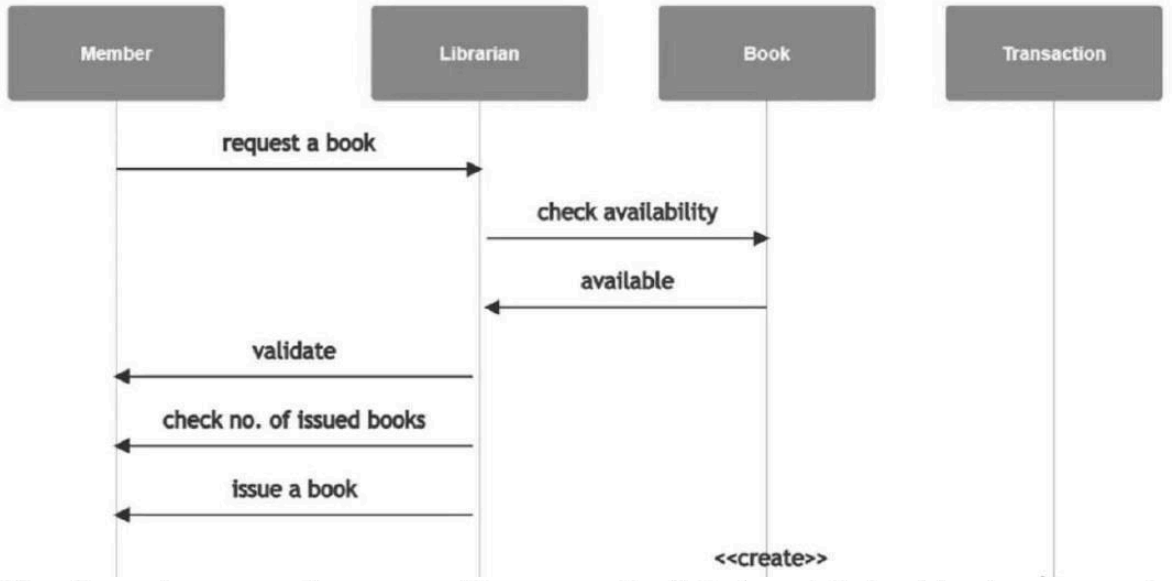
Sequence Diagram(Advanced):

Explanation:





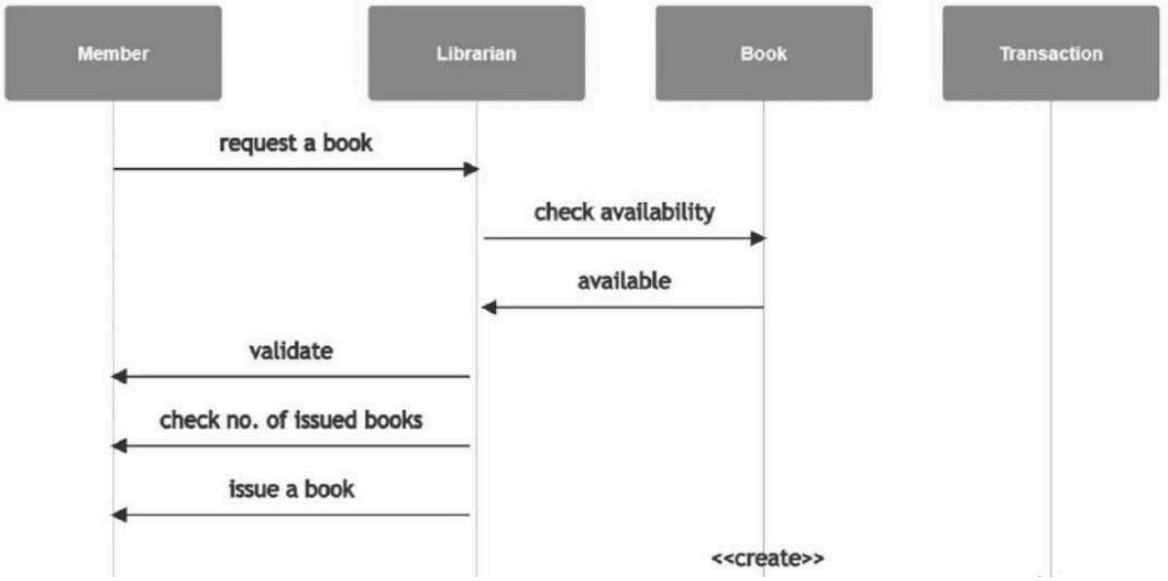
Explanation:

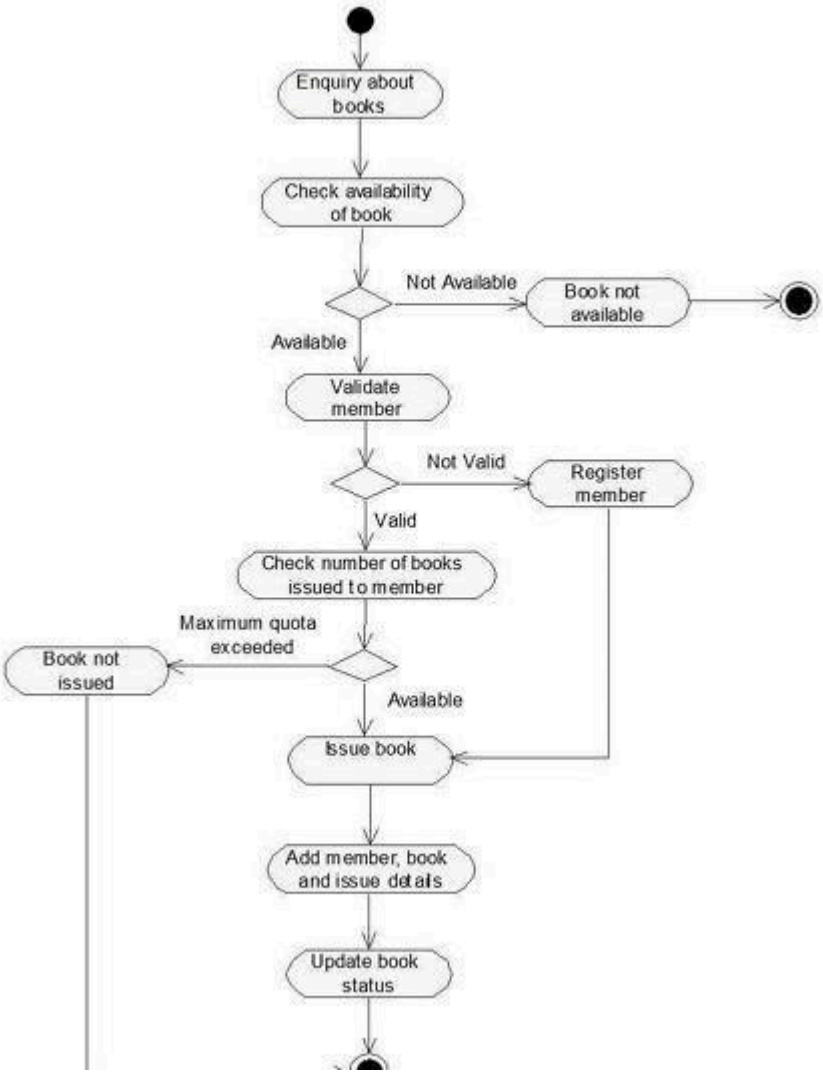


The advanced sequence diagram provides a more detailed view of the book-issuing process. It starts when the Librarian checks the availability of a selected book. The Book object responds with availability status. The librarian then validates the member and checks how many books the member has already borrowed. If issuing is allowed, a Transaction record is created, containing details of the member and the book. The librarian then updates the book status, marking it as issued, and updates the member's record to reflect the new loan. This detailed diagram shows all internal checks—availability, member validation, issuing limits—and the creation of proper records, giving a complete view of how the system handles book issuing behind the scenes.

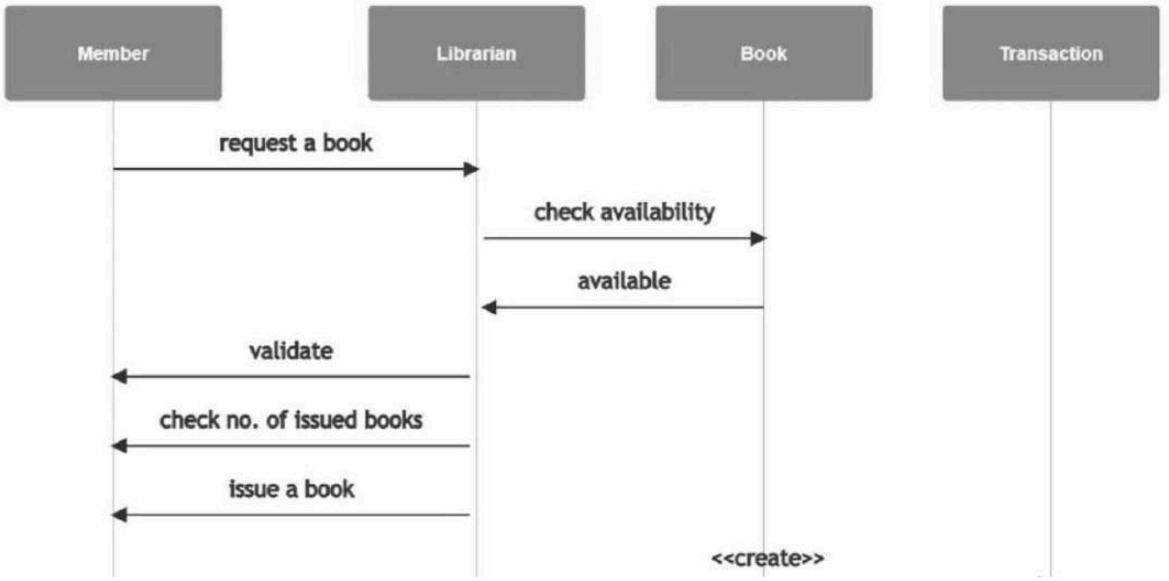
Activity Diagram(Simple):

Explanation:



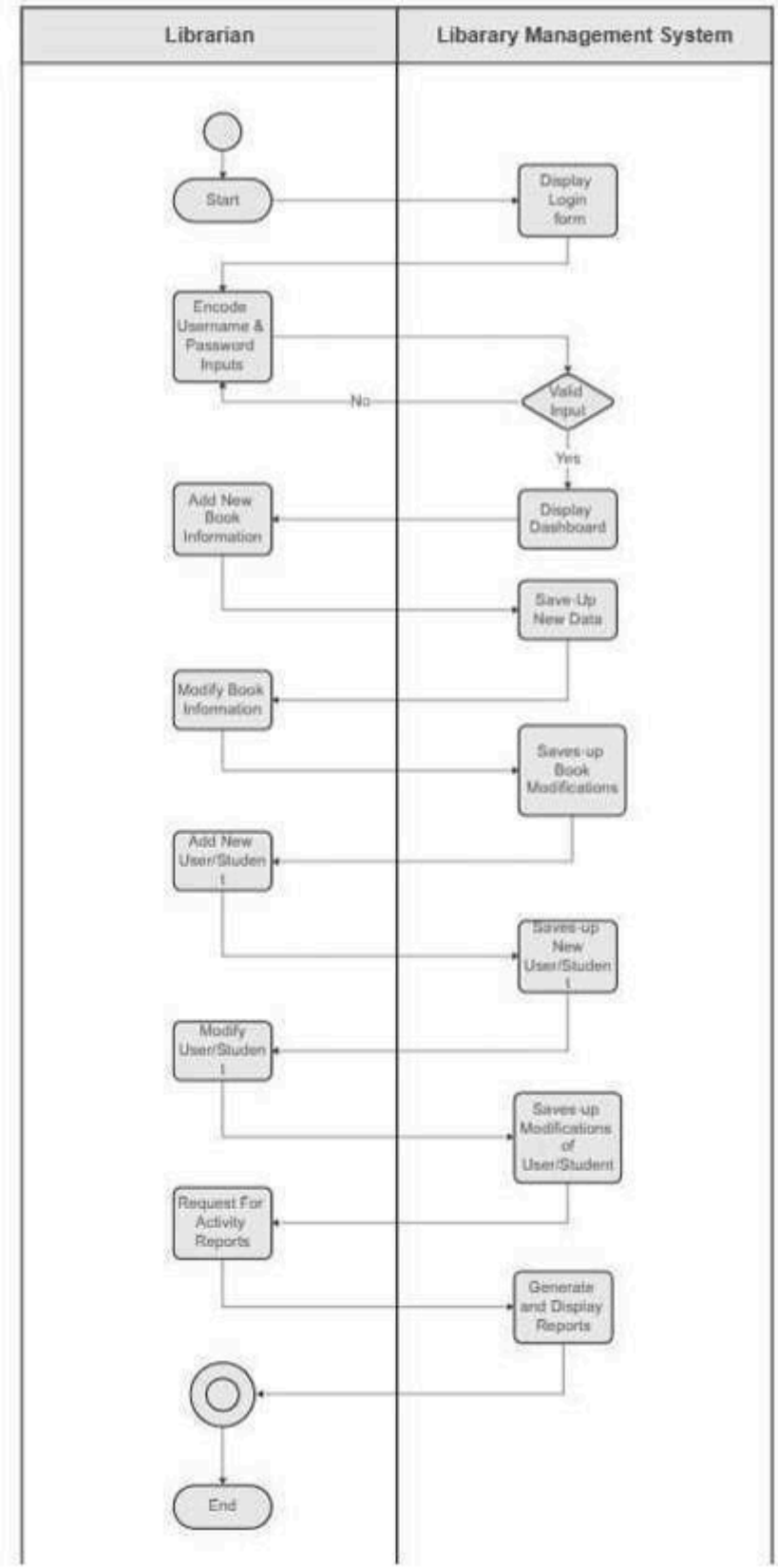


Explanation:



The simple activity diagram shows the basic steps followed when a user wants to issue a book. The process starts with an inquiry about the book, followed by checking book availability. If the book is not available, the process ends. If it is available, the system then validates the member. If the member is not valid, they are asked to register. After validation, the system checks how many books the member has already borrowed. If the maximum limit is exceeded, the book is not issued. Otherwise, the book is issued, and the system updates the book status. This simple diagram covers only the main flow of requesting, verifying, and issuing a book.

Activity Diagram(Advanced):



Explanation:

The advanced activity diagram provides a more detailed and complete view of the book-issuing workflow. In addition to checking availability and validating the member, it clearly shows decision points such as book unavailability, invalid member status, and quota limits. It also includes additional actions like adding member book issue details, updating the complete book record, and handling both "Book not issued" and "Issue Books" outcomes. By showing all alternative paths and internal updates, the advanced diagram presents the full operational logic of how the library processes book inquiries, member eligibility, quota checks, issuing actions, and record maintenance. It reflects the complete backend process of issuing a book in a real library system.

4. Stock Maintenance System SRS:

Stock Maintenance System.

Problem statement:

manual stock tracking often leads to errors, delays & stockouts, making it difficult to manage inventory effectively. A SMS is needed to automate stock updates, track product availability and generate timely reports.

1. Introduction:

1.1 Purpose:

The purpose of this document is to define the requirements for a stock maintenance system that automates inventory tracking, reduces cost, and real-time stock updates.

1.2 Scope:

System allows buissneses to manage products, update stock levels & generate reports.

1.3 Overview:

The system will include:

- Product catalog, stock report
- automatic stock updates
- Low stock alert

2. General description

The stock maintenance system will maintain real-time inventory data. It will track incoming and outgoing stock, notify users about shortages, and allow administrators to manage product details efficiently.

3. Functional requirements.

~~User Interface : Dashboard for admin & staff~~

- System shall allow adding, updating & deleting product details
- System shall update stock levels automatically
- System shall generate inventory reports
- System shall send alerts for low stock items

4. Interface requirements

User Interface : Dashboard for admin, staff & staff

External Interface : Barcode scanner support (optional)

5. Performance requirements :

- Must handle 10,000 stock items effectively
- Stock updates should reflect <2 seconds

6. Design Constraints

- Must use relational database (MySQL)
- Should run on standard desktop systems.

7. Non-functional requirements.

- Security : Role-base access for staff / admin
- Scalability : Should support expanding product ~~expiries~~ categories
- Usability : Simple dashboard for non-tech staff

8. Preliminary Schedule and Budget :

Schedule

Month 1 : Requirement and design

Month 2 : Development

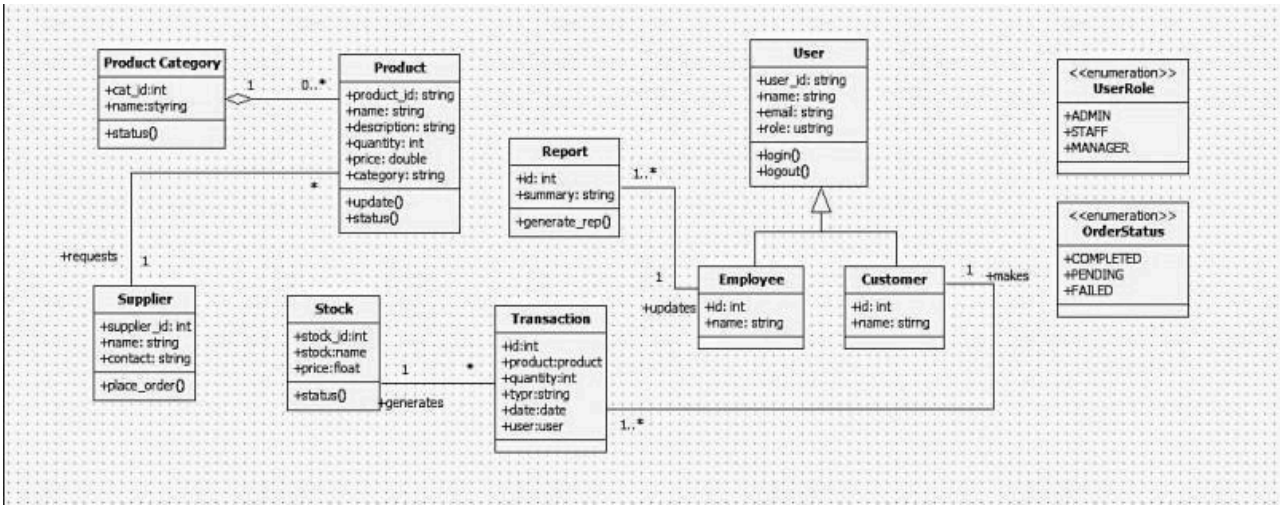
Month 3 : Testing

Budget

Development + setup : $7,000

Maintenance : $1500 / year.

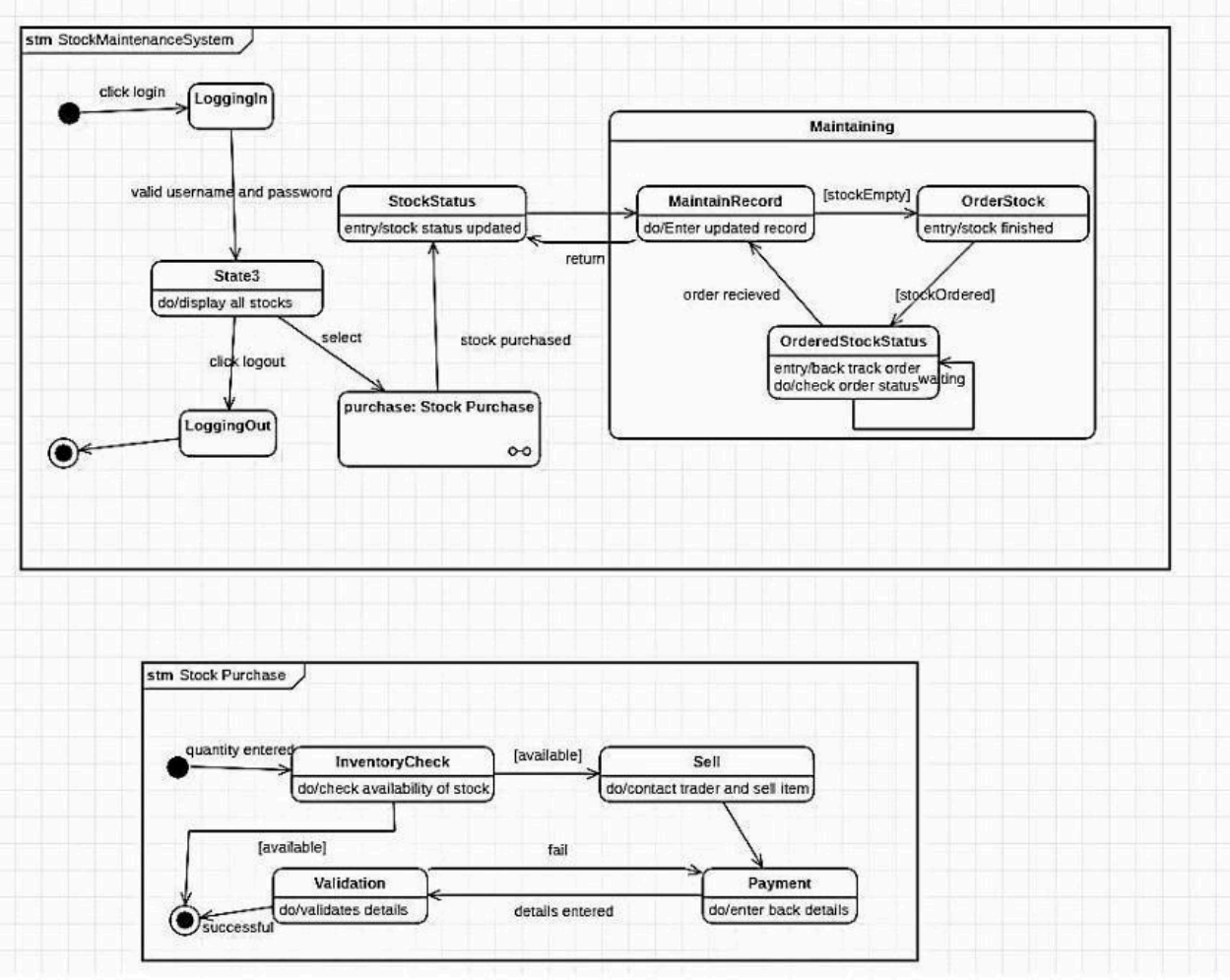
Class Diagram:



Explanation:

This class diagram visually represents the key entities and their relationships in a Sales Management System, showing how different objects such as Users, Customers, Admins, Suppliers, Stocks, Products, Transactions, and Sales interact within the system. Each class is defined with specific attributes and methods, and the arrows indicate relationships—such as Customers making Transactions, Admins managing Stocks and contacting Suppliers, and Sales being associated with Customers—highlighting the flow of information and control throughout the platform. This structure helps ensure efficient handling of sales, inventory, user actions, and supplier coordination, forming a cohesive framework for managing business operations digitally.

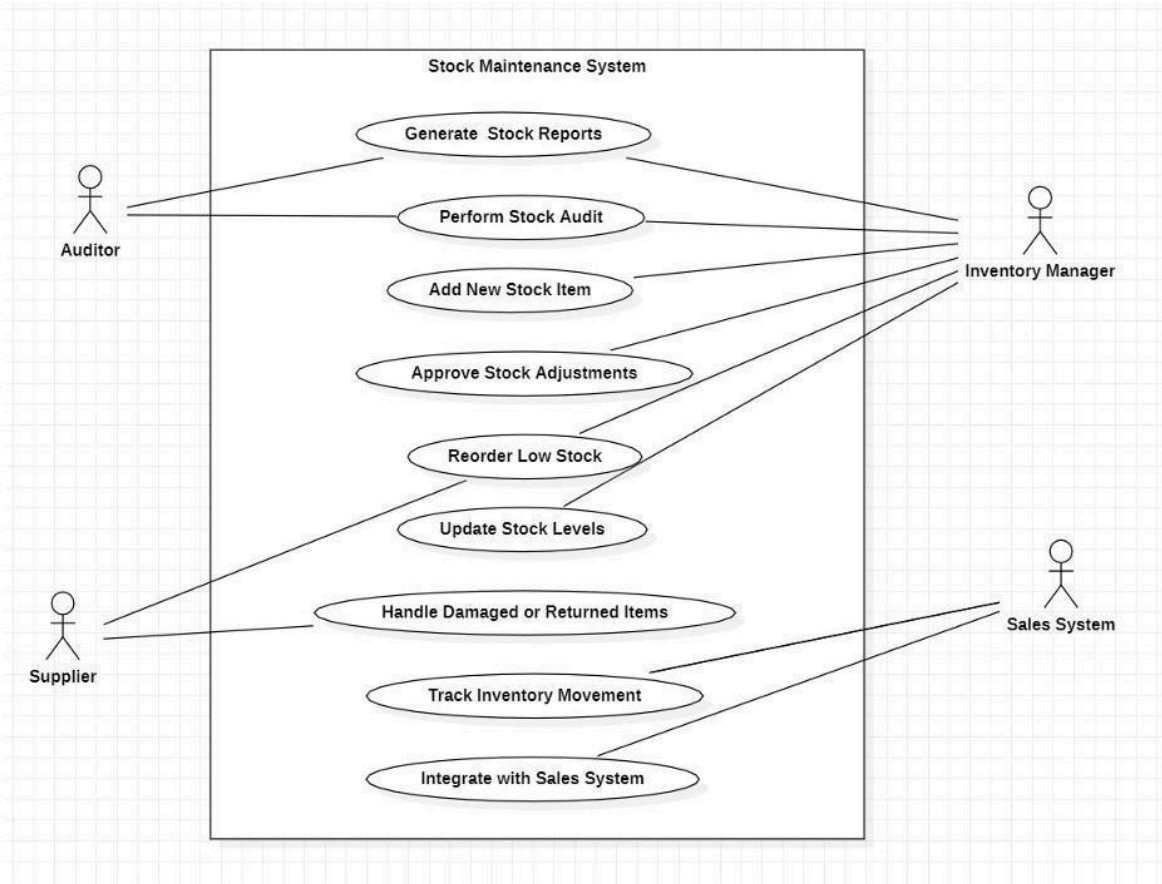
State Diagram:



Explanation:

This advanced state diagram models the Stock Maintenance System, focusing on two core processes: managing overall stock status and handling stock purchase transactions. The top section illustrates user actions like logging in, navigating the homepage to view stocks, initiating stock purchases, and logging out. Once inside the maintenance area, states transition between updating records, ordering new stock if inventory runs out, and tracking the status of those orders—ensuring stock levels remain sufficient through cyclic monitoring and updating.

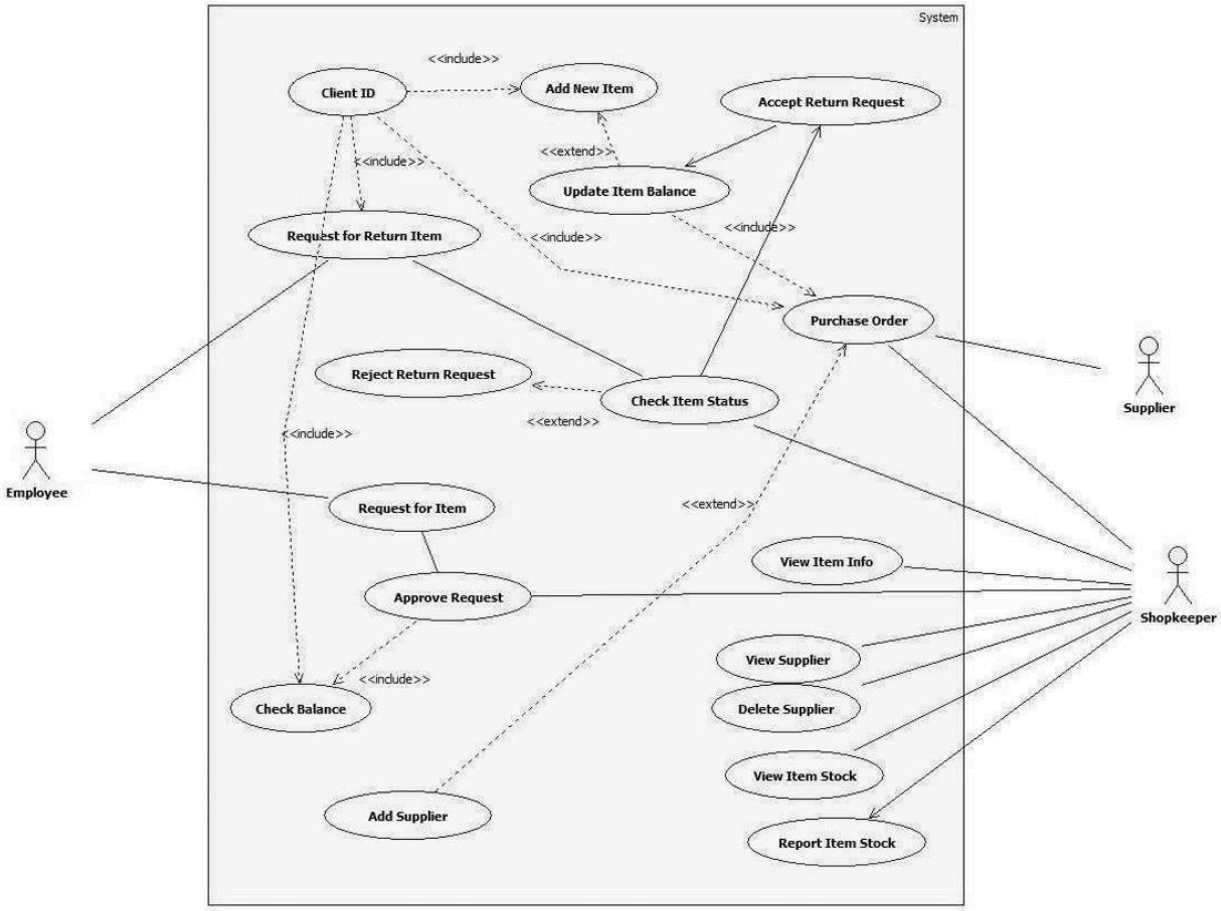
Use case diagram(Simple):



Explanation:

This use case diagram illustrates the interactions between two primary actors—Store Manager and Supplier—in a stock management system. The Store Manager handles inventory operations such as adding, updating, and viewing stock levels, as well as generating reports for analysis. The Supplier is responsible for requesting supply and delivering stock to the store. Each actor is linked to specific system functions, clarifying their roles and responsibilities. This visual representation helps in understanding system requirements, streamlining workflows, and guiding software development. It ensures that all user interactions are captured, making the system more efficient, user-centric, and aligned with business objectives.

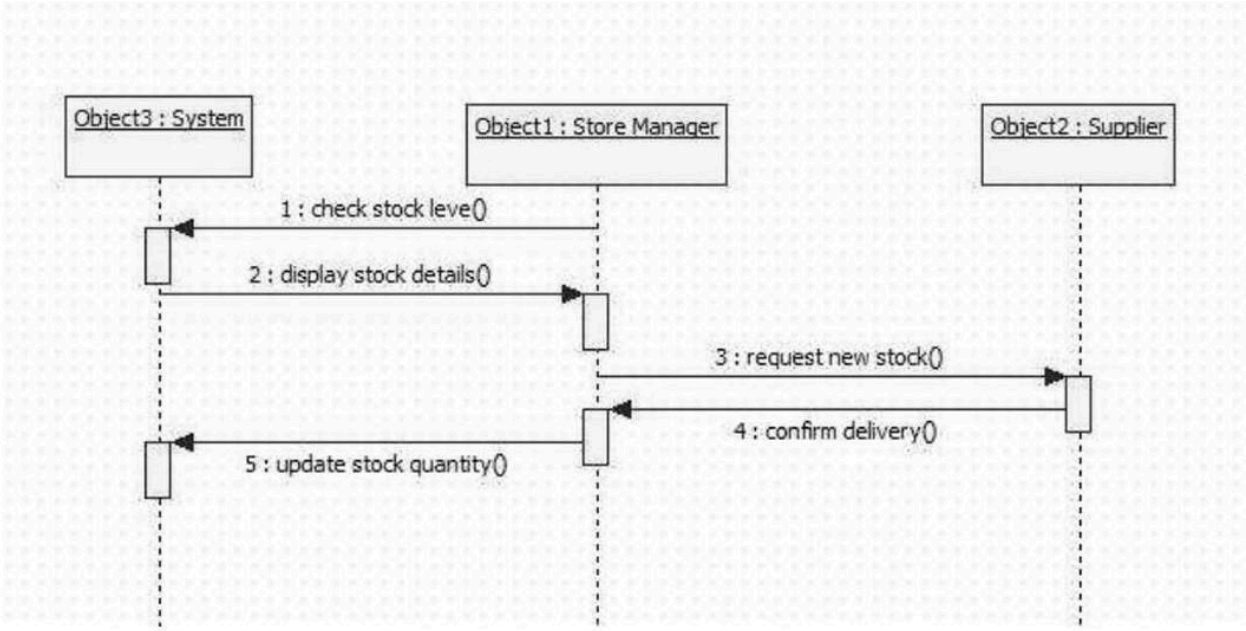
Use Case Diagram(Advanced):



Explanation:

This use case diagram outlines the roles of Manager, Store Staff, and Supplier in a retail system. The Manager oversees buying stock, making payments, and supervising staff. Store Staff handle inventory reporting, product quality checks, and selling stock. The Supplier delivers orders, receives payments, and manages quality issue reports. Relationships like <> and <> show dependencies—for example, buying stock includes giving payment and may extend to reporting quality issues, which in turn includes returning damaged goods. This structured view clarifies responsibilities, enhances system design, and ensures smooth coordination among stakeholders for efficient inventory and supply chain management.

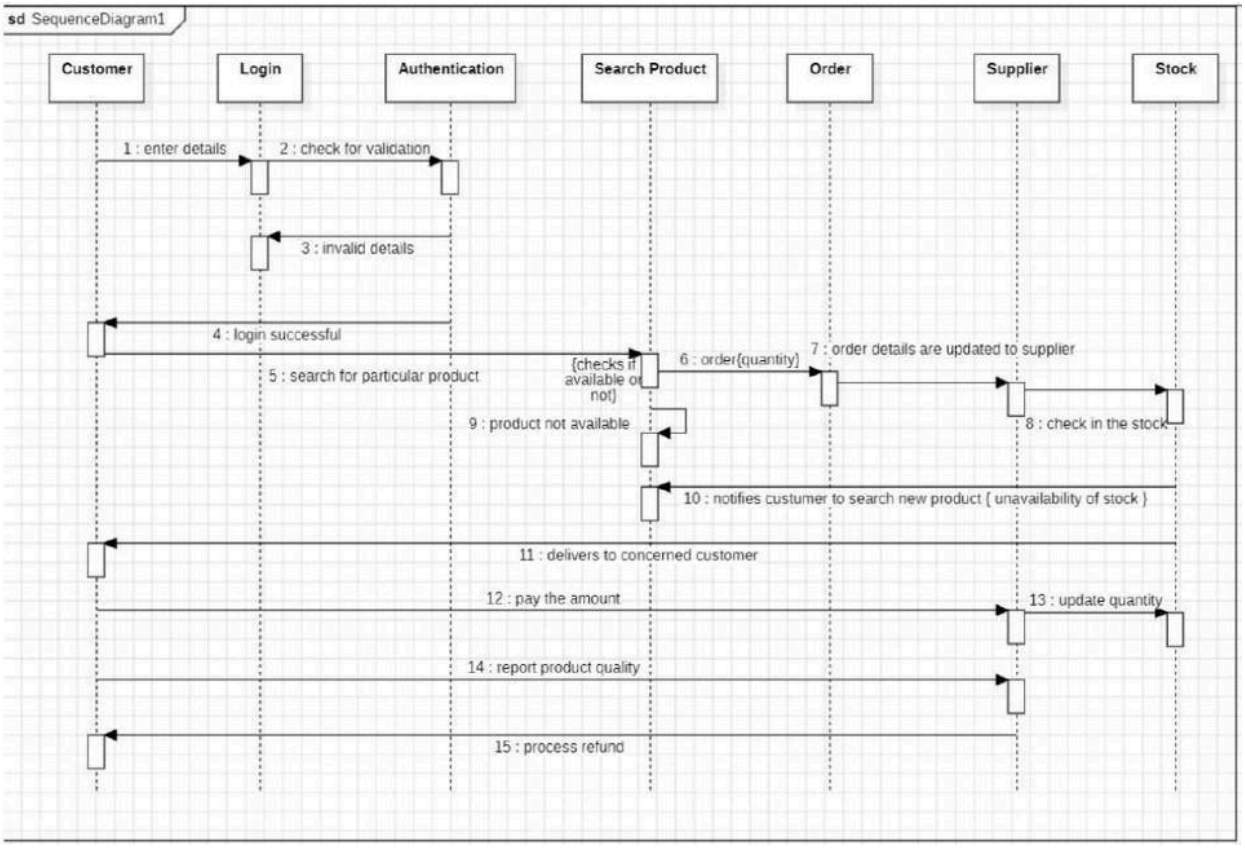
Sequence Diagram(Simple):



Explanation:

This UML sequence diagram illustrates the interaction between the Store Manager, Supplier, and System in a stock management workflow. The Store Manager initiates the process by checking stock levels through the System, which responds by displaying stock details. If stock is insufficient, the Manager requests new stock from the Supplier, who confirms delivery. Following this, the Manager updates the stock quantity in the System to reflect the new inventory. This sequence clearly outlines the communication flow and operational dependencies among the actors and system components, helping developers understand system behavior and ensuring accurate implementation of inventory-related functionalities.

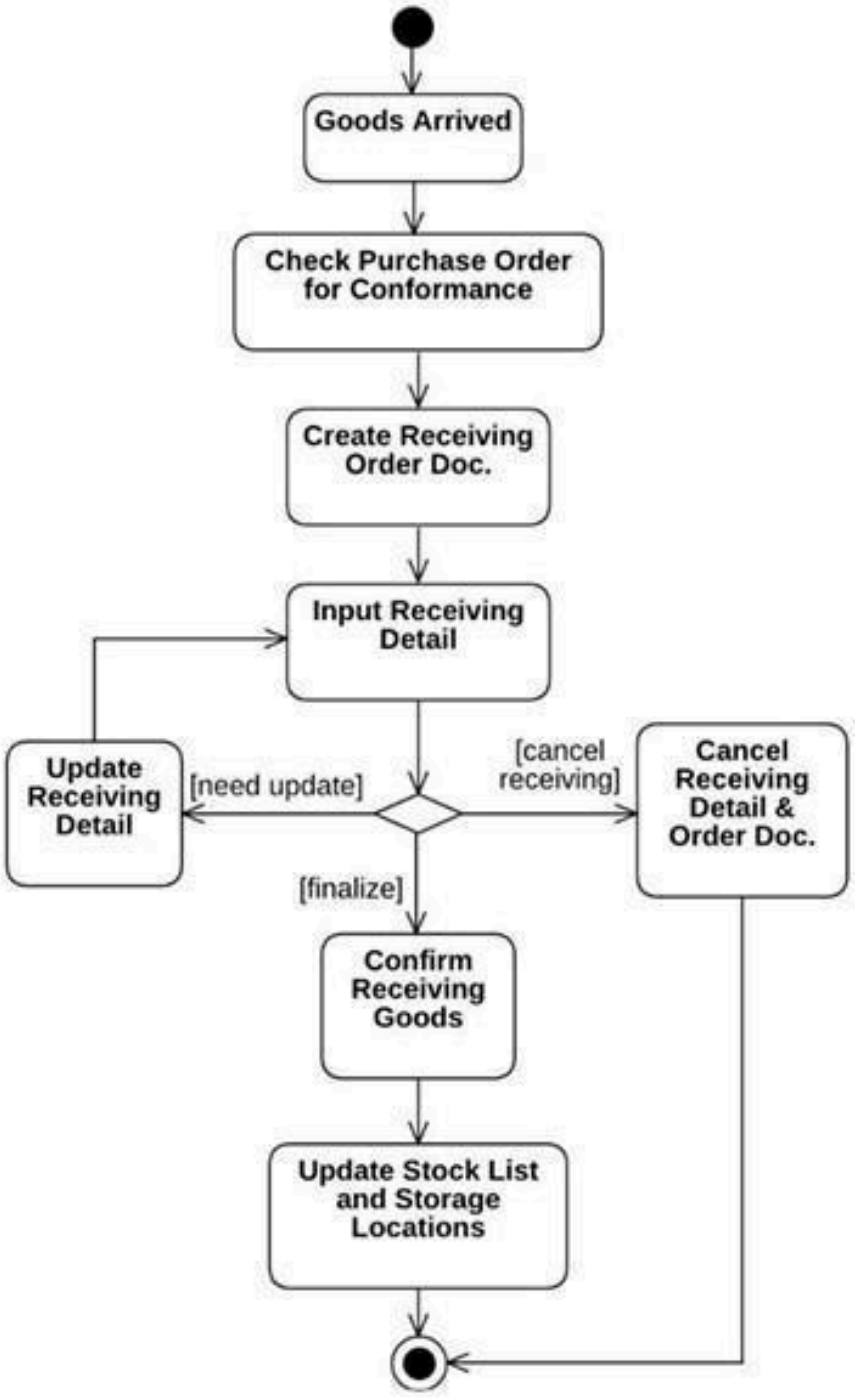
Sequence Diagram(Advanced):



Explanation:

This UML sequence diagram for the Old Stock Management System outlines the step-by-step interaction between entities like Customer, Login, Authentication, Search Product, Order, Supplier, and Stock. The process begins with customer detail entry and validation. Upon successful login, the customer searches for a product and checks its quantity. If unavailable, the system prompts them to search for alternatives. Once an order is placed, the supplier is notified, and delivery is arranged. The customer pays, stock quantities are updated, and any quality issues can be reported, triggering a refund process. This flow ensures efficient order handling, inventory tracking, and customer satisfaction.

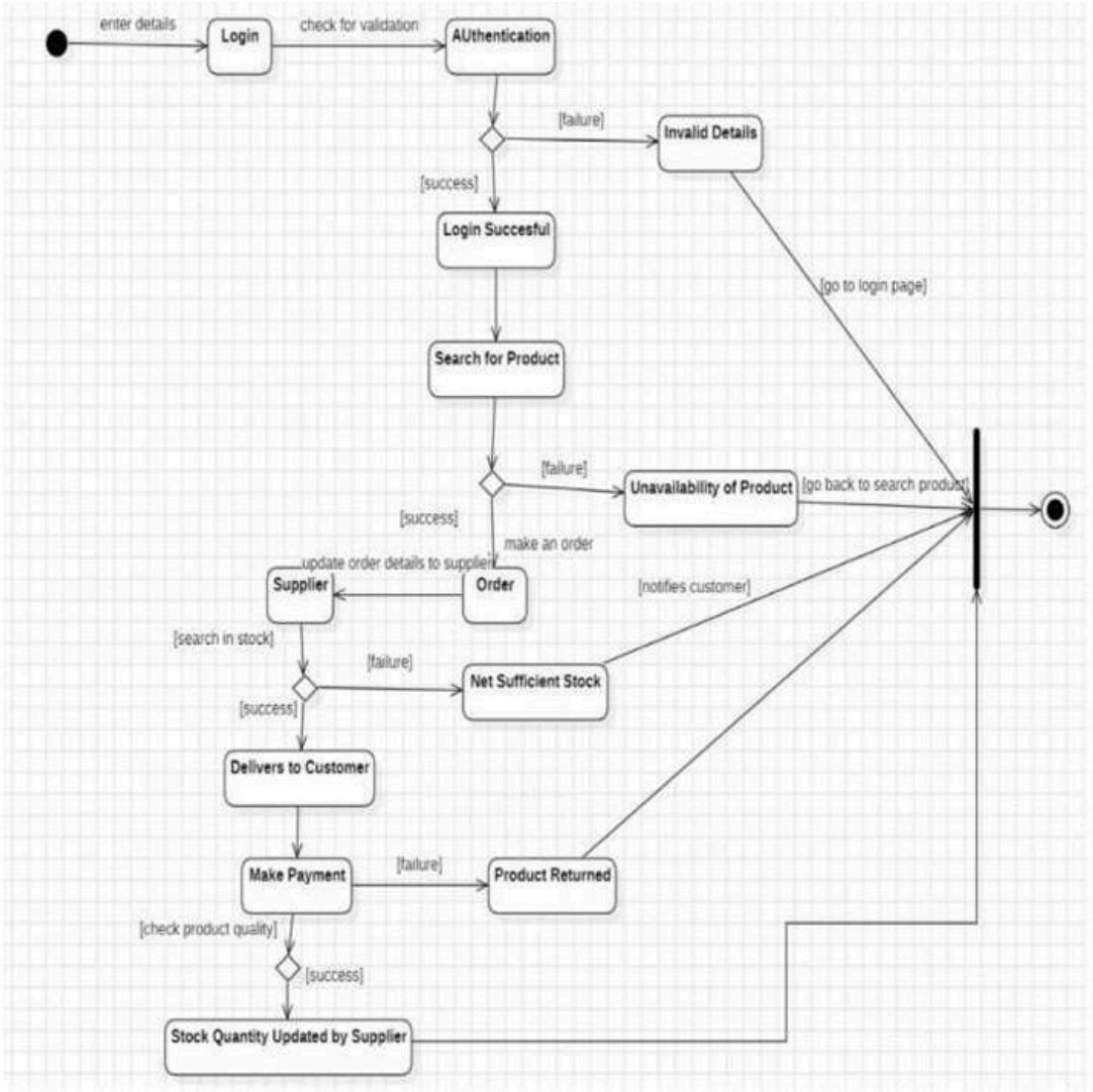
Activity Diagram(Simple):



Explanation:

This flowchart illustrates the complete journey of a customer in an online shopping system, starting from entering login details to receiving the product. The process begins with authentication, where invalid credentials redirect the user back to the login page. Upon successful login, the customer searches for a product. If the product is unavailable, they are prompted to search again. Once a product is selected, the system checks stock availability. If stock is insufficient or the product is returned, the customer is notified. This structured flow ensures that only valid users proceed and that product availability is verified before order placement.

Activity Diagram(Advanced):



Explanation:

After the order is successfully placed, the system interacts with the supplier to fetch stock and order details. The supplier searches inventory and updates the system accordingly. If delivery fails, the customer is notified immediately. Successful deliveries lead to the payment phase, where transaction failures are also communicated. This decision-based flow ensures transparency and error handling at each step. The supplier plays a critical role in maintaining stock accuracy and fulfilling orders. The system's ability to notify users at every failure point enhances customer experience and operational reliability, making the process robust and user-friendly.

5.Passport Automation System SRS:

Passport

Problem statement :

Manual passports application & verification process are slow, error-prone and inconvenient for applic
A passport automation system is needed to digitize applications, streamline verification and provide applicants with real-time status updates.

1. Introduction

1.1 Purpose

The purpose of this document is to defin requir for a passport automation system that simplifi the application, verification & issuance of passp

1.2 Scope

The system allow applicants to submit passport application online, upload required documents track application status & book appointments.

2. General Description :

The passport automation system will provide a user-friendly portal for applicants and an administrative module for officials.

3. Functional requirements.

- System allow applicants to register & submit application online.
- System allow users to upload documents
- System enable appointment scheduling
- System allow officials to verify and approve applications.

4. ~~Interface~~ Performance Requirements.

- Must handle thousands of concurrent application.
- Application submission and status queries should respond in <3 seconds.

5. Interface requirements

- User Interface: web portal and mobile app for applications
- ~~App~~ External interface: Integration with national ID databases & payment gateways.

6. Design Constraints.

- Must comply with government data security policies
- Must store data securely in an encrypted database

7. Non-functional Req

Security : strong encryption + user authentication

Reliability : 99.9% uptime

usability : Simple interface for applicants of all background.

8. Preliminary Schedule and Budget.

Month 1 : Req + Design

Month 2-3: Development

Month 4 Testing + Integration

Month 5 : Deployment

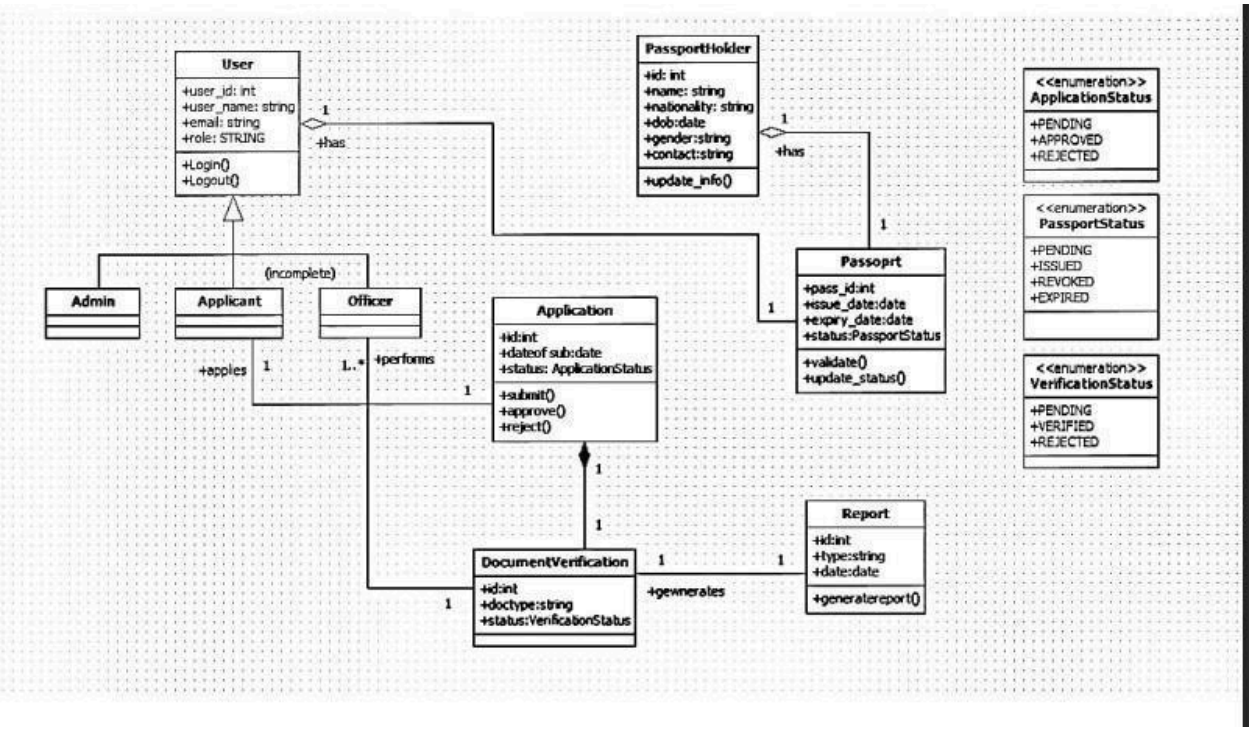
Budget

Development + setup $50,000

annual maintenance $10,000/year.

28/8

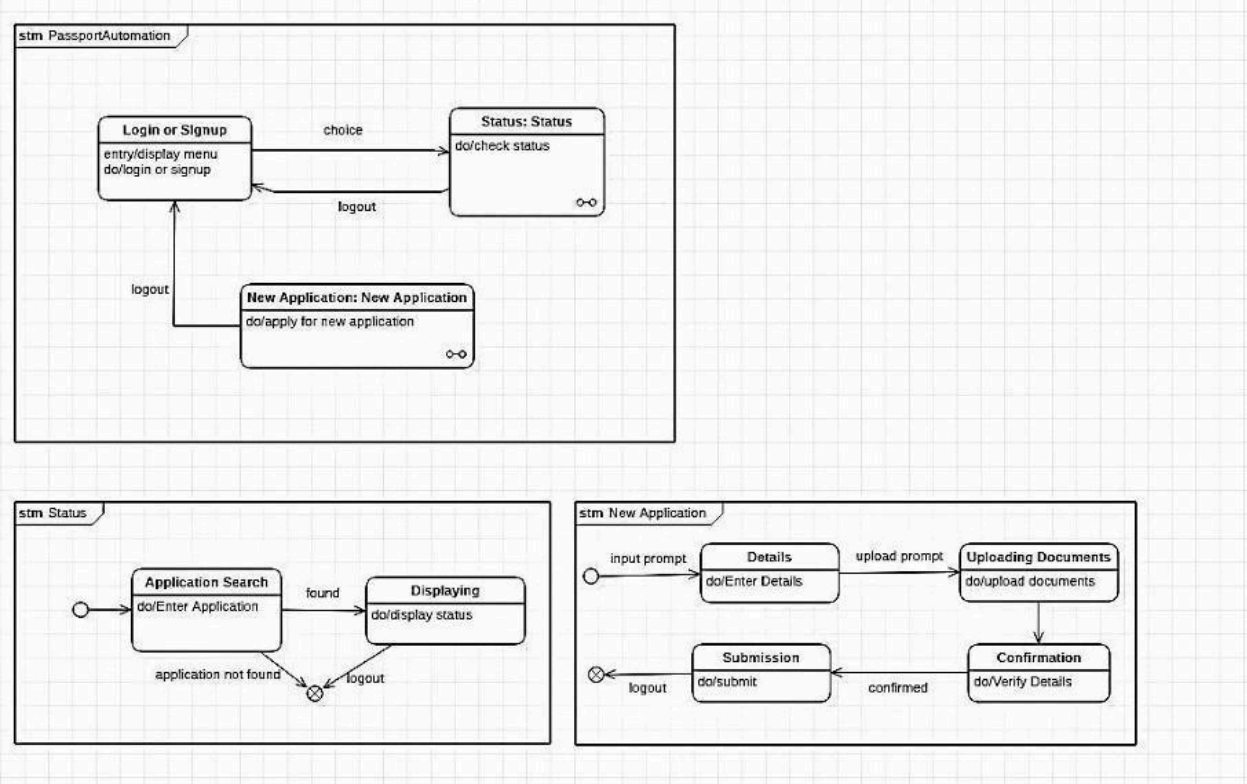
Class Diagram:



Explanation:

This class diagram represents an online Passport Automation System. An Applicant (inheriting basic details from Person and Registration) fills and submits an Application, which an Administrator validates, approves, or declines. Once approved, the applicant books an Appointment for document verification. At the center, Document Verification staff verify the applicant's original documents (docType, place, etc.) during the scheduled slot. The system maintains applicant info, tracks application status, manages appointments, and logs all actions (register, update, cancel, reschedule) while ensuring only verified and approved applicants proceed to the final appointment stage.

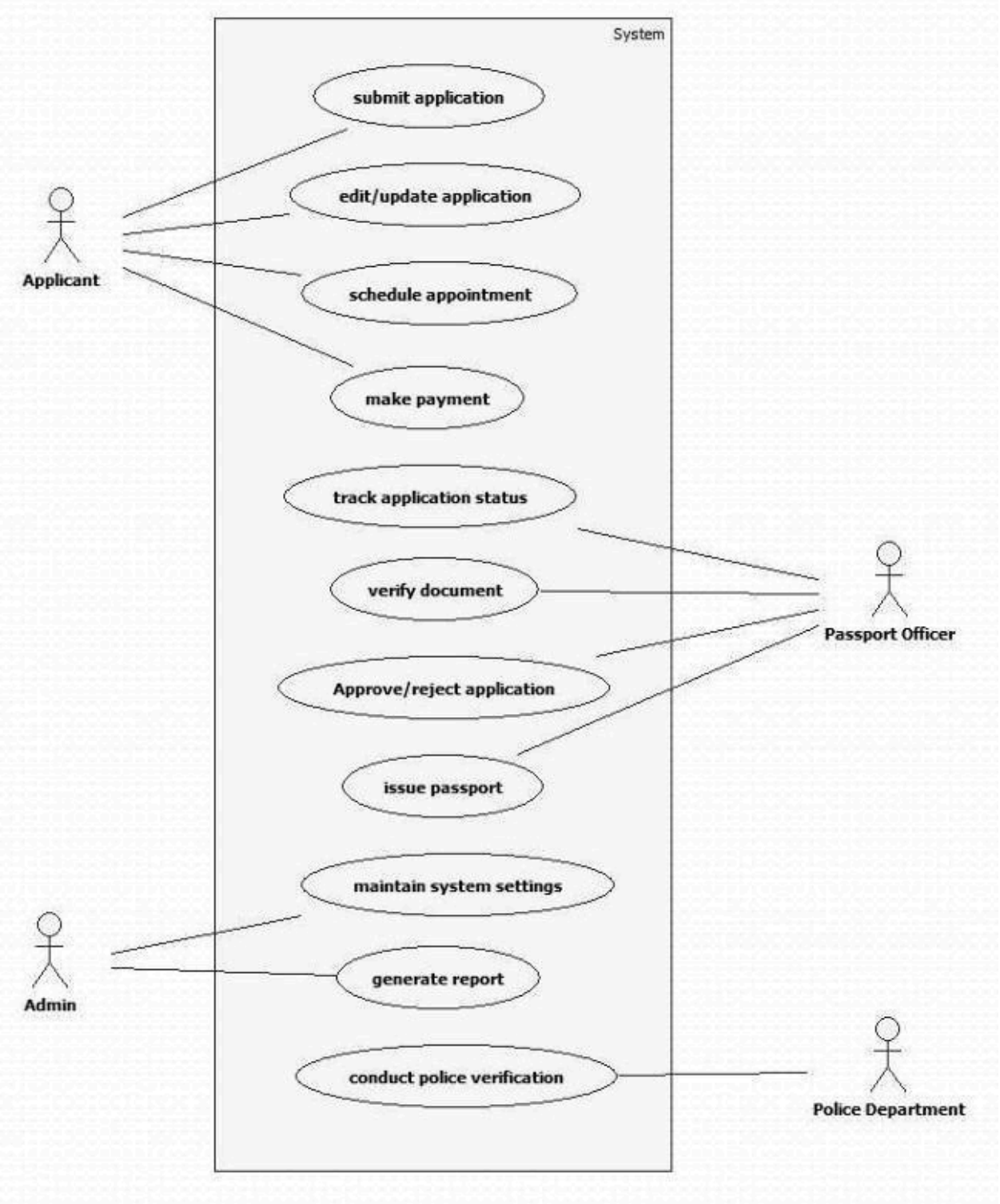
State Diagram:



Explanation:

The activity diagram presents the main menu of a Passport Automation System, where users begin by logging in or signing up. After successful authentication, they are prompted to choose between two primary options: checking the status of an existing application or starting a new passport application. The "Status" path allows users to enter their Application ID; if the record is found, the current status is displayed instantly. If not found, the process terminates gracefully with a logout. This streamlined design enables applicants to quickly track progress without re-submitting documents, enhancing user convenience and reducing support queries.

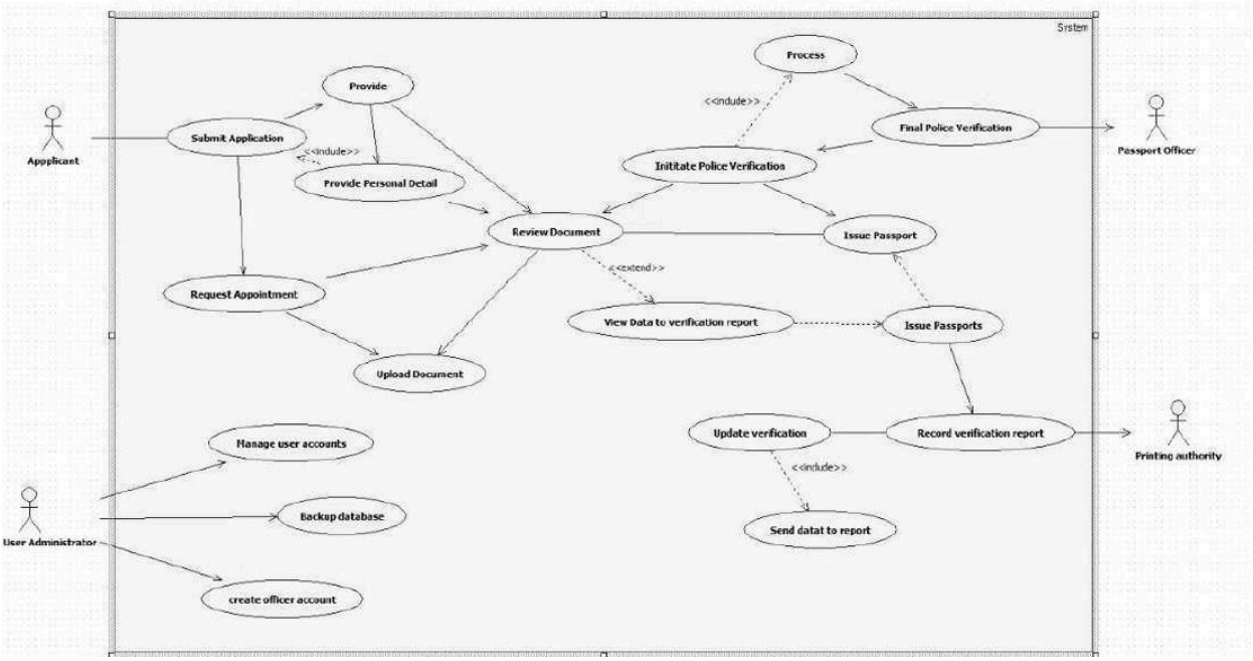
Use case diagram(Simple):



Explanation:

This use case diagram represents a Passport Automation System involving three actors: Applicant, Passport Officer, and Admin. The Applicant can Apply for Passport, Schedule Appointment for document verification, and Check Status of their application anytime. The Passport Officer is responsible for Verify Documents during the appointment and subsequently Issue or Reject the Passport based on authenticity and police verification reports. The Admin handles administrative tasks by managing user accounts (Manage Users) and updating application statuses (Update Application Status) throughout the process. The diagram clearly separates citizen-facing, officer-level, and administrative functions, ensuring smooth, role-based interaction with the system.

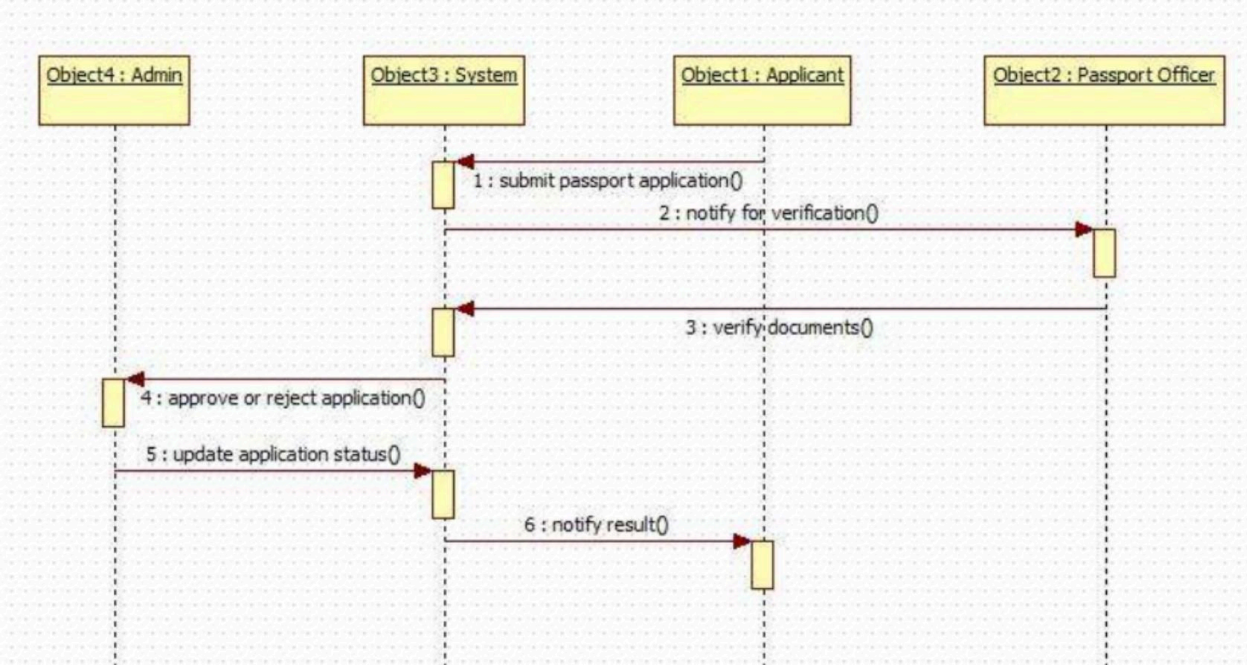
Use Case Diagram(Advanced):



Explanation:

The use case diagram outlines a comprehensive Passport Automation System with four actors. The Applicant registers on the portal, logs in, fills the application form, pays fees via an external Payment Gateway, tracks status, and schedules an appointment. The "Schedule Appointment" use case includes receiving notifications and mandatory document verification. The Passport Officer verifies documents, conducts interviews, and approves or rejects the application. The Police actor performs background verification and submits the police report. This citizen-centric flow ensures online submission, transparent tracking, and seamless integration of payment and notification services.

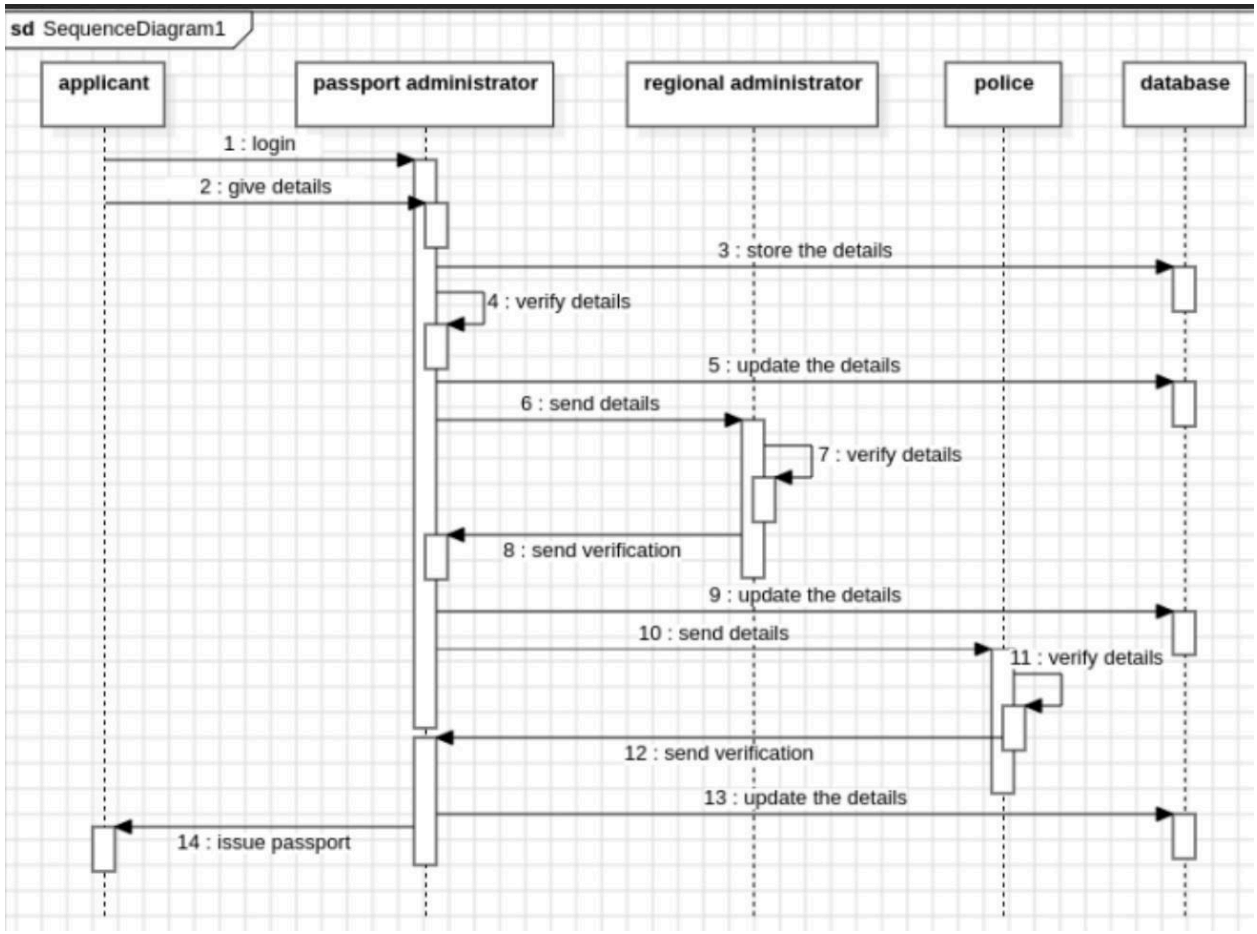
Sequence Diagram(Simple):



Explanation:

This sequence diagram illustrates the interaction flow in a Passport Automation System. The Applicant (Object1) initiates by submitting the passport application to the System (Object3). The System immediately notifies the Passport Officer (Object2) for verification. The Officer verifies the documents and sends the approval/rejection decision back to the System. The System then forwards this decision to the Admin (Object4) to update the official application status. Finally, the System notifies the Applicant about the result (approved or rejected). This clear, step-by-step message exchange ensures transparent communication, proper verification, and timely status updates among applicant, officer, system, and admin.

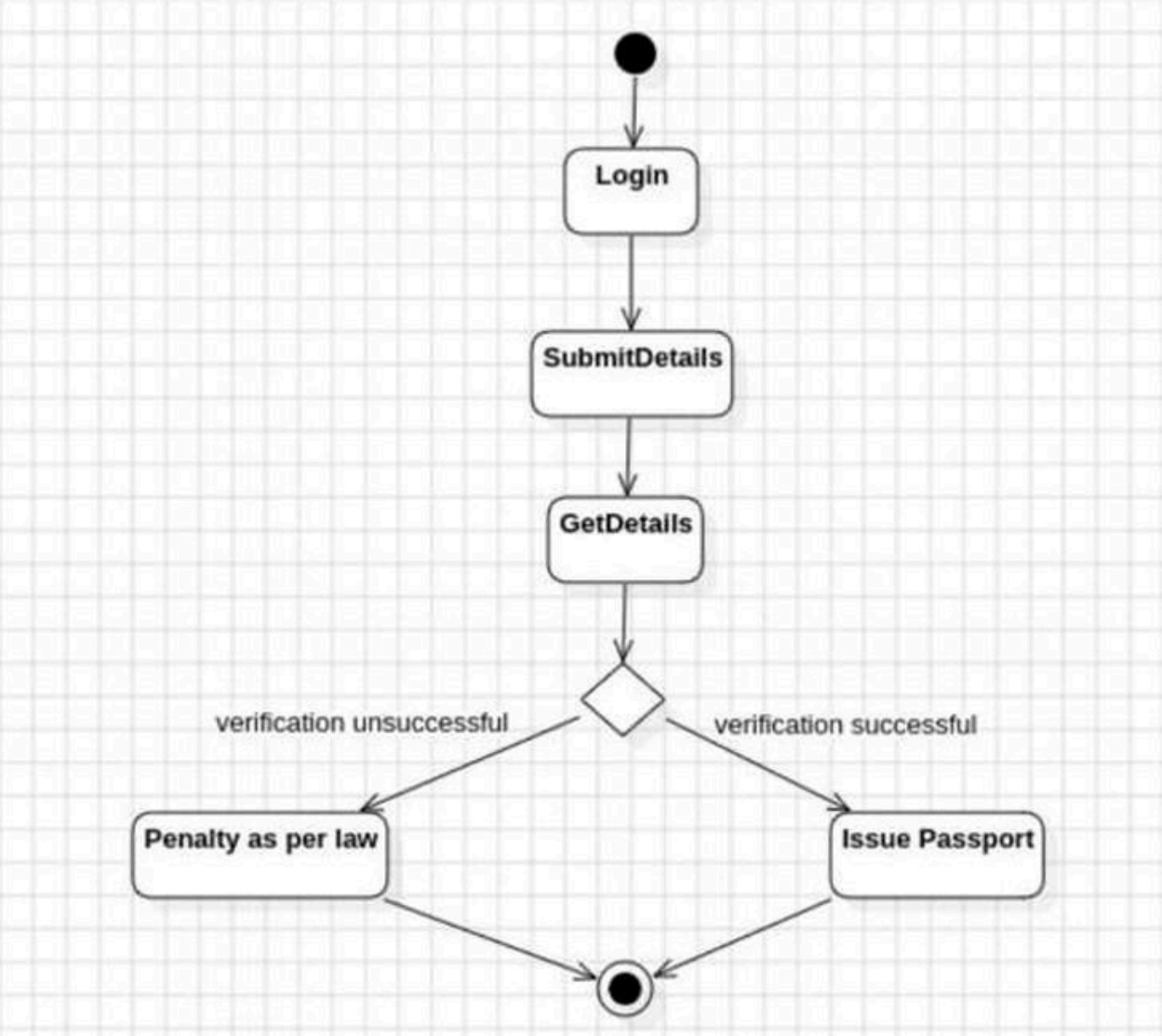
Sequence Diagram(Advanced):



Explanation:

The sequence diagram depicts the complete passport application verification process. The Applicant first logs in and submits personal details to the Passport Administrator. The Administrator verifies the details, stores them in the Database, and forwards them to the Regional Administration. The Regional Administration updates the Database and sends the details to the Police for background verification. After the Police verify and return the report, the Regional Administration updates the Database again and notifies the Passport Administrator. This multi-level verification involving local, regional, and police authorities ensures thorough scrutiny of applicant information before final processing.

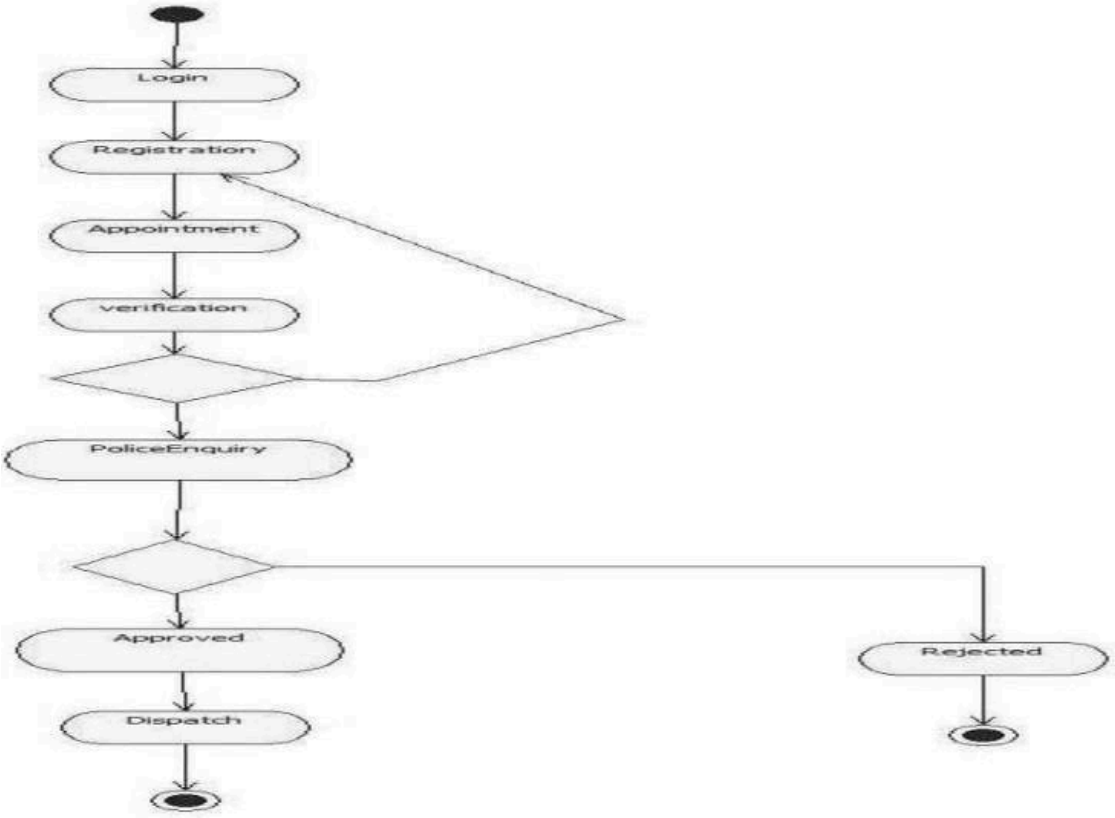
Activity Diagram(Simple):



Explanation:

This activity diagram illustrates the passport renewal/issuance process under strict verification. The applicant begins by logging in, submits personal and supporting details, and the system retrieves and verifies the stored information. A decision node checks verification outcome: if successful, the passport is issued immediately; if unsuccessful (due to discrepancies, criminal record, or invalid documents), a penalty is imposed as per law. The simple, linear flow with a single critical decision point emphasizes automated background checks and legal compliance, ensuring only eligible applicants receive passports while enforcing penalties for fraudulent or ineligible cases.

Activity Diagram(Advanced):



Explanation:

The activity diagram outlines the complete passport application lifecycle. It begins with user Login followed by Registration of personal details. The applicant then books an Appointment for document verification, after which physical Verification takes place at the passport office. A crucial decision point follows verification: if documents and identity are valid, the process continues; otherwise, it may loop back or terminate. The flow then proceeds to Police Enquiry for background and criminal record checks, ensuring eligibility. This structured sequence guarantees that only verified applicants advance to the final approval stage.