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

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

ARUGUNTA HAMSIKA

1BM22CS054

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Oct 2024 -Jan 2025

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(22CS6PCSEO) laboratory has been carried out by ARUGUNTA HAMSIKA (1BM22CS054) during the 5th Semester Oct 2024- Jan2025.

Signature of the Faculty Incharge:

Anusha S
Assistant professor

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

TABLE OF CONTENTS

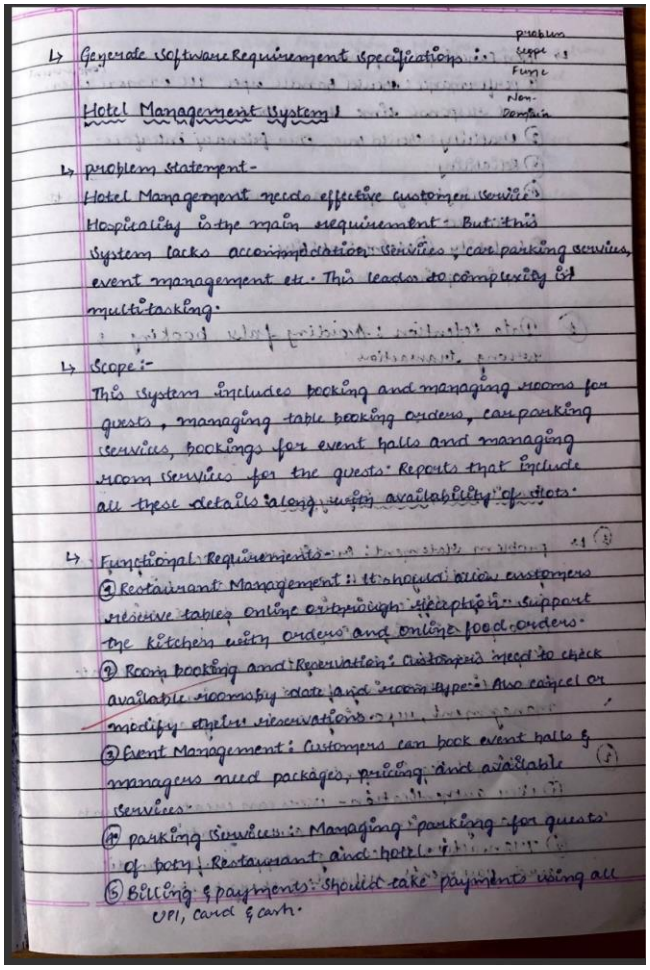
TOPIC	PAGE NO.
1. HOTEL MANAGEMENT SYSTEM	1
2. CREDIT CARD PROCESSING SYSTEM	14
3. LIBRARY MANAGEMENT SYSTEM	26
4. STOCK MAINTENANCE SYSTEM	37
5. PASSPORT AUTOMATION SYSTEM	49

1. Hotel Management System

1.1 Problem Statement

Design the software to support a computerized hotel management system for a network of hotels, integrating both manual operations by reception staff and an automated online portal. Each hotel maintains its own database to manage bookings, customer records, and financial transactions, while the online portal interacts with a central system to handle reservations, payments, and service requests. The system must enable guests to book rooms, reserve party halls, access amenities, and make payments securely while ensuring seamless integration with individual hotel systems. It should handle concurrent bookings and service requests accurately, maintain data confidentiality, and support centralized features like party hall reservations and restaurant access. Hotels will manage their own internal operations, and the cost of the shared system will be distributed based on the usage of online bookings and additional services.

1.2 Software Requirements Specification Document



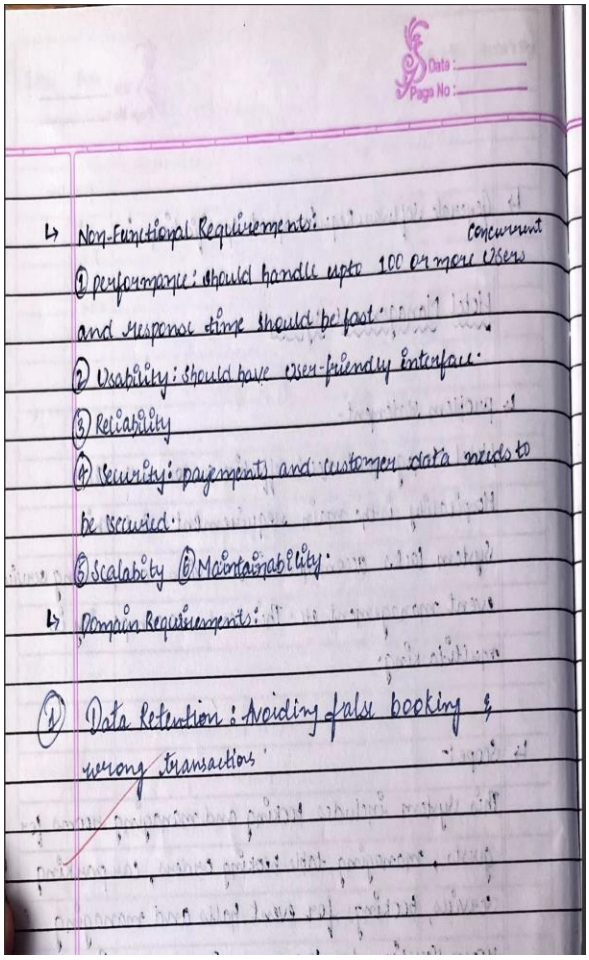
↳ Generate Software Requirement specifications :

Hotel Management System

↳ problem statement -
Hotel Management needs effective customer service. Hospitality is the main requirement. But this system lacks accommodation services, car parking service, event management etc. This leads to complexity in multitasking.

↳ Scope :-
This system includes booking and managing rooms for guests, managing table booking orders, car parking services, bookings for event halls and managing room services for the guests. Reports that include all these details along with availability of slots.

↳ Functional Requirements :
① Restaurant Management : It should allow customers reserve tables online or through chatbot. Support the kitchen with orders and online food orders.
② Room booking and Reservation : Customers need to check available rooms by date and room type. Also cancel or modify their reservations.
③ Event Management : Customers can book event halls & managers need packages, pricing and available services.
④ parking services : Managing parking for guests of both Restaurant and hotel.
⑤ Billing & payments : Should take payments using all UPI, Card & cash.



↳ Non-Functional Requirements :
① performance : should handle upto 100 or more Users Concurrent and response time should be fast.
② Usability : Should have user-friendly interface.
③ Reliability
④ Security : payements and customer data needs to be secured.
⑤ Scalability ⑥ Maintainability.

↳ Domain Requirements :

① Data Retention : Avoiding fake booking & wrong transactions

1.3 Class Diagram:

1.3.1 Simple Class Diagram

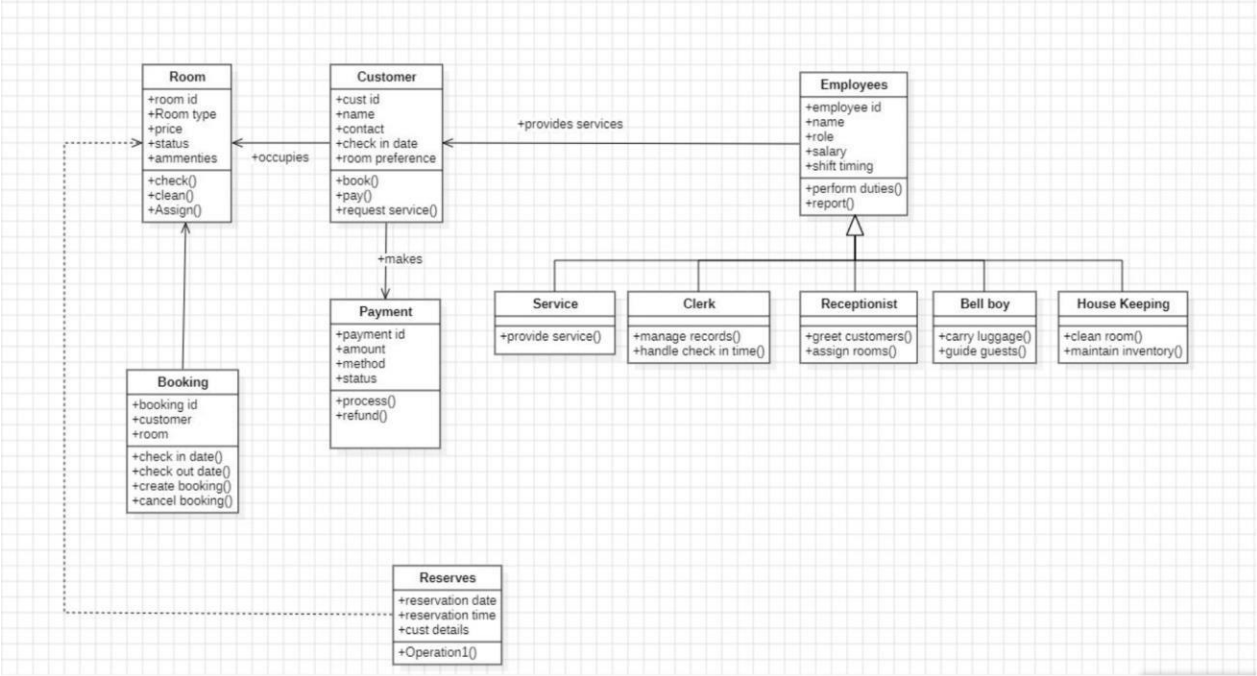


fig 1.3.1

Classes and Relationships:

- Customer: Books rooms and makes payments.
- Room: Includes details like room number, type, and guests.
- Employees: Manages hotel operations, divided into Manager and Personnel subclasses.
- Booking: Handles all the booking details
- Payment: Handles payment mode and transactions.

1.3.2 Advanced Class Diagram

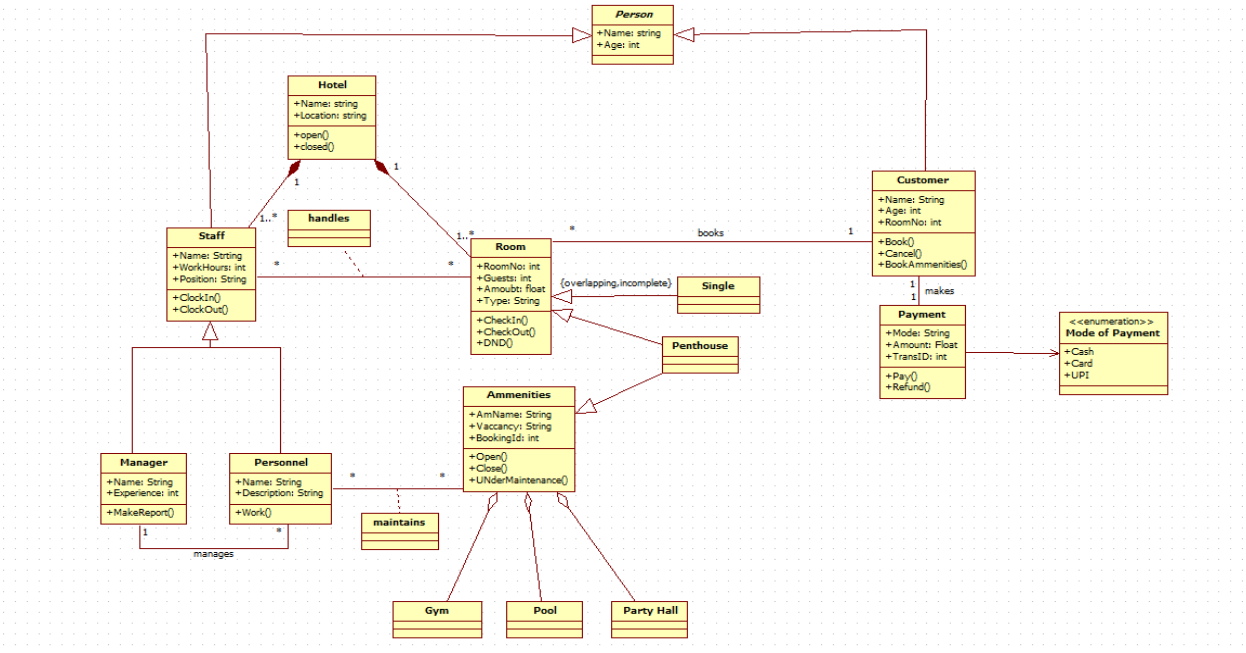


fig 1.3.2

This diagram represents the class structure of a hotel management system.

Key Classes:

- Person: A base class with attributes Name and Age.
- Hotel: Manages rooms and amenities using methods like open() and close().
- Room: Attributes include RoomNo, Guests, Amount, Type, and methods for checking in/out.
    - Single and Penthouse: Specialized room types inheriting from Room.
- Customer: A subclass of Person with methods like Book() and Cancel().
- Payment: Handles transactions with attributes like Mode and Amount.
- Amenities: Includes facilities like Gym, Pool, and Party Hall, managed by the hotel.
- Staff: Handles hotel operations, with subcategories:
    - Manager: Handles reports and staff management.
    - Personnel: General staff with a Work() method.

Relationships:

- Customers book rooms and amenities and Staff maintains hotel operations.
- Payments are associated with room bookings.

1.4 State Diagram

Simple State Diagram

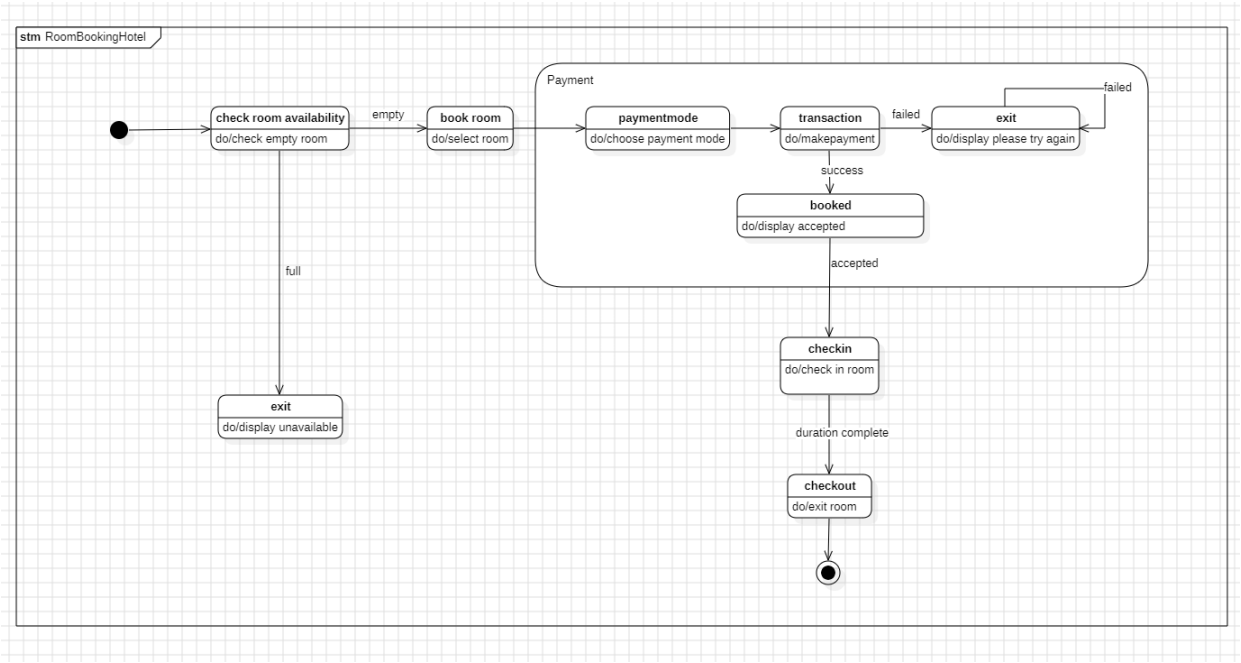


fig 1.4.1

States:

- Initial: Checking room availability.
- Intermediate: Booking, payment processing, and checking in.
- Final: Room checkout and completion of stay.

Transitions:

- Payment can lead to success (booking) or failure (exit).
- After checking in, guests proceed to the checkout phase post-duration completion.

Advanced State Diagram

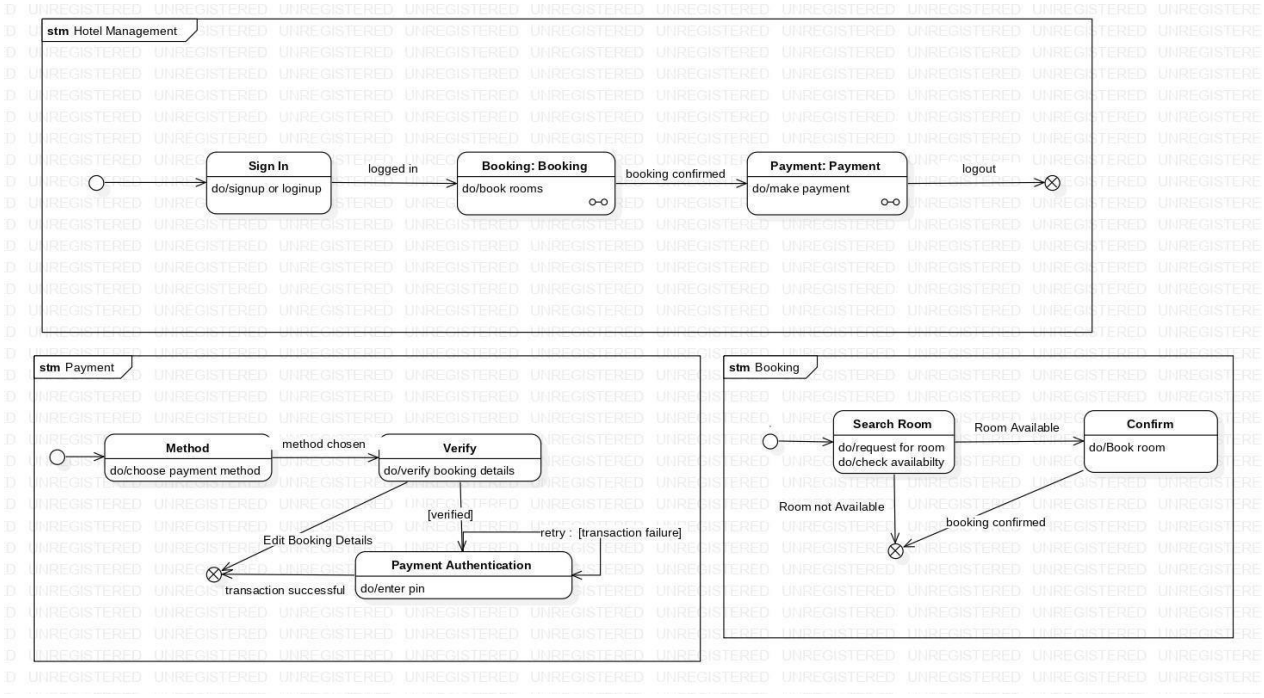


fig 1.4.2

Key States and Transitions:

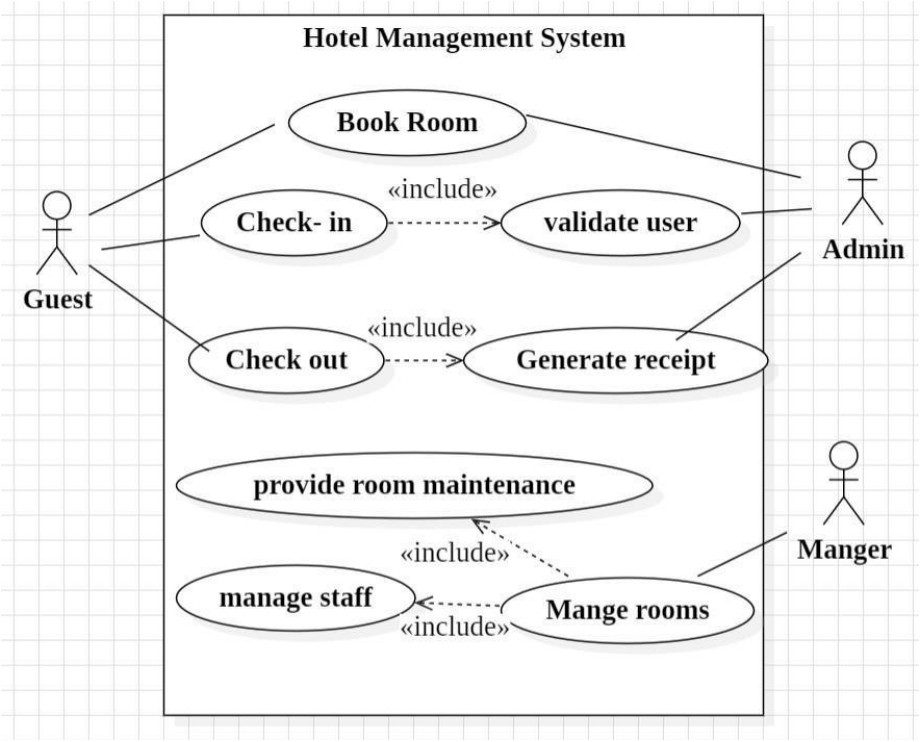
1. Initial State:
    - The diagram starts with an initial black circle, representing the starting point of the process.
2. States:
    - Idle State: Represents the system or object (e.g., a room) being available or awaiting action.
    - Registration: The state where the guest creates a new account.
    - Login: The state where credentials are verified.
    - Room Selection: A state where the guest chooses a room.
    - Booking: The state where room booking details are filled in and submitted.
    - Payment Processing:
        - The system transitions into processing payments after booking confirmation.
        - It involves communicating with the Payment Gateway.
    - Confirmation: The state where the system confirms the payment and booking.
    - Logout: A state for ending the session.
    - Maintenance (if applicable): For rooms or amenities undergoing maintenance.
3. Transitions:
    - Triggered by events such as login success, payment confirmation, or room availability checks.

- Specific actions (like sending payment signals, filling booking details) move the system from one state to another.

4. Final State:
    - The diagram ends with a filled black circle, representing the termination of the process, such as after the user logs out.

1.5 Use case diagram

Simple Use Case Diagram

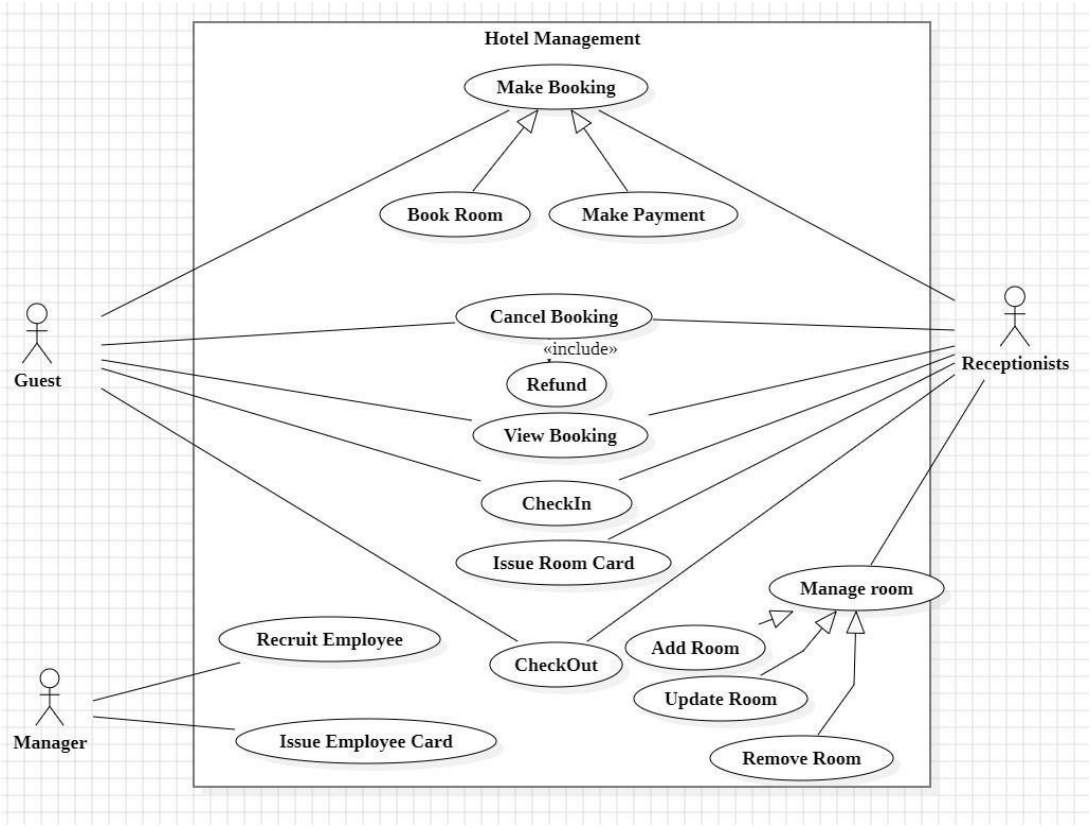


This diagram outlines the interactions between different actors and the system. It identifies the primary use cases:

1. Guest: Can book a room, check-in, and check out. Booking includes validating the user and generating a receipt.
2. Admin: Validates user details.
3. Manager: Manages rooms and staff, and oversees room maintenance.

The relationships between use cases are shown using include relationships, which indicate shared functionality such as validation and receipt generation.

1.5.2 Advanced Use Case Diagram



This diagram depicts the functionalities of a hotel management system and the roles interacting with it:

- Guest: Can make a booking, make payment, cancel booking (which includes refund), view booking, check-in, check-out, and issue a room card.
- Receptionists: Assist guests in booking, managing bookings, issuing room cards, and checking in/out. They also manage rooms (adding, updating, or removing).
- Manager: Handles employee recruitment and issuing employee cards.
- System Scope: Demonstrates the collaboration between different actors (guest, receptionist, manager) and their actions within the hotel management system.

1.6 Sequence Diagram

Simple Sequence Diagram

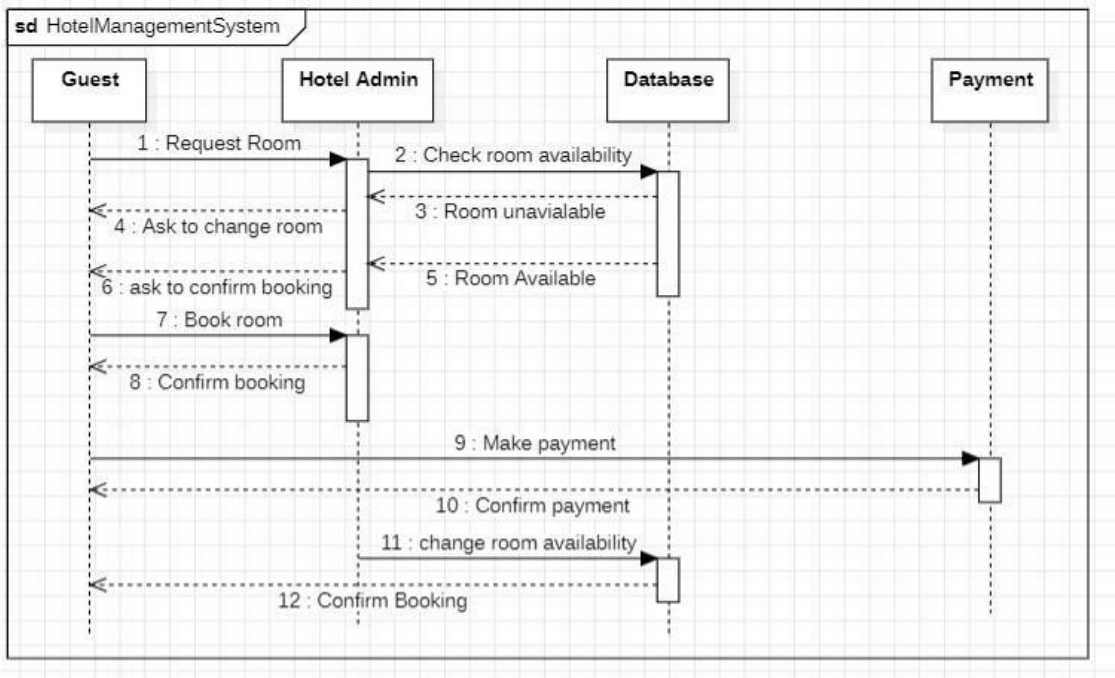


fig 1.6.1

1. Guest requests a room.
2. Hotel Admin checks room availability in the Database.
    - If unavailable, the admin suggests changes.
    - If available, the admin proceeds to confirm the booking.
3. Payment is made and confirmed through the Payment system.
4. The database is updated to reflect the room's availability status.
5. The final booking is confirmed.

This captures the step-by-step interactions to fulfill a guest's booking request.

Advanced Sequence Diagram

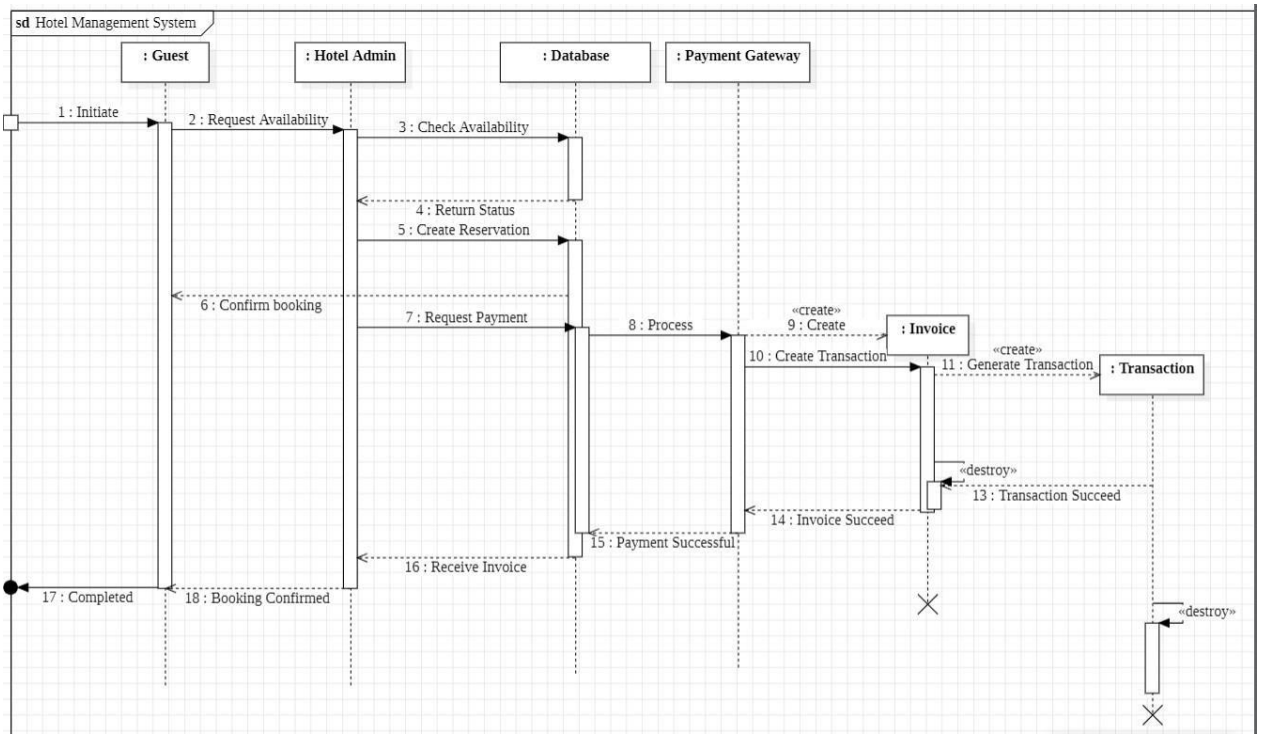


fig 1.6.2

This diagram explains the flow of events when a guest interacts with the hotel management system:

1. Guest Initiates Request: The guest starts by requesting room availability.
2. Hotel Admin Checks Availability: Admin queries the database for room availability.
3. Reservation Process:
    - Availability status is returned.
    - A reservation is created in the database.
4. Payment Process:
    - Payment is requested via the payment gateway.
    - Payment gateway processes the transaction and generates an invoice.
    - Transaction success or failure is communicated back to the system.
5. Booking Confirmation: Once payment is successful, the guest is notified with a booking confirmation.

1.7 Activity Diagram

Simple Activity Diagram

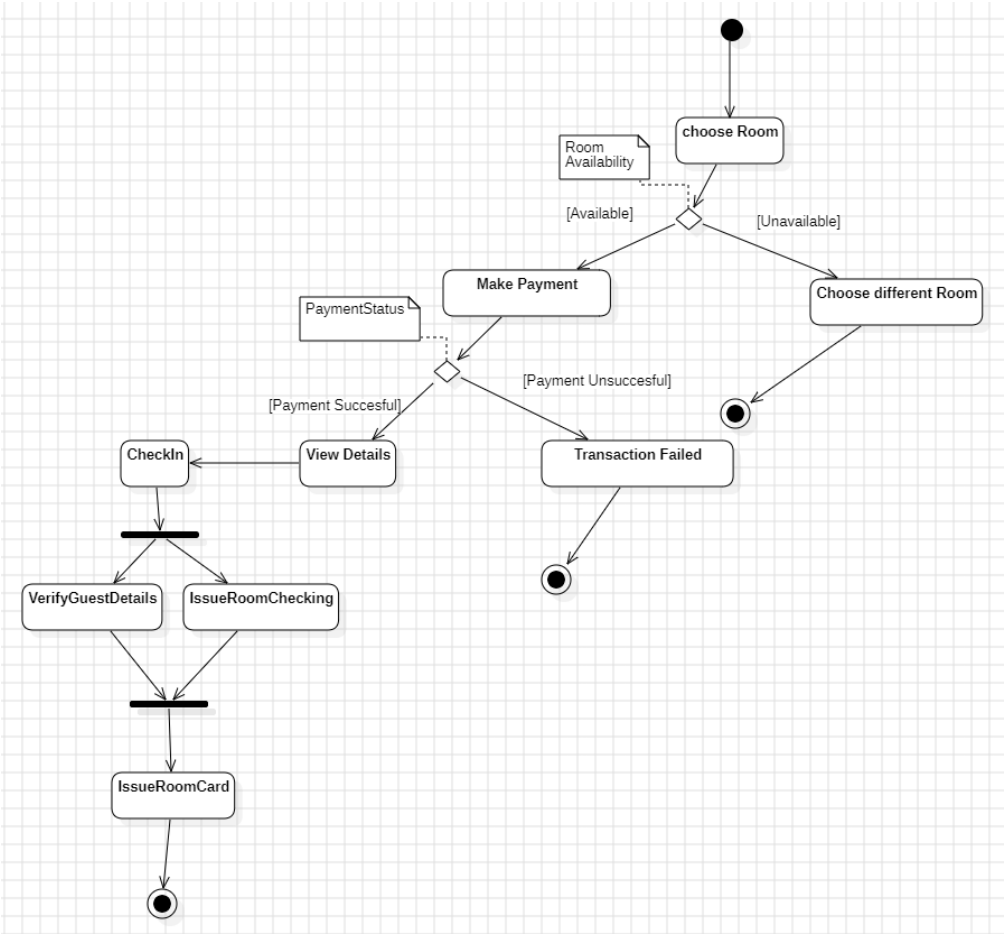


fig 1.7.1

This diagram represents the workflow of the room booking and check-in process:

1. Starts with the guest choosing a room.
2. Decision points check room availability:
    - If available, the guest proceeds to make payment.
    - If unavailable, the guest can choose a different room.
3. Successful payment leads to verification and issuing of room details.
4. Unsuccessful payment results in a transaction failure.
5. The check-in process includes verifying guest details and issuing a room card, completing the activity.

Advanced Activity Diagram

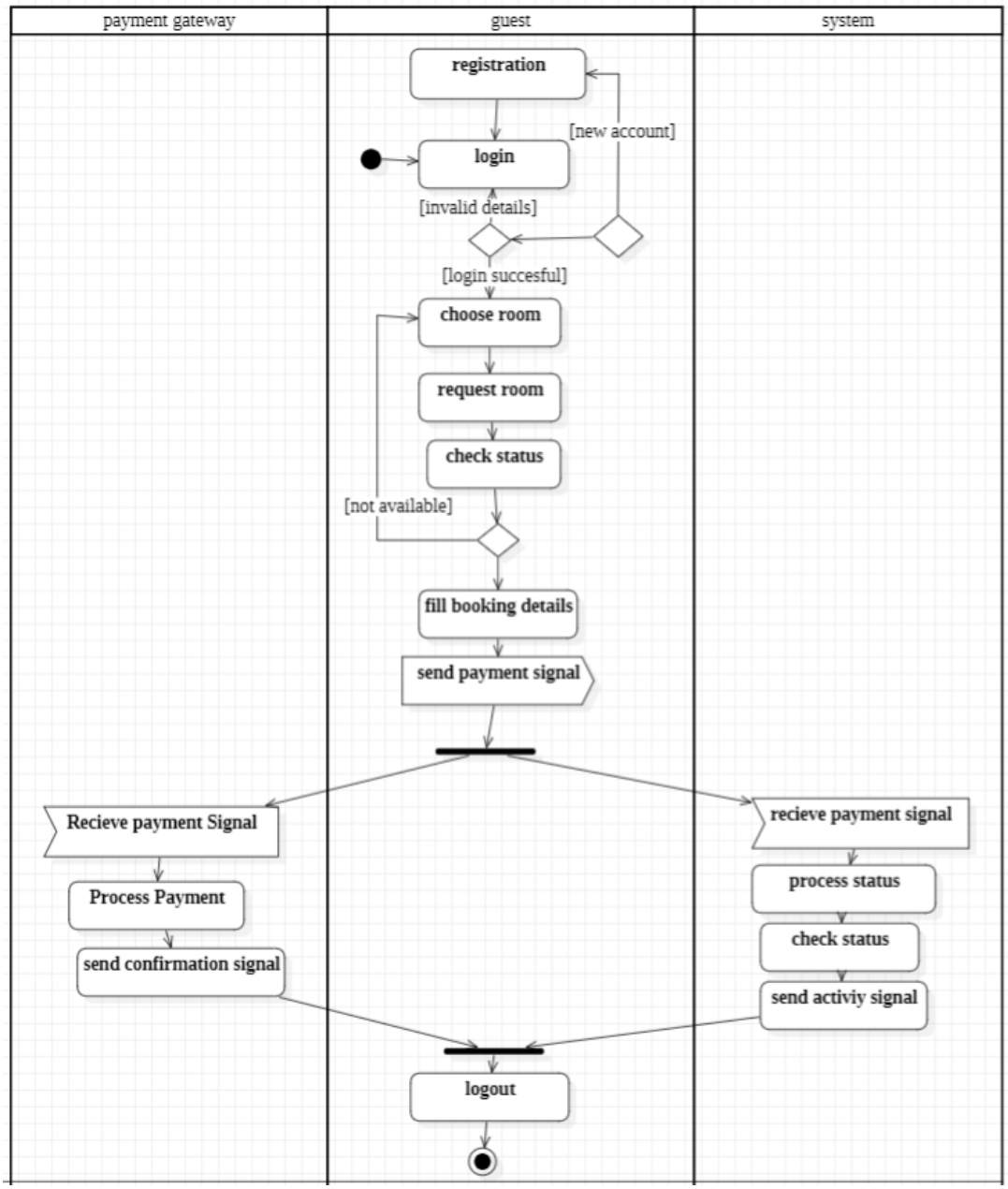


fig 1.7.2

This diagram illustrates the step-by-step process of a hotel booking system using an activity diagram. The main swimlanes are:

- Payment Gateway: Handles payment transactions.
- Guest: Represents a user interacting with the system.
- System: The hotel booking system managing the core logic.

Key Activities:

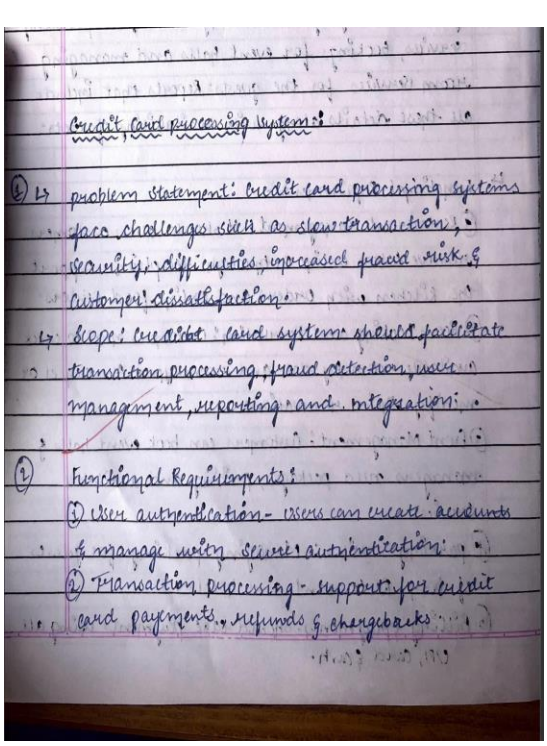
1. Registration: A guest registers for a new account.
2. Login: The user logs in, with checks for valid credentials.
3. Choose and Request Room: Once logged in, the guest chooses and requests a room. If unavailable, the process repeats.
4. Fill Booking Details: After room confirmation, the user fills in booking details.
5. Send Payment Signal: The system processes payment via a gateway.
6. Confirmation: The payment is processed and confirmed.
7. Logout: The guest logs out after completing the process.

2. Credit Card Processing System

2.1 Problem Statement

Design a software system to support the processing of credit card payments for financial institutions, integrating both manual transaction processing and automated fraud detection. Each institution manages its own database for customer records, transactions, and payment history, while the central system handles payment approvals, fraud monitoring, and regulatory compliance. The system should enable real-time payment processing, secure fraud detection, and adherence to industry regulations, ensuring accuracy and scalability. It must handle concurrent transactions, maintain data security, and provide efficient fraud prevention tools while supporting centralized features like compliance tracking and reporting. The cost of the shared system will be distributed based on transaction volume and fraud monitoring usage**.**

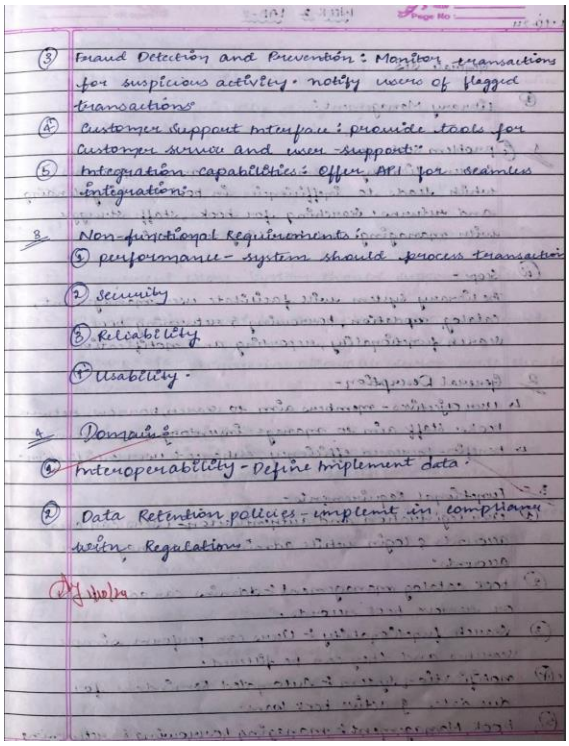
2.2 Software Requirements Specification (SRS) Document



Credit card processing system:

1. problem statement: Credit card processing systems face challenges such as slow transaction, security difficulties, increased fraud risk & customer dissatisfaction.
2. Scope: Credit card system should facilitate transaction processing, fraud detection, user management, reporting and integration.

2. Functional Requirements:
   1. User authentication - users can create accounts & manage with secure authentication.
   2. Transaction processing - support for credit card payments, refunds & chargebacks.



3. Fraud Detection and Prevention: Monitor transactions for suspicious activity. notify users of flagged transactions.
4. Customer Support interface: provide tools for customer service and user support.
5. Integration capabilities: Offer API for seamless integration.

3. Non-functional Requirements:
   1. performance - system should process transaction
   2. security
   3. Reliability
   4. Usability.

4. Domain:
   1. Interoperability - Define implement data.
   2. Data Retention policies - implemit in compliance with Regulations.

2.3 Class Diagram

Simple Class Diagram

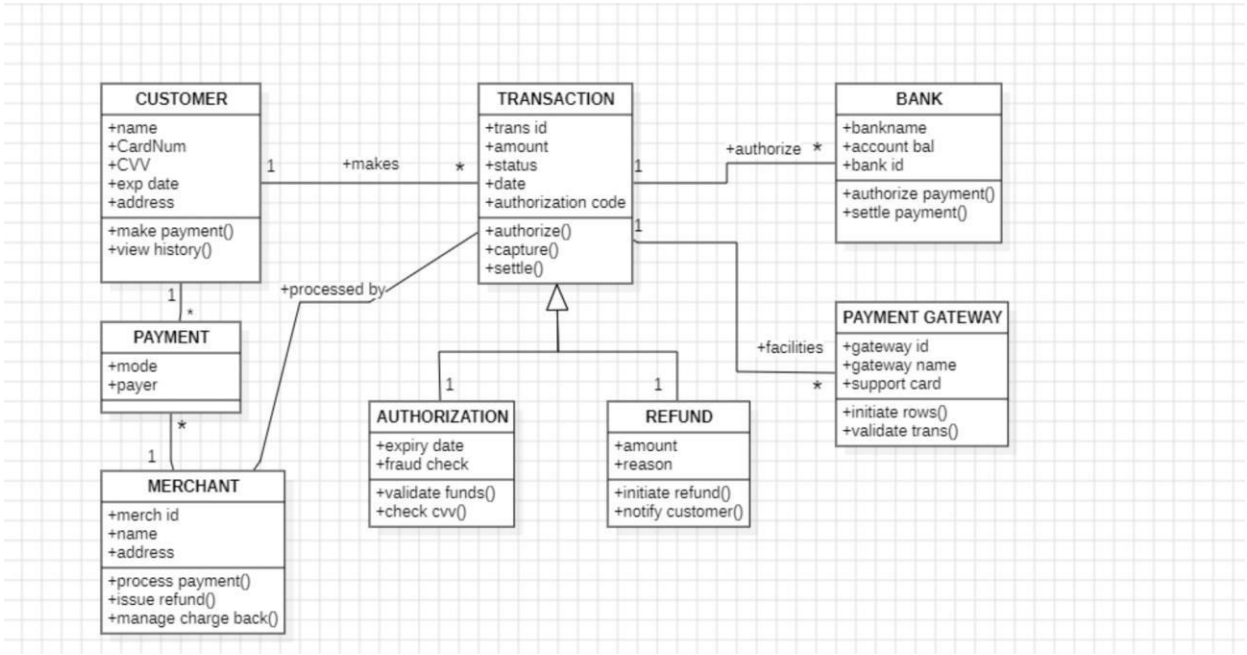


fig 2.3.1

Classes include:

- Customer: Holds personal and card details, with methods for making payments and viewing history.
- Transaction: Manages transaction details, including authorization, capture, and settlement.
- Authorization and Refund: Specialized processes linked to transactions, handling fraud checks and refunds respectively.
- Merchant and Bank: Facilitate transactions and manage accounts.
- PaymentGateway: Validates and initiates transactions.

Relationships:

- Customers initiate transactions, processed by merchants and banks.
- Transactions involve multiple stages, such as authorization and refunds.

Advanced Class Diagram

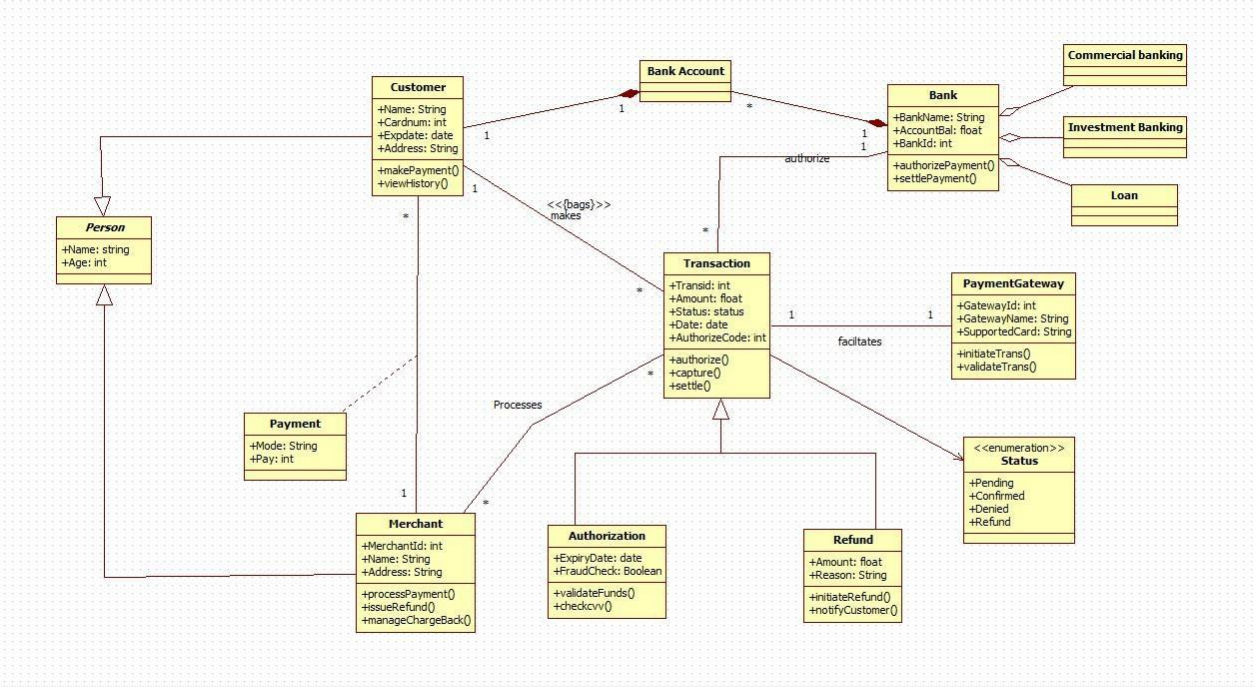


fig 2.3.2

Classes

1. Person: Base class for individuals.
2. Customer:
    - Inherits from Person.
    - Represents bank customers with details like address and card information.
3. Bank: Represents banks with attributes like name and ID.
4. Bank Account:
    - Linked to Bank.
    - Represents accounts with attributes like balance and account holder details.
5. Transaction:
    - Represents financial transactions.
    - Attributes include amount, date, and status.
6. Payment: Represents payment methods used in transactions.
7. Merchant: Represents businesses accepting payments.
8. Payment Gateway: Facilitates payment processing.
9. Authorization: Represents transaction authorizations.
10. Refund: Represents refunds for transactions.

Relationships

1. Inheritance: Customer inherits from Person.
2. Aggregation: Bank Account is aggregated by Bank.
3. Association: Customer is associated with Bank Account.
4. Composition: Transaction is composed of Payment.
5. Dependency: Transaction depends on Payment Gateway.
6. Enumeration: Status defines states for transactions (e.g., pending, confirmed).

2.4 State Diagram

Simple State Diagram

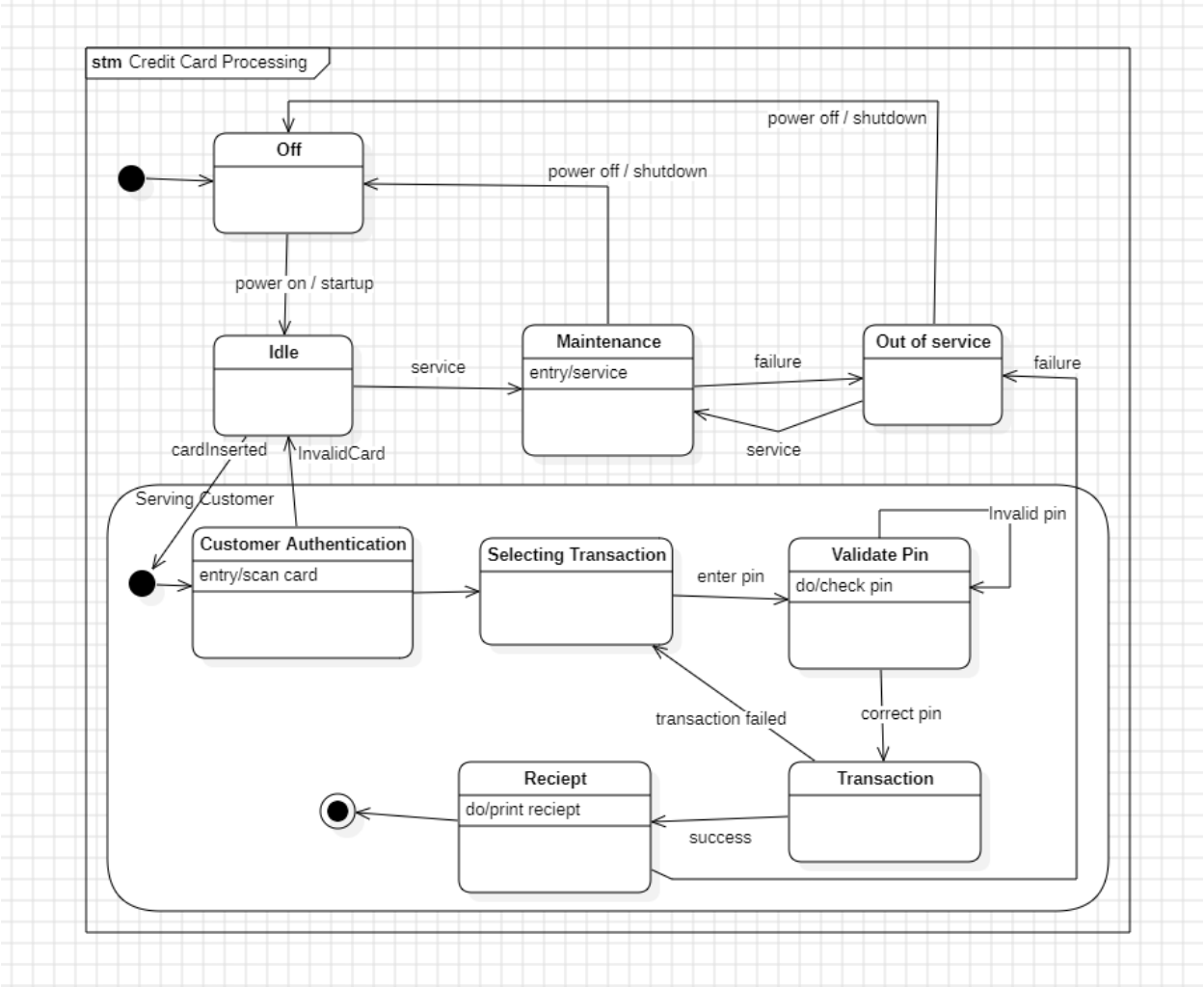


fig 2.4.1

- States include Off, Idle, Maintenance, Out of Service, and Serving Customer.
- Captures activities like powering on/off, card authentication, PIN validation, and transaction processing.
- Includes maintenance and error scenarios, such as invalid cards or failed PIN validation.
- Focuses on ensuring smooth transaction flow and handling failures or service disruptions.

Advanced State Diagram

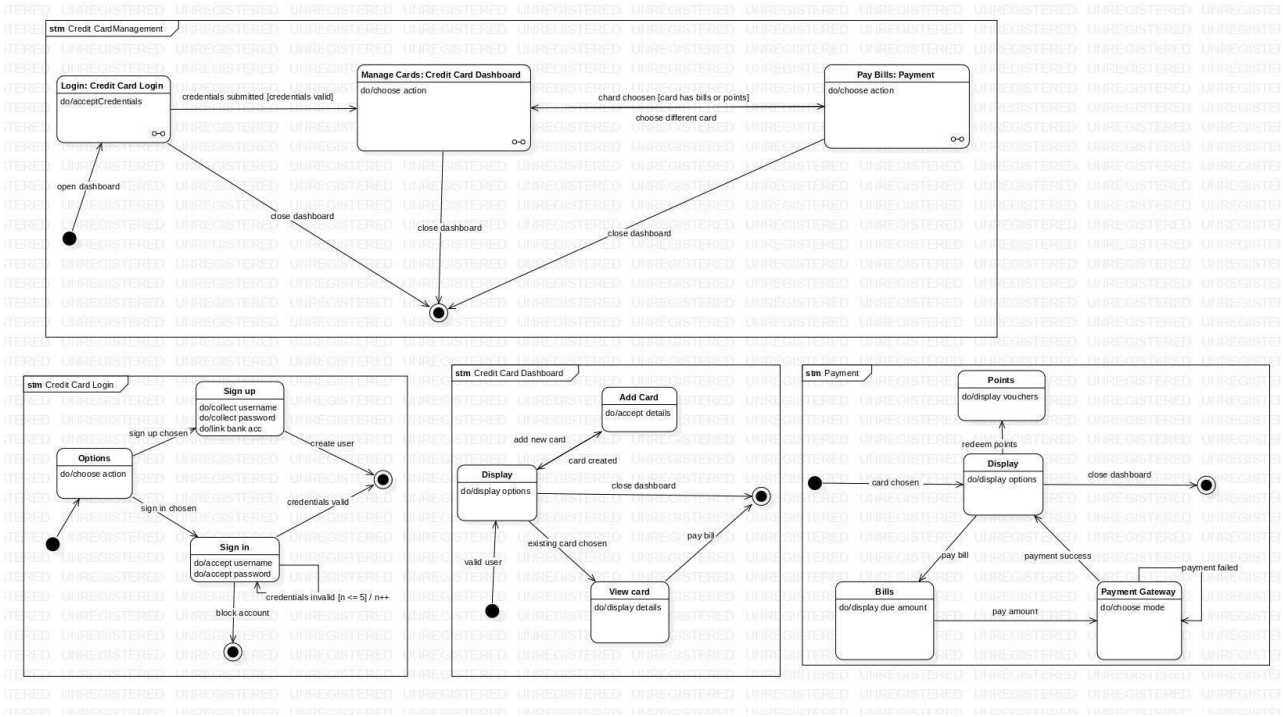


fig 2.4.2

- The diagram captures user actions such as signing up, signing in, managing cards, paying bills, and redeeming points.
- States include Login, Credit Card Dashboard, and Payment.
- Transitions occur based on user inputs, like entering valid credentials, adding a card, or completing a payment.
- Highlights error handling (e.g., invalid login attempts and blocking accounts after a certain limit).

2.5 Use case diagram

Simple Use Case Diagram

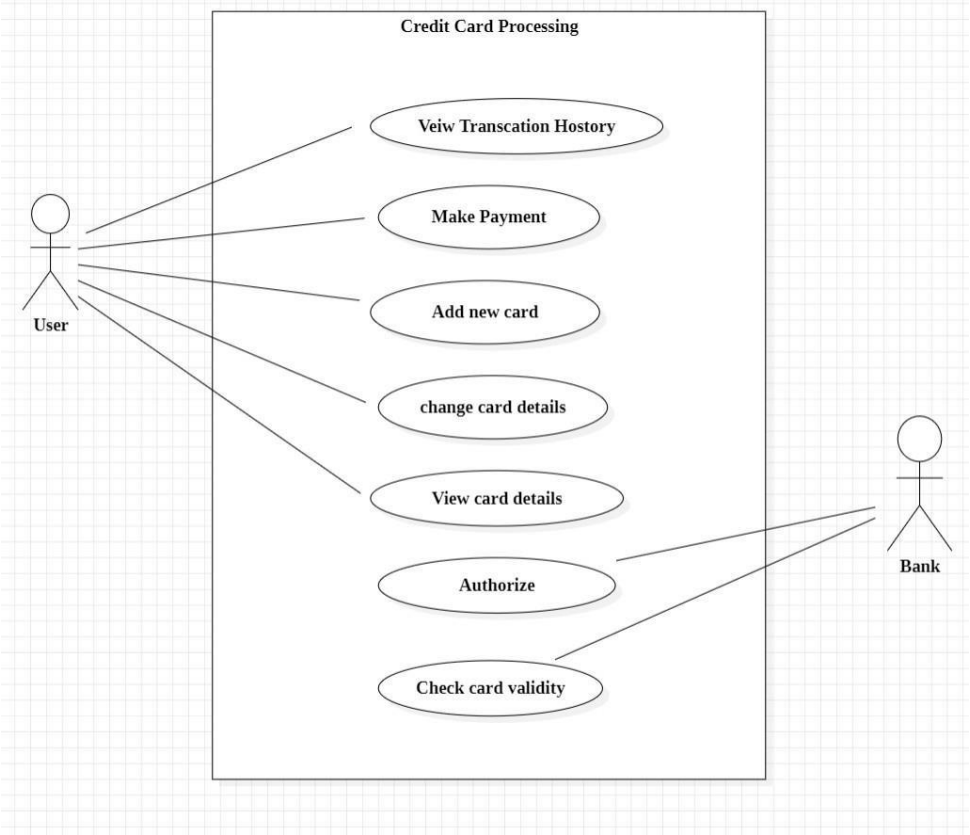


fig 2.5.1

Description: This diagram shows the primary use cases for a credit card processing system, focusing on the interaction between users (e.g., User and Bank) and the system.

Key Use Cases:

- View Transaction History: Allows the user to review past transactions.
- Make Payment: Facilitates bill payments via the credit card system.
- Add New Card: Enables users to add a new credit card to their profile.
- Change Card Details: Allows modifications to existing credit card details.
- View Card Details: Lets users access information about their cards.
- Authorize: Interaction with the bank to approve transactions.
- Check Card Validity: Validates the card's status with the bank.

Advanced Use Case Diagram

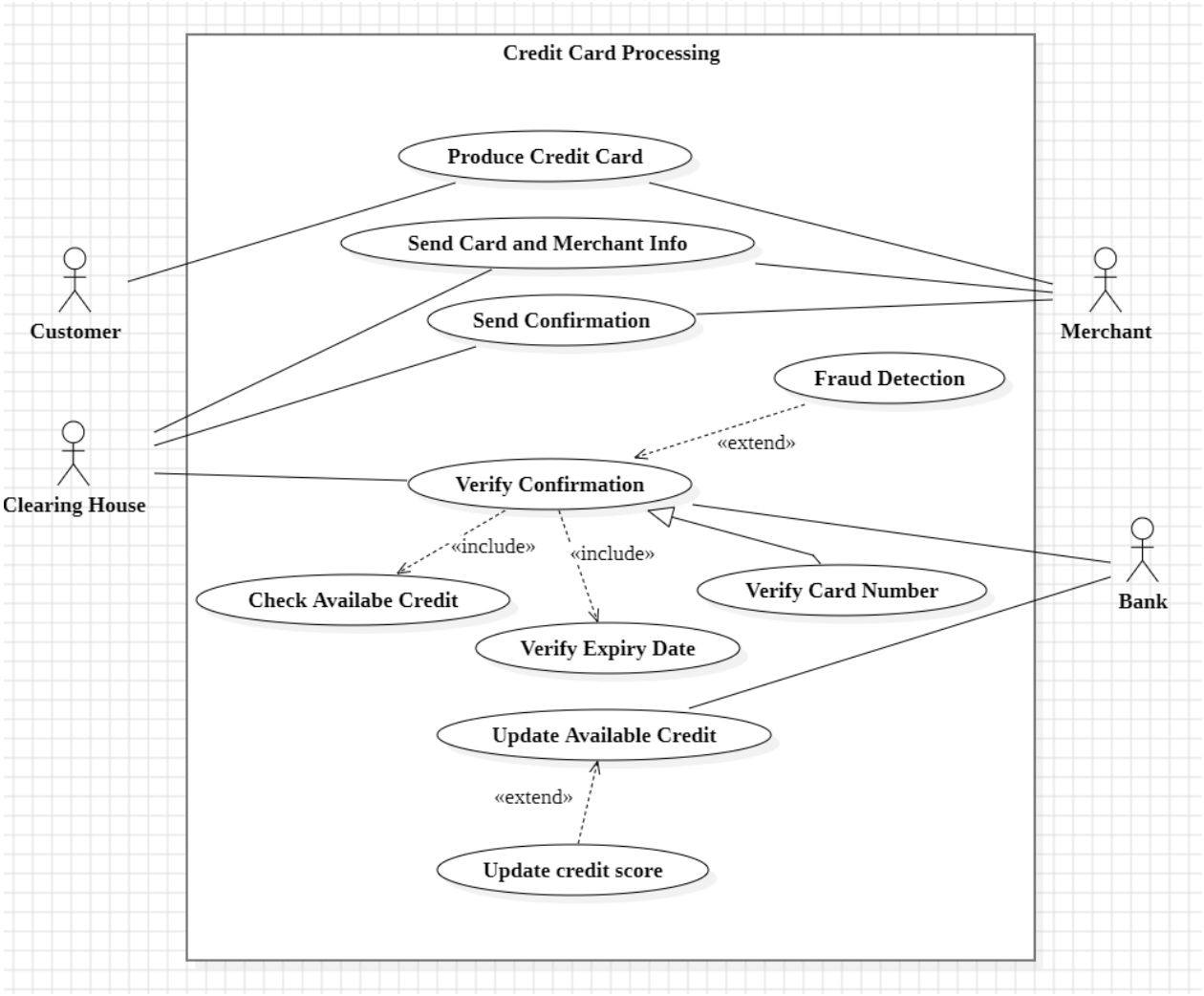


fig 2.5.2

The advanced use case diagram expands on the simple version by providing a more detailed and nuanced view of the credit card processing system. It introduces additional use cases, such as:

- Customer Authentication: Ensures the user is authorized to access and use the credit card.
- Fraud Prevention: Detects and mitigates fraudulent activities during transactions.
- Enhanced Verification: Includes steps like advanced card number checks, dynamic credit validation, and expiry verification.
- Communication with External Systems: Highlights interactions with the Clearing House and Bank to validate transactions and update the system.

2.6 Sequence Diagram

Simple Sequence Diagram

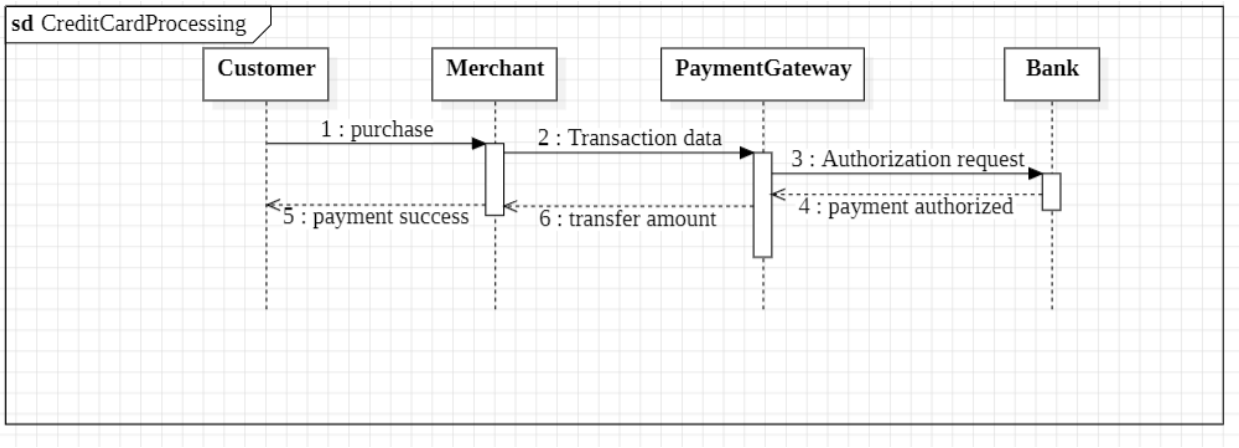


fig 2.6.1

Description: This diagram represents the sequence of interactions among the system components during a credit card transaction.

Steps:

1. Customer initiates a purchase request with the Merchant.
2. The Merchant sends transaction data to the Payment Gateway.
3. The Payment Gateway requests authorization from the Bank.
4. The Bank validates the transaction and sends back authorization.
5. Payment Gateway notifies the Merchant of a successful payment.
6. The Merchant confirms the purchase to the Customer and transfers the amount to the bank.

Advanced Sequence Diagram

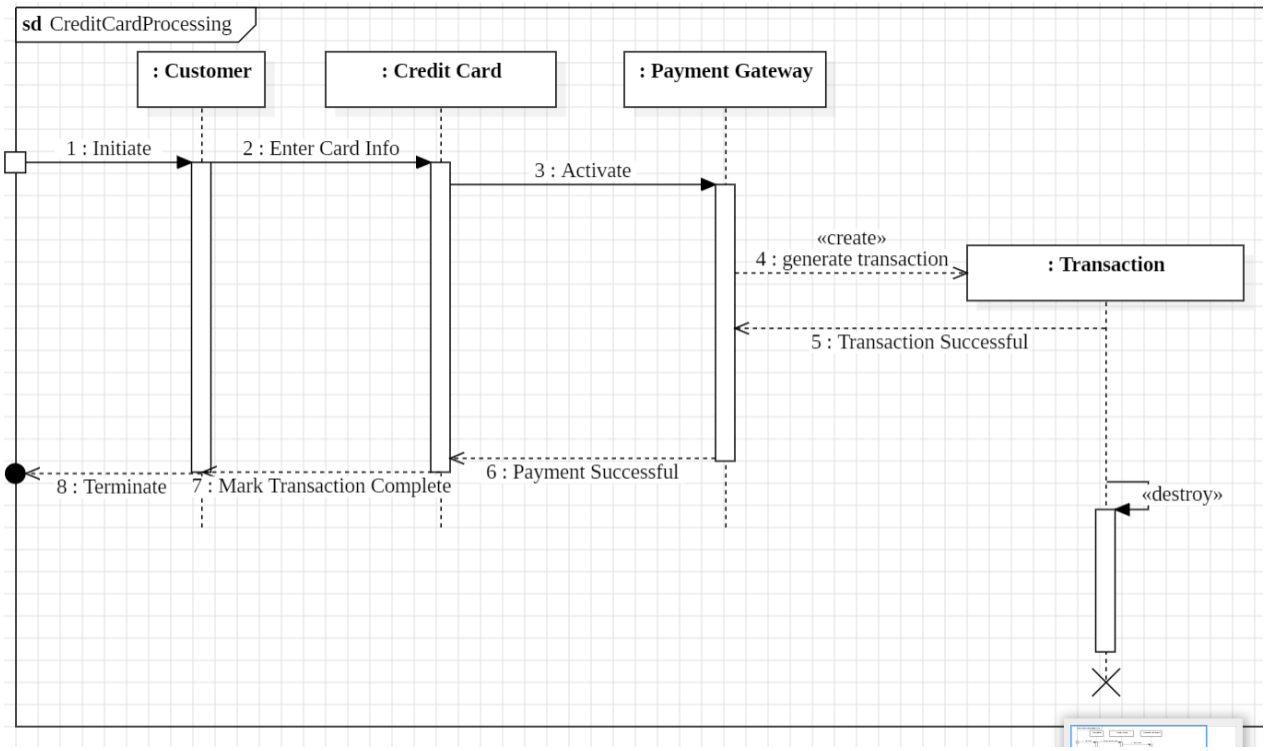


fig 2.6.2

1. Customer Initiates Transaction: The process starts when the customer enters their credit card details.
2. Authentication & Validation: The system authenticates the customer's credentials and validates the credit card details, including expiry date and available balance.
3. Fraud Detection Check: The system performs fraud detection by analyzing patterns or using external fraud detection services.
4. Generate Transaction: Once validated, the payment gateway creates a transaction object and sends it for processing.
5. External Communication: The payment gateway interacts with the bank or clearing house to authorize the payment.
6. Transaction Completion: Upon approval, the system marks the transaction as successful and notifies the customer and merchant.

2.7 Activity Diagram

Simple Activity Diagram

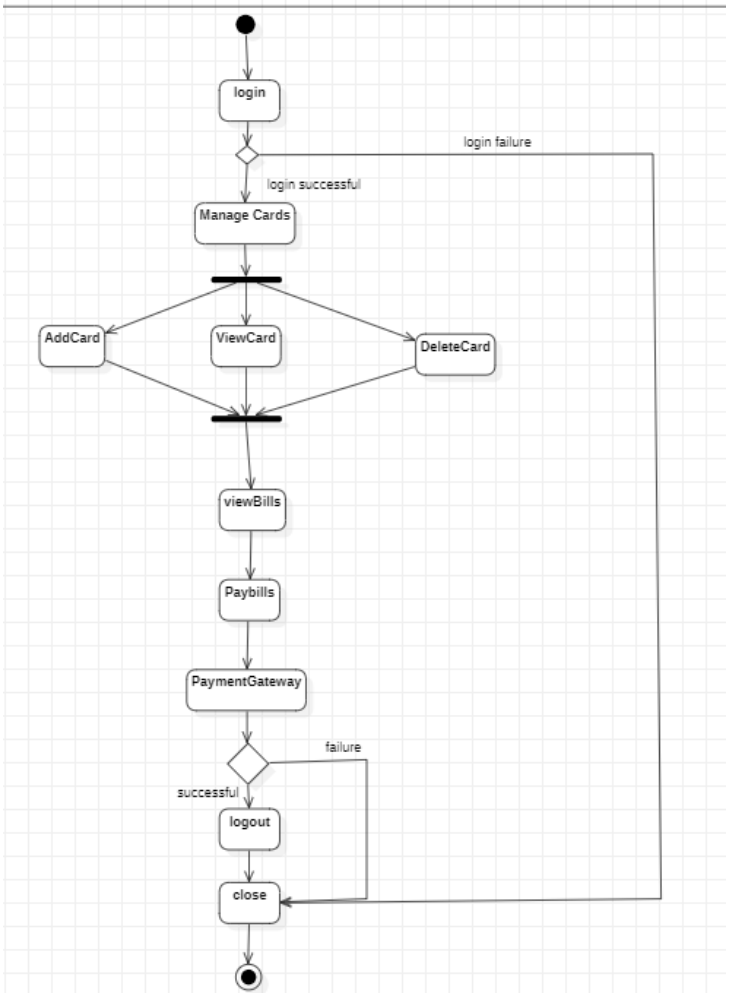


fig 2.7.1

Description: This diagram illustrates the flow of activities in the credit card processing system, from logging in to managing cards and paying bills.

Key Activities:

- Login: Starts the process with either successful authentication or failure.
- Manage Cards: Includes adding, viewing, or deleting credit cards.
- View Bills and Pay Bills: Users can check outstanding bills and make payments.
- Payment Gateway: Handles the payment process, with success leading to logout and failure prompting a retry.
- Logout: Ends the session, closing the interaction with the system.

Advanced Activity Diagram

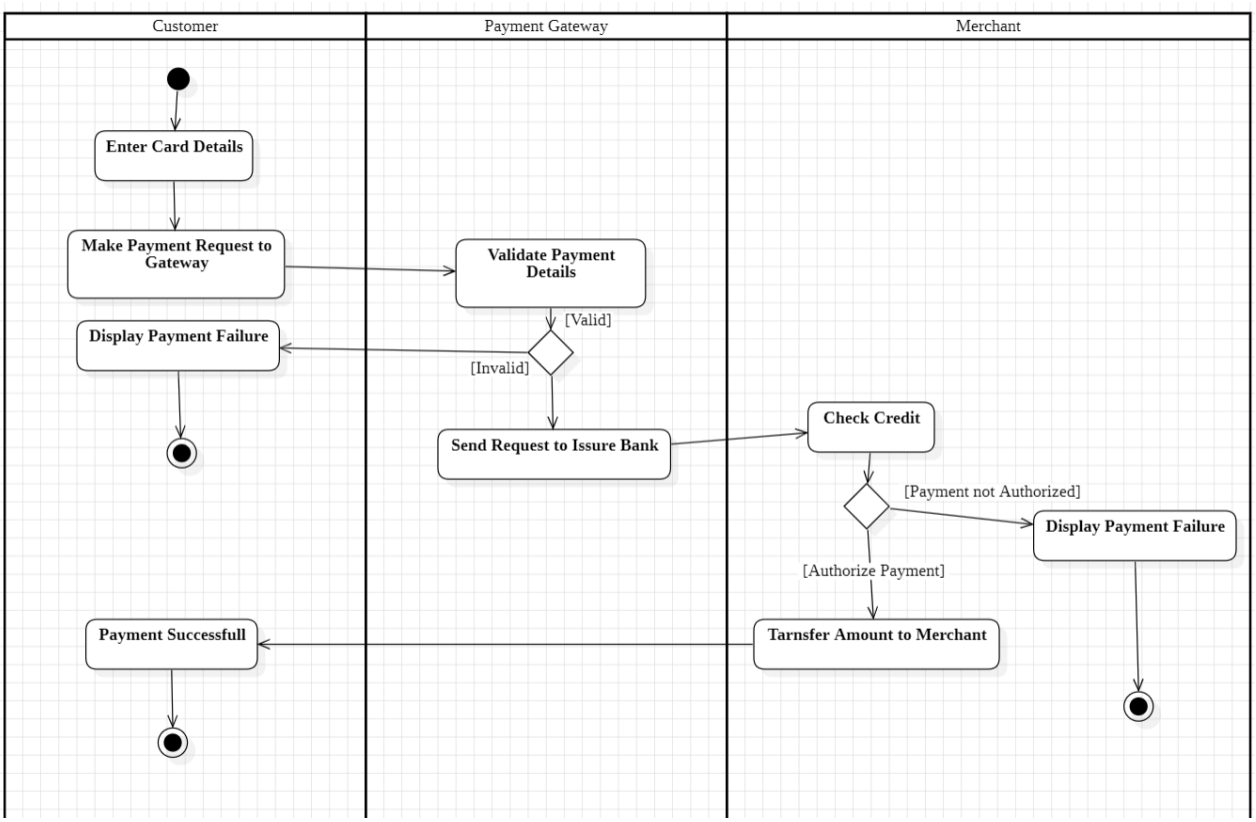


fig 2.7.2

This activity diagram containing swimlanes for Customer, Payment Gateway and Merchant , illustrates the payment process in the credit card system:

- Customer: Enters card details and makes a payment request to the payment gateway.
- Payment Gateway: Validates the card details and sends a request to the issuing bank.
- Issuing Bank:
    - Checks the credit status of the customer.
    - If payment is authorized, transfers the amount to the merchant.
    - If payment fails (invalid card details or insufficient credit), failure is displayed to the customer.
- Outcome:
    - Payment success: Funds are transferred to the merchant.
    - Payment failure: Relevant failure message is shown to the customer.

3. Library Management System

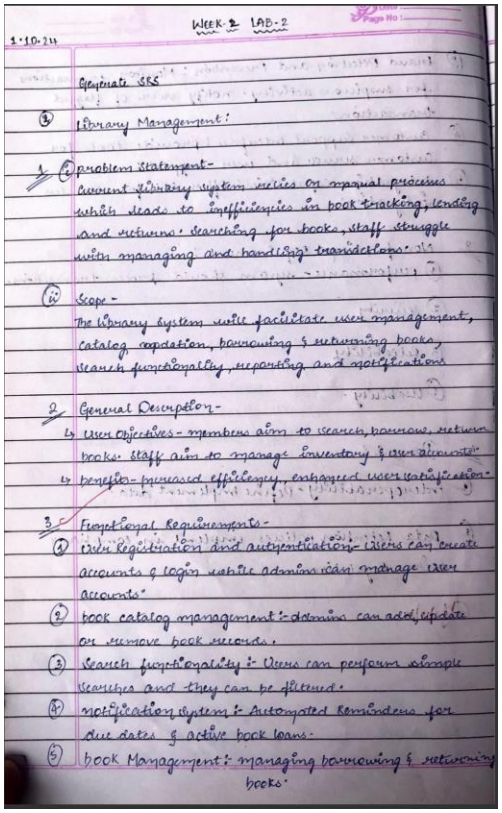
3.1 Problem Statement

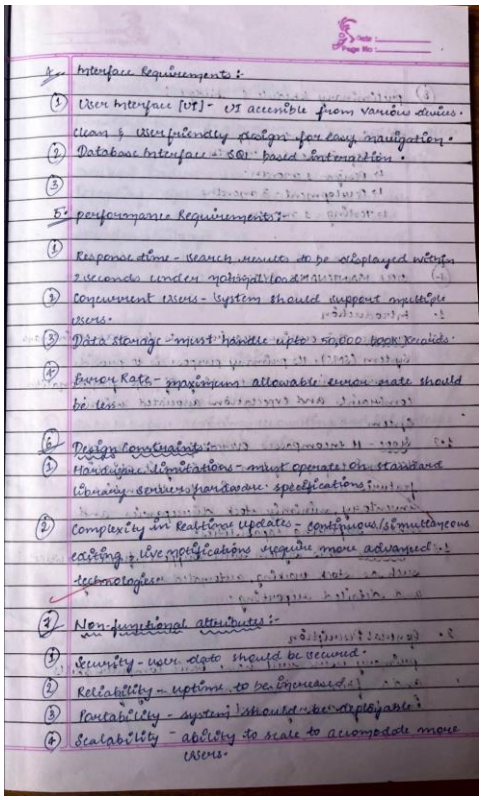
The state diagram illustrates the workflow of an online food delivery system, starting with the Order Placement state, where customers select items from the menu and confirm their order. In the Order Processing phase, the system verifies the availability of items in the restaurant's inventory and processes the payment through integrated payment gateways, ensuring a seamless transaction. The order details are then passed to the Kitchen Preparation state, where items are prepared according to the customer's specifications.

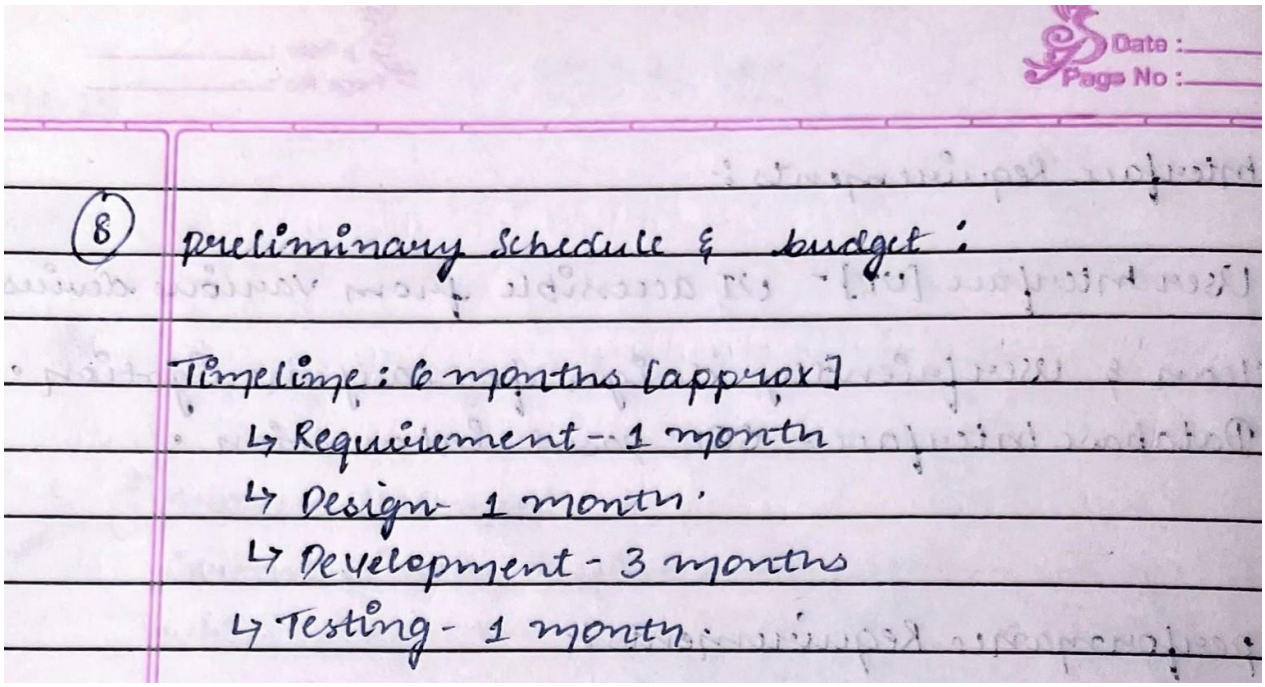
Once prepared, the order moves to the Packaging and Dispatch state, where the food is packed and assigned to a delivery agent. During the Delivery Tracking state, customers can track the real-time location of their order using GPS integration. Finally, the process transitions to the Order Completion state, where the delivery is confirmed, and the customer is prompted to provide feedback.

The system incorporates advanced features such as real-time inventory checks, automated delivery assignment, and customer feedback integration to enhance the user experience. The state diagram represents the entire workflow as a state machine, with nested states like Payment Verification, Order Assignment, and Delivery Tracking, showcasing the hierarchical and modular nature of the system. Additionally, concurrent states, such as inventory checks and payment processing in the Order Processing phase, ensure efficiency and reliability.

3.2 Software Requirements Specification (SRS) Document







3.3 Class Diagram

Simple Class Diagram

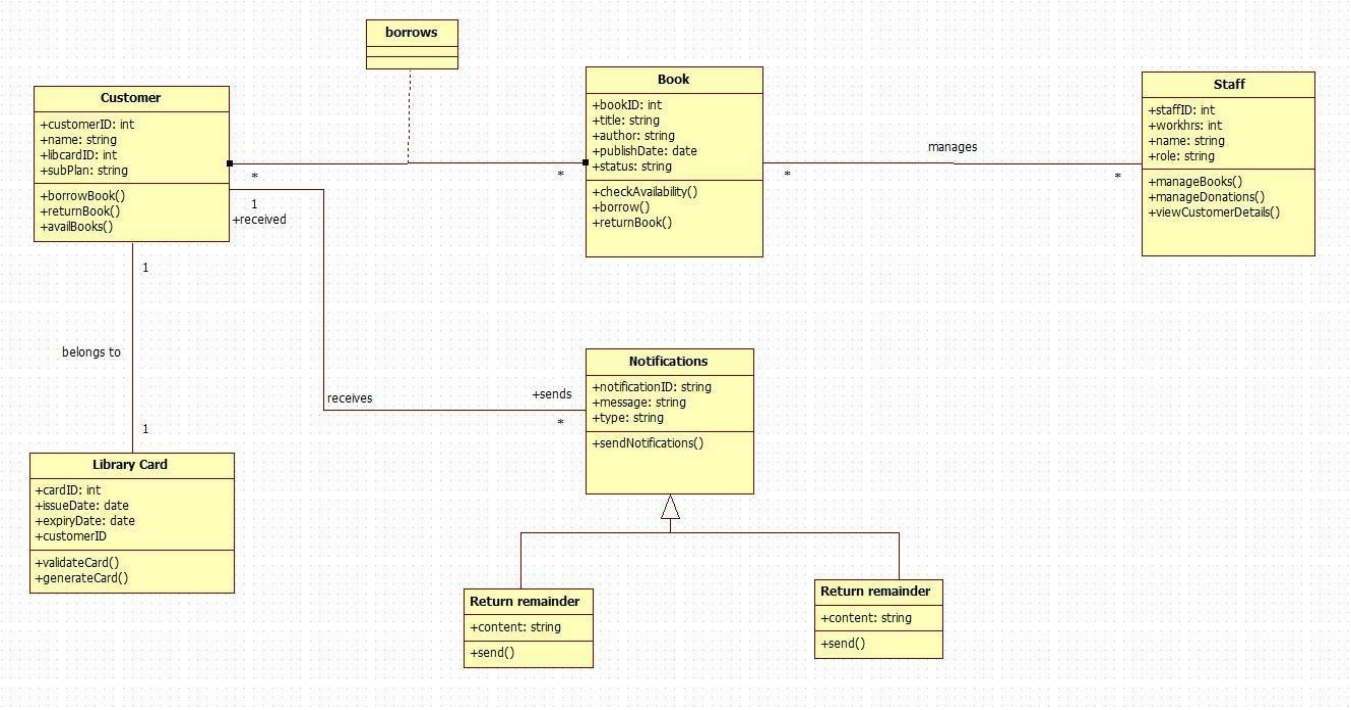


fig 3.3.1

- Person Class: A generalized superclass for various roles, such as customers and staff.

- Customer Class: Includes attributes like customer ID and library card and methods for borrowing, returning, and viewing available books.
- Book Class: Details attributes like title, author, and status and methods for checking availability and borrowing/returning books.
- Library Card Class: Associates with customers and tracks issued/expiry dates, with methods for validation.
- Staff Class: Manages books and oversees customer activities.
- Notifications Class: Used for communication with users, e.g., reminders or availability updates.

Advanced Class Diagram

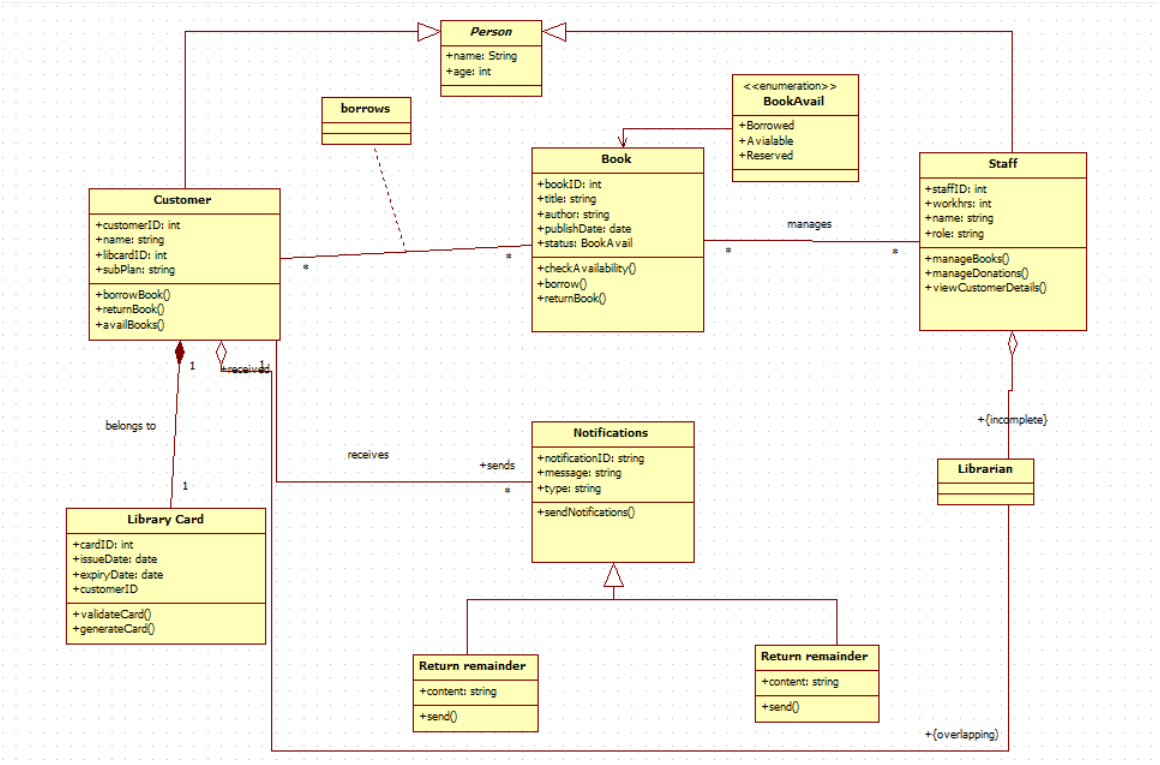


fig 3.3.2

This expanded class diagram includes more explicit connections between entities:

- Borrow Relationship: Connects customers and books, detailing the process flow.
- Notifications System: Tracks and sends messages regarding books or due dates.
- Return Reminders: Specializes in follow-ups for borrowed books, ensuring timely returns.

3.4 State Diagram

Simple State Diagram

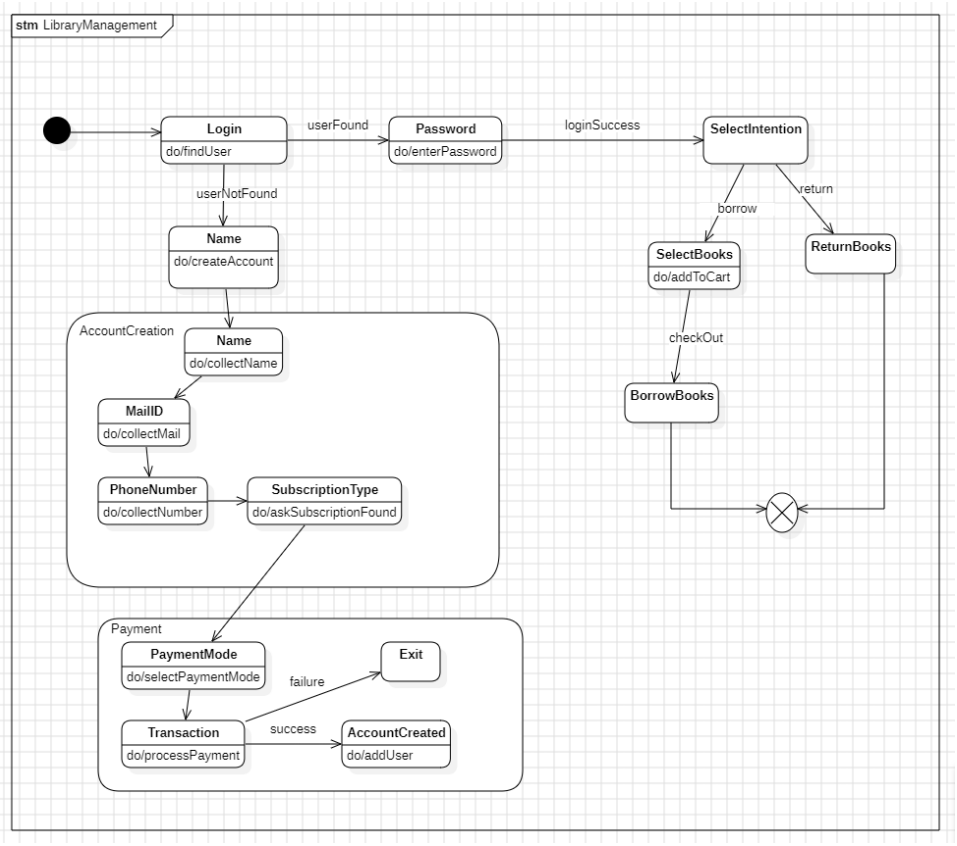


fig 3.4.1

This diagram describes the various states and transitions within a library management system.
Key highlights:

- Login Process: Users provide credentials to enter the system.
- Searching State: Users can search for books, which involves inputting details, checking availability, and displaying results.
- Issuing State: If a book is found and requested, the system updates the database and issues the book.
- Updating Issue Card: Maintains the record of issued books for each user.
- Logout Process: Indicates the user's exit from the system. Sub-state diagrams (e.g., Searching) further detail inner workflows like inputting book details, checking the database, and handling search failures.

Advanced State Diagram

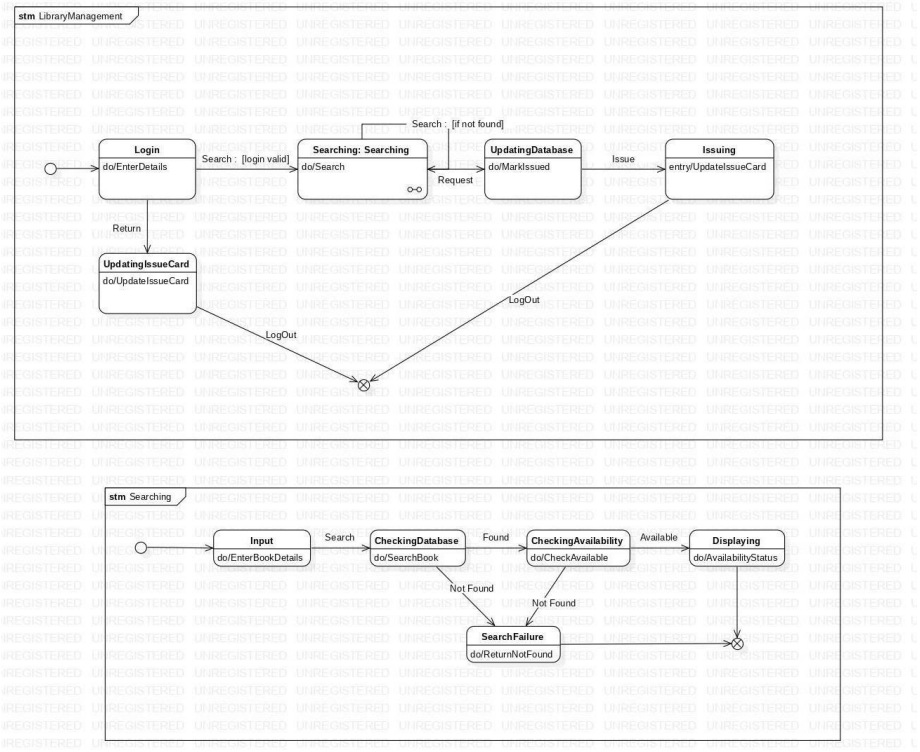


fig 3.4.2

This diagram elaborates on additional functionalities like:

- Account Creation: For new users, the system collects personal details (name, email, phone number) and subscription type before account activation.
- Transaction Handling: Payment modes and subscription processing are covered here, with success leading to account creation.
- Book Management: After logging in, users can select intentions like borrowing or returning books, with distinct workflows for both actions.

3.5 Use case diagram

Simple Use Case Diagram

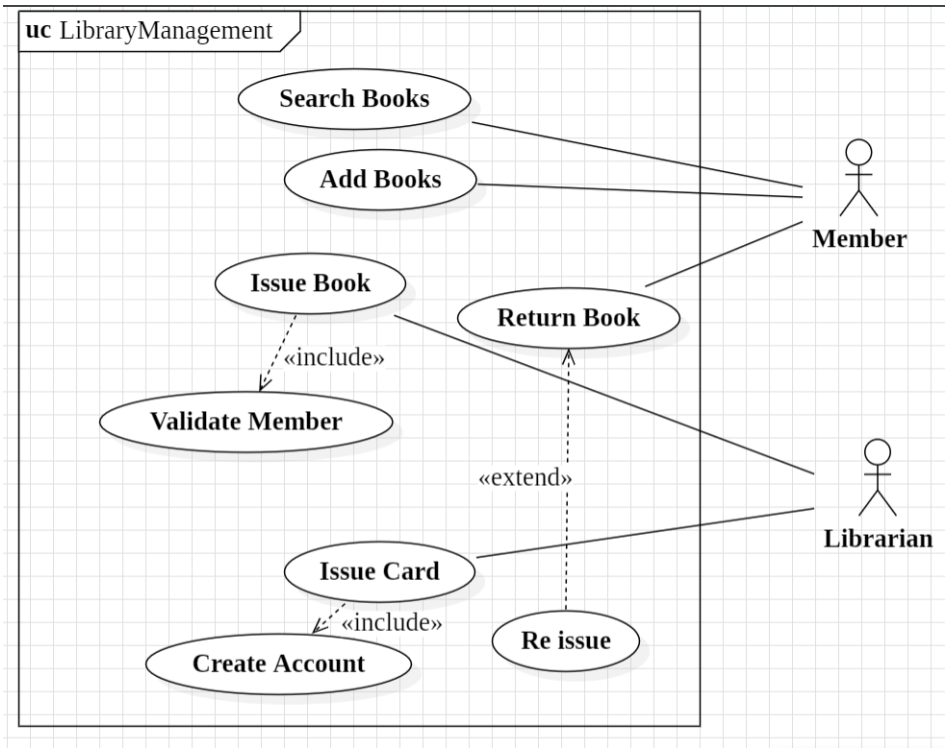


fig 3.5.1

The use case diagram represents the functionalities of a library management system. It highlights the interactions between two primary actors, Members and Librarians, and the system. Members can search for books, return books, and request book issuance, which includes validating membership. Librarians manage tasks such as adding books, issuing cards, and reissuing books, which extends the book-return functionality. The diagram uses relationships like <<include>> and <<extend>> to depict dependencies between processes, ensuring clarity in the flow of operations.

Advanced Use Case Diagram

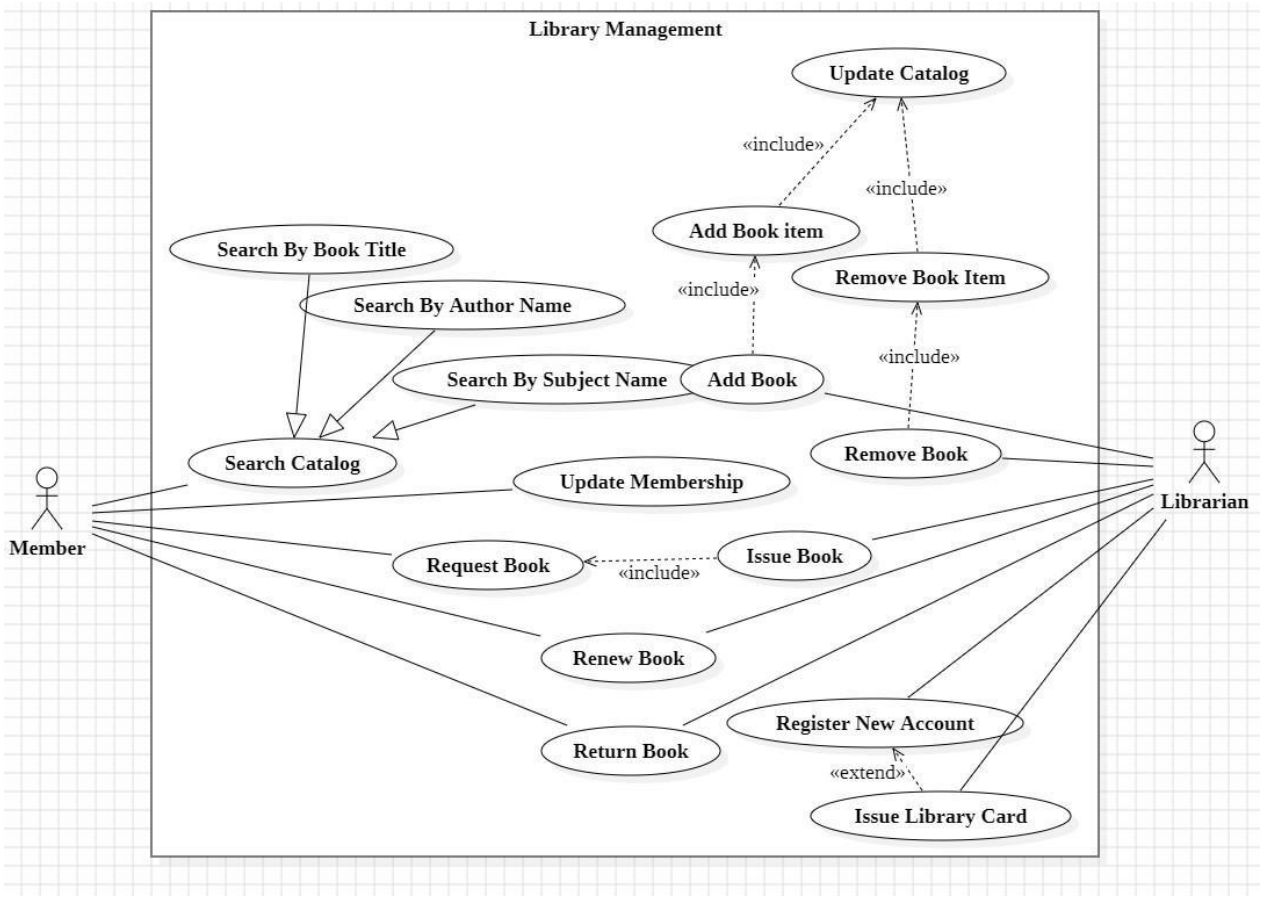


fig 3.5.2

The use case diagram models the interactions between users (actors) and the Library Management System.

- Actors: The main actors are the Member and Librarian.
- Use Cases: It includes actions such as searching the catalog (by title, author, or subject), requesting books, returning books, and updating membership for members. For librarians, it includes issuing books, updating the catalog (adding/removing books), and managing user accounts.
- Relationships: Dependencies like include and extend define additional functionalities and shared responsibilities between use cases.

3.6 Sequence Diagram

Simple Sequence Diagram

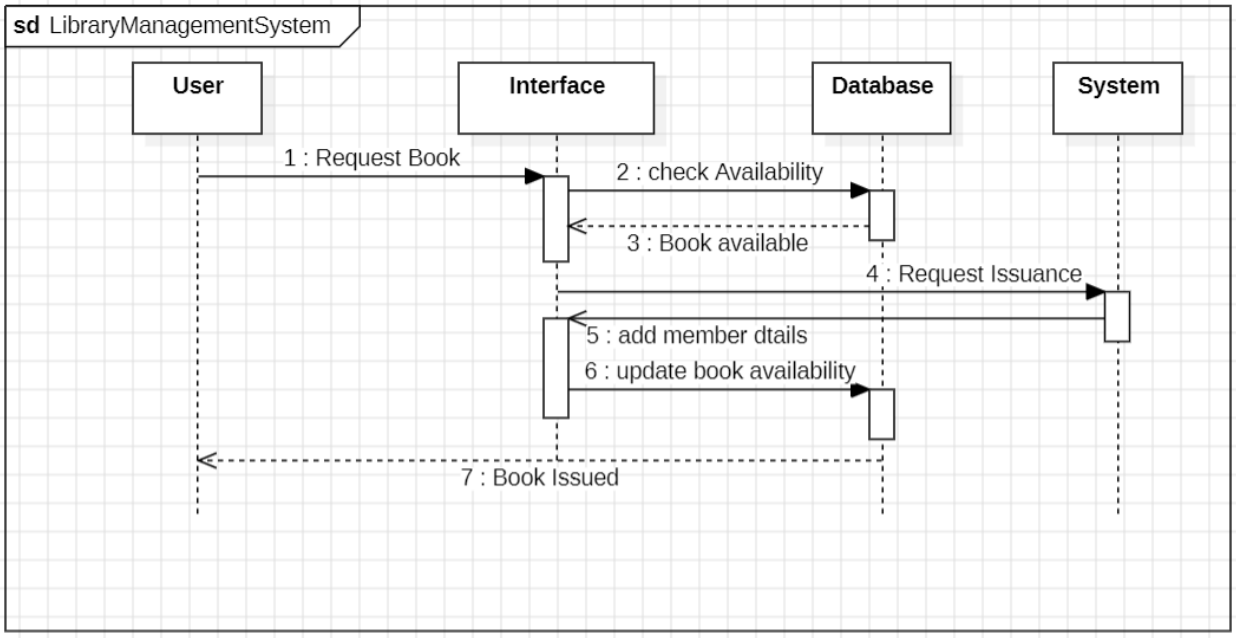


fig 3.6.1

The sequence diagram illustrates the dynamic interaction among components of the library management system during a book issuance process. The flow begins with the user requesting a book through the interface. The system checks the book's availability in the database, and if available, the issuance request is processed. Member details are added, the database is updated to reflect the new status, and the book is issued to the user. This step-by-step flow highlights the real-time communication between the user, interface, database, and system components.

Advanced Sequence Diagram

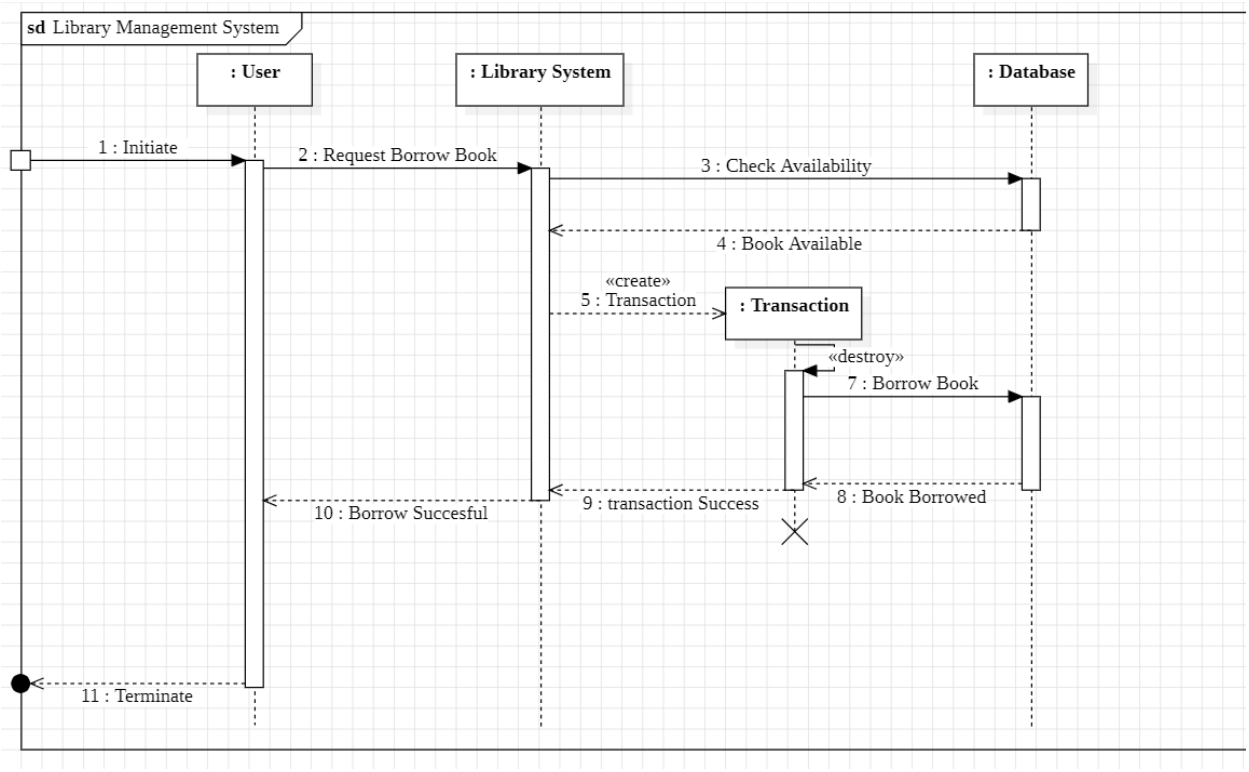


fig 3.6.2

This diagram illustrates the sequence of interactions during the book borrowing process.

- Lifelines: The key participants are the User, Library System, Database, and a temporary Transaction object.
- Flow: The user initiates a book borrow request, which is processed by the library system by checking the database for availability. If the book is available, a transaction is created, the book is issued, and the transaction ends.
- Messages: Shows synchronous and asynchronous messages exchanged among components, ensuring smooth operation.

3.7 Activity Diagram

Simple Activity Diagram

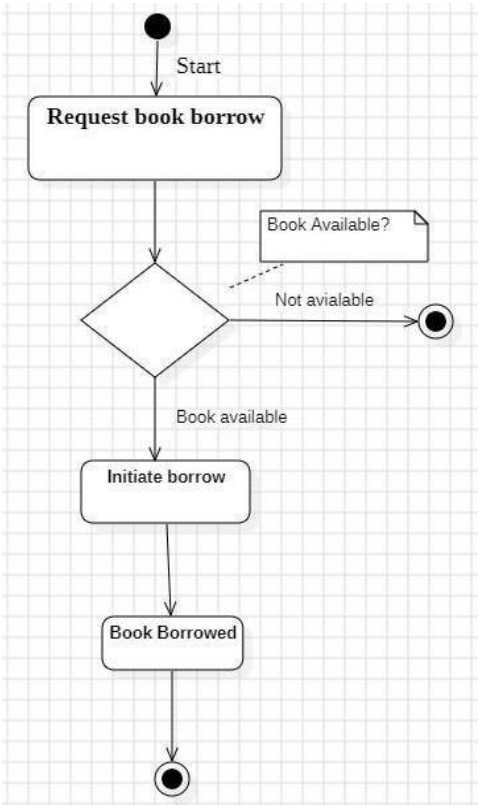


fig 3.7.1

The activity diagram outlines the workflow of borrowing a book from the library. It starts with the user requesting to borrow a book. A decision node checks the book's availability. If the book is not available, the process ends; otherwise, the borrowing process is initiated. Upon successful initiation, the book is borrowed, and the process concludes. This diagram emphasizes the conditional logic and sequential steps involved in the borrowing process.

Advanced Activity Diagram

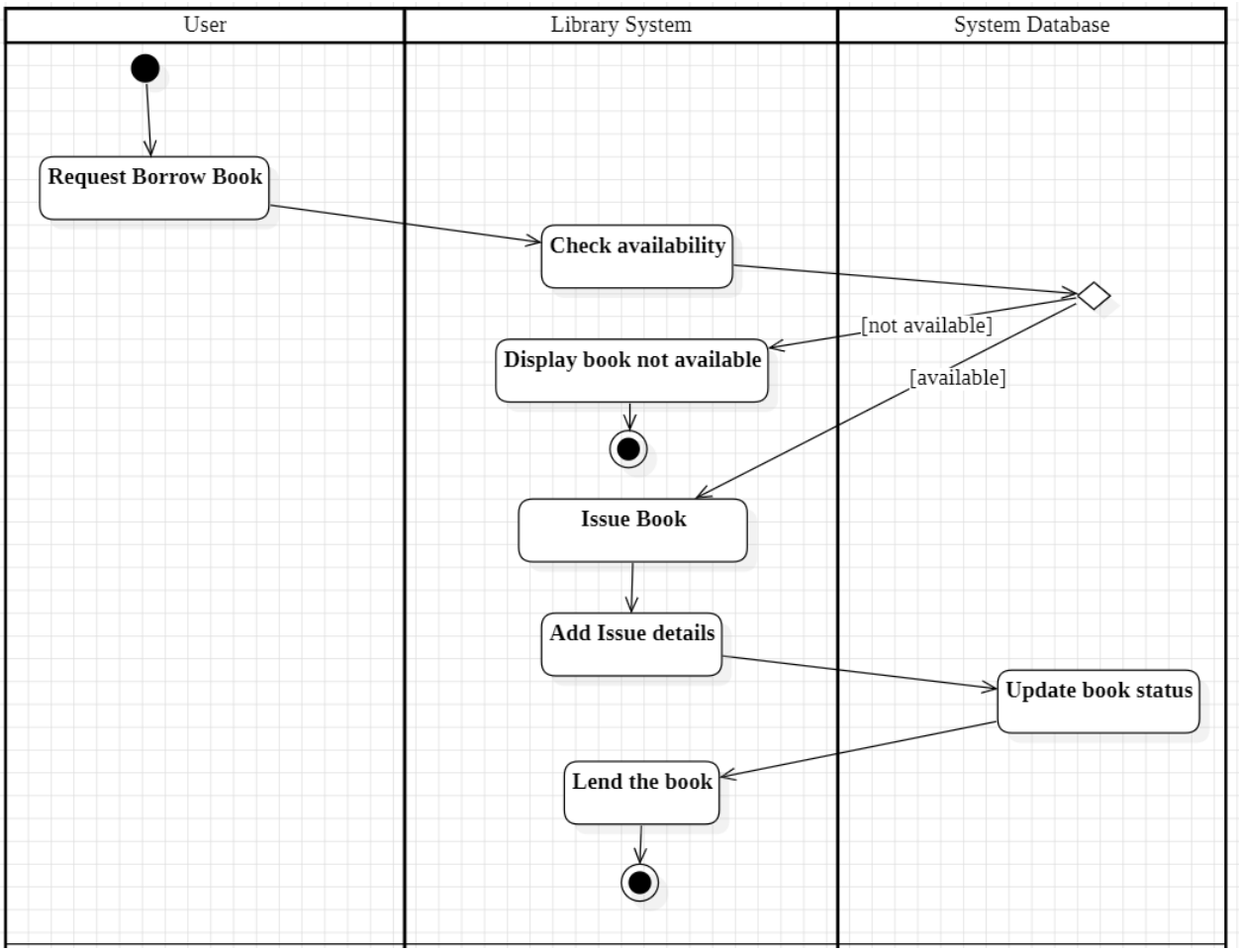


fig 3.7.2

The activity diagram represents the workflow for borrowing a book from the library.

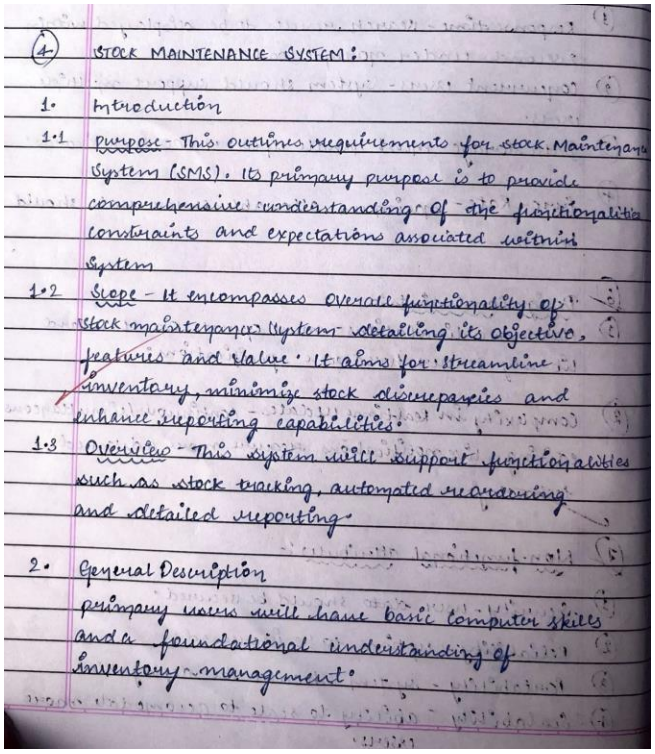
- Flow: The process starts when a user requests to borrow a book.
- Steps: The system checks the availability of the book in the database. If unavailable, a "book not available" message is displayed. If available, the book is issued, details are recorded, and the book is handed to the user.
- Swimlanes: The activities are divided among three entities: User, Library System, and System Database, showing their responsibilities.

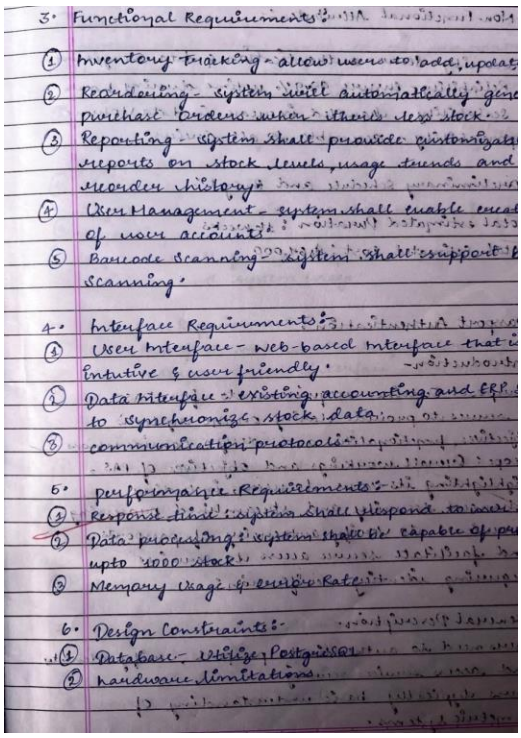
4. Stock Maintenance System

2.1 Problem Statement

The Stock Maintenance System is designed to efficiently manage and track inventory levels, orders, sales, and deliveries, providing functionalities such as adding, updating, and removing stock items, generating inventory and sales reports, managing suppliers, and issuing low-stock notifications. Catering to inventory staff, managers, and suppliers, the system ensures real-time stock updates, intuitive interfaces, and role-based access control. Key features include automated low-stock alerts, analytical dashboards, supplier management, and seamless integration with existing ERP systems. Developed using a microservices architecture, the system emphasizes scalability, usability, and security, supporting at least 10,000 stock items while ensuring data encryption and rapid processing of updates within 2 seconds. With a projected timeline of 5 months and a budget of ₹35 lakhs, the project covers all stages from design to deployment, ensuring a robust and user-friendly solution for effective inventory management.

2.2 Software Requirements Specification (SRS) Document





7. Non-Functional Attributes :-

1. Security - Secure authentication & data encryption.
2. Reliability - ensure 99.9% uptime during hours.
3. Scalability - handled increase in data volume.
4. portability - accessible to all devices.

8. preliminary Schedule and Budget :-

Total estimated Duration : 16 weeks

Total Estimated cost : Rs 64,000

4.3 Class Diagram

Simple Class Diagram

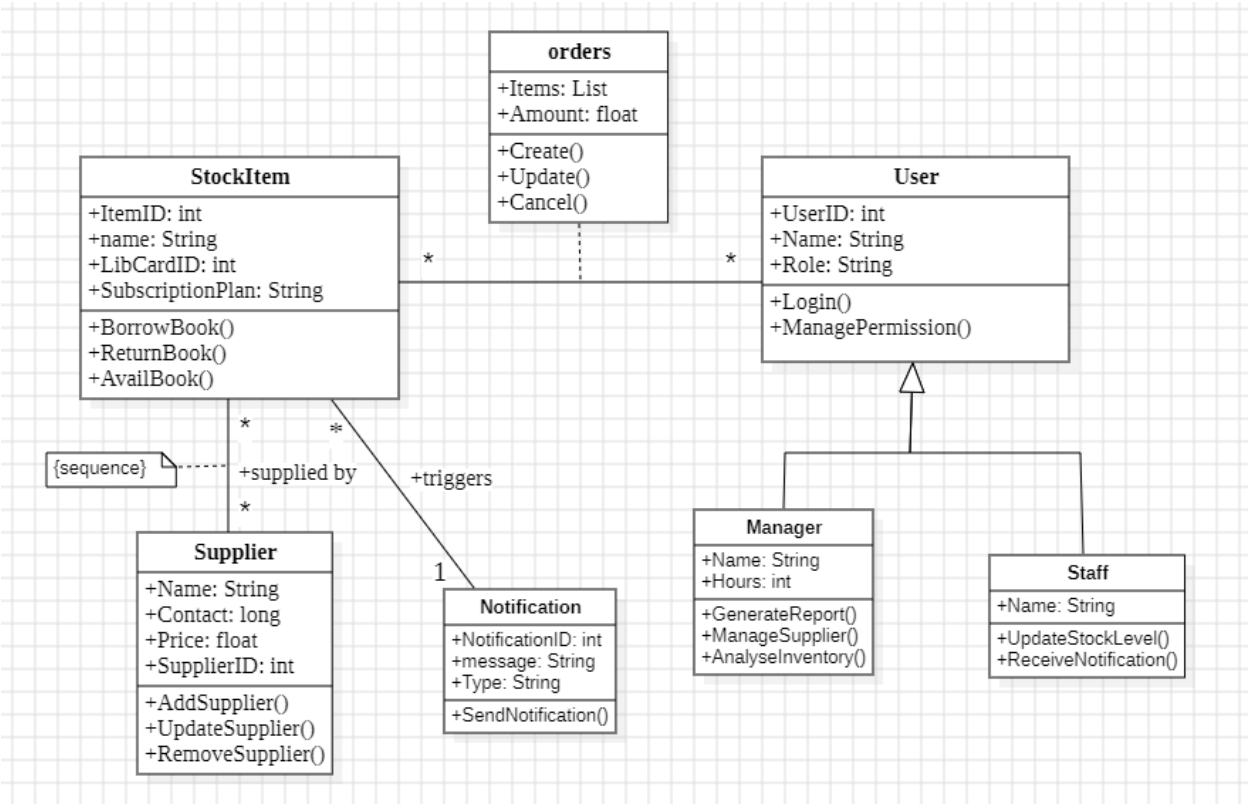


fig 4.3.1

Actors:

- User: Includes specific roles such as Manager and Staff.

Use Cases:

- Login, ManagePermission: General operations performed by users.
- AddSupplier, UpdateSupplier, RemoveSupplier: Actions related to supplier management.
- GenerateReport, AnalyzeInventory: Manager-specific activities.
- ReceiveNotification, UpdateStockLevel: Actions primarily performed by staff.

Relationships:

- Links between actors and their respective use cases.

Advanced Class Diagram

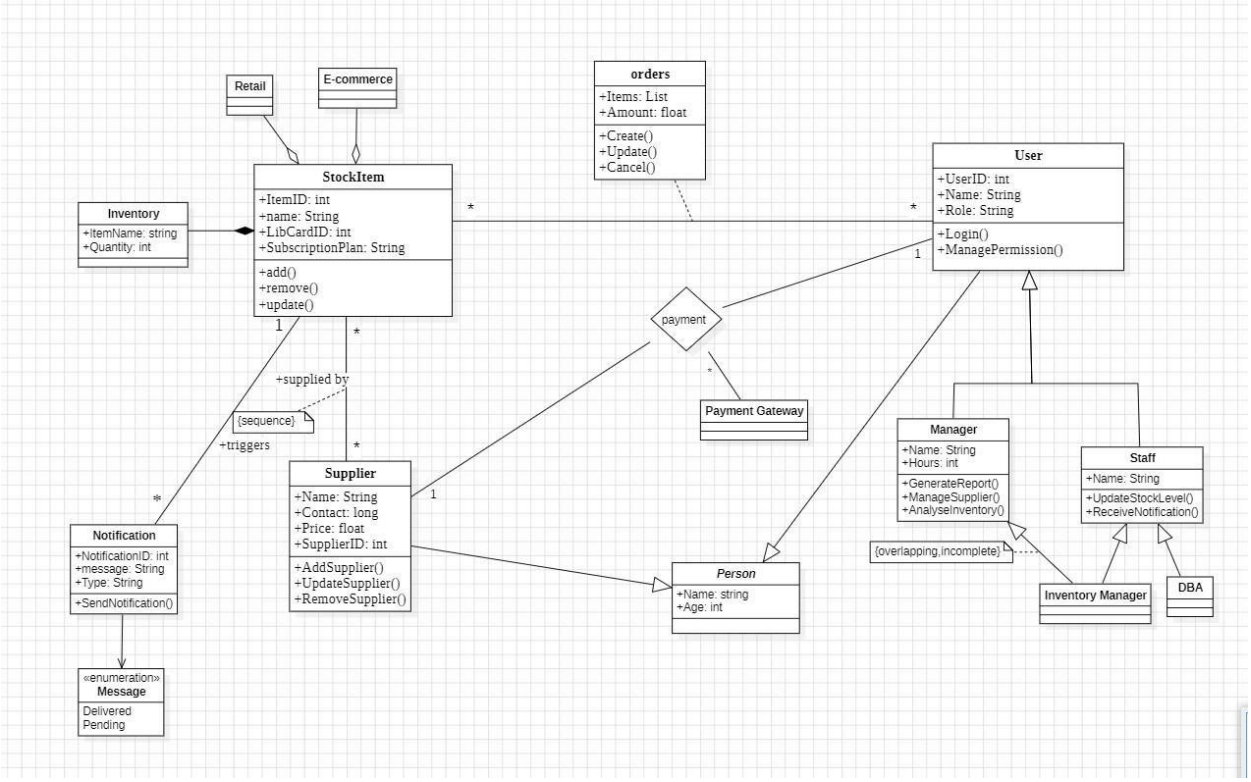


fig 4.3.2

- Classes:
    - StockItem: Represents items in the stock with attributes such as ItemID, name, and SubscriptionPlan. Methods include add(), remove(), and update().
    - Supplier: Tracks suppliers with attributes like Name, Contact, and SupplierID. It includes methods to add, update, or remove a supplier.
    - User: Represents system users with roles such as Manager or Staff, supporting actions like Login() and ManagePermission().
    - Manager and Staff: Specific types of users with additional operations like GenerateReport() for Managers and UpdateStockLevel() for Staff.
    - Notification: Handles notifications in the system, with details such as NotificationID and message.
- Relationships:
    - A StockItem is supplied by multiple Suppliers.
    - Notifications can be triggered by stock events.
    - Managers oversee suppliers, and Staff manage stock levels.
- Enumerations:

3.4 State Diagram

Simple State Diagram

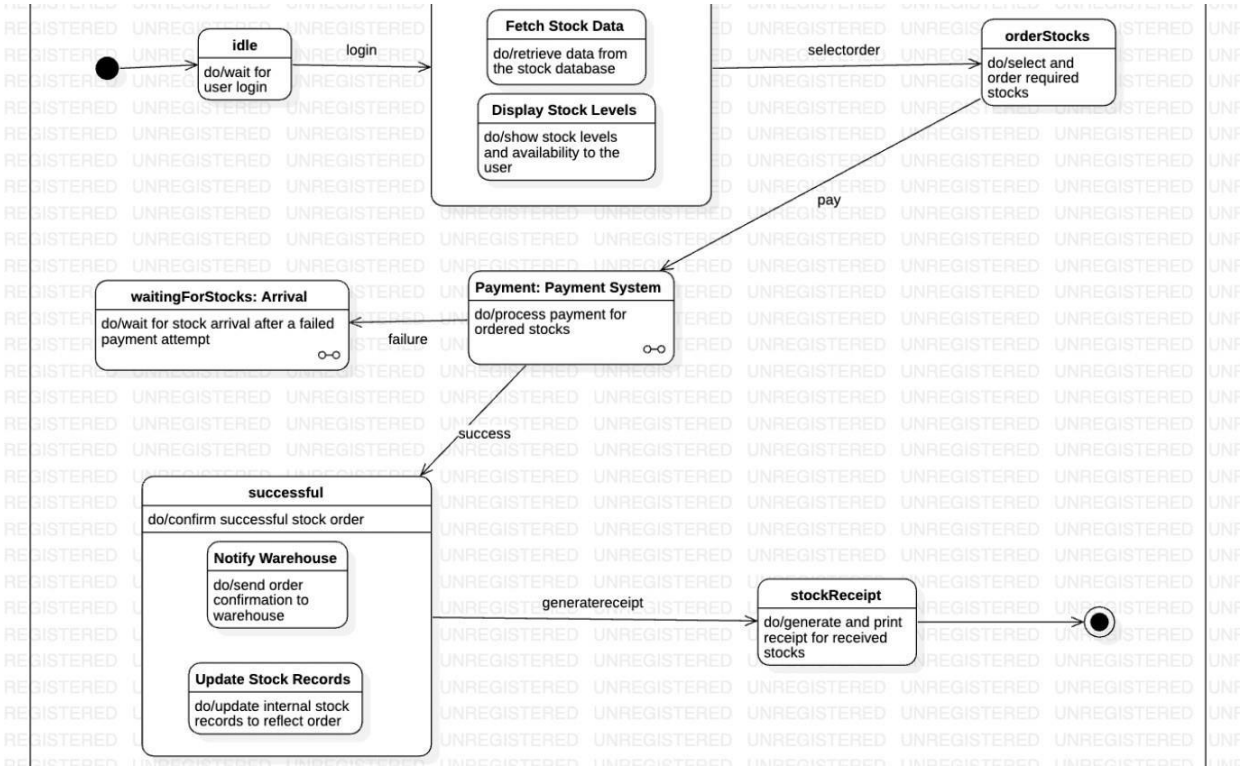


fig 4.4.1

The diagrams illustrate a Stock Maintenance System and its Stock Purchase subprocess, showcasing how stock levels are managed and purchases are handled.

Stock Maintenance System

- LoggingIn: Users log in with valid credentials.
- StockStatus: Displays the current stock levels. Users can view stocks, update records, or initiate purchases.
- MaintainRecord: Allows updates to stock records.
- OrderStock: Automatically triggers when stock levels are empty, placing orders for replenishment.
- OrderedStockStatus: Tracks the status of placed orders, including back orders or pending orders.
- LoggingOut: Users log out to end the session.

Stock Purchase

- InventoryCheck: Verifies if the requested stock quantity is available.
- Validation: Checks the entered details; if valid, the process continues.
- Payment: User completes the purchase by entering payment details.
- Sell: Finalizes the transaction by contacting the trader or seller.

Advanced State Diagram

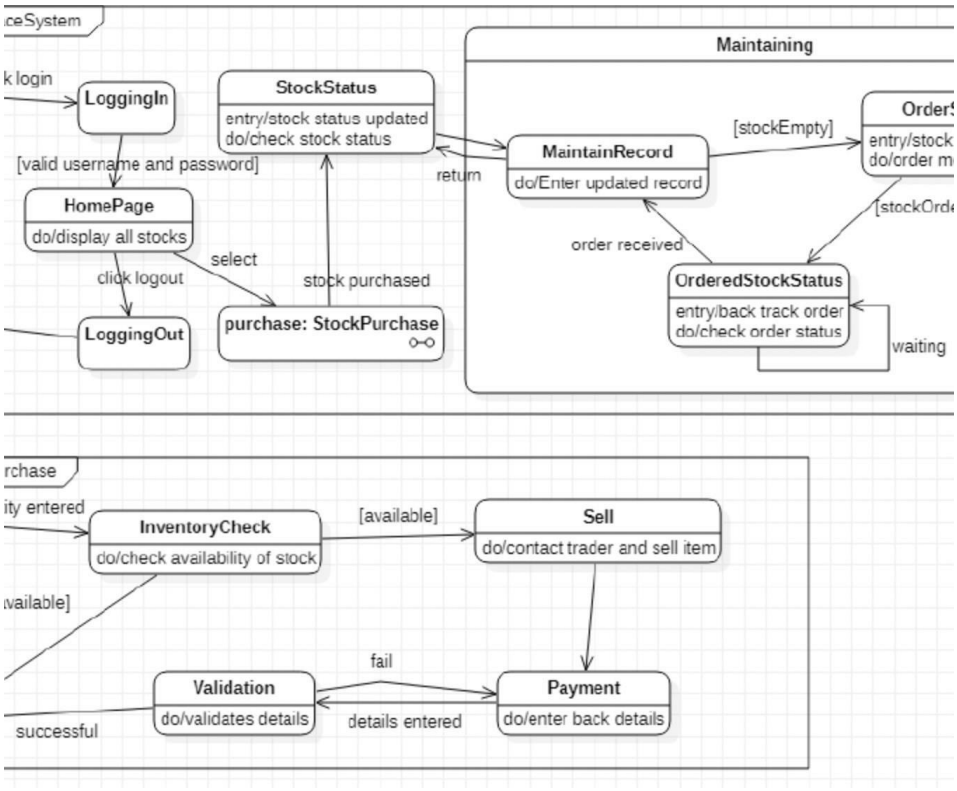


fig 4.4.2

- States:
    - Various conditions a stock item can exist in, such as Available, Out of Stock, or Reserved.
- Transitions:
    - Arrows representing the change from one state to another, triggered by events like Add, Remove, or Update.
- Start and End States:
    - Indicates the initialization and final state of a stock item.
    -

4.5 Use case diagram

Simple Use Case Diagram

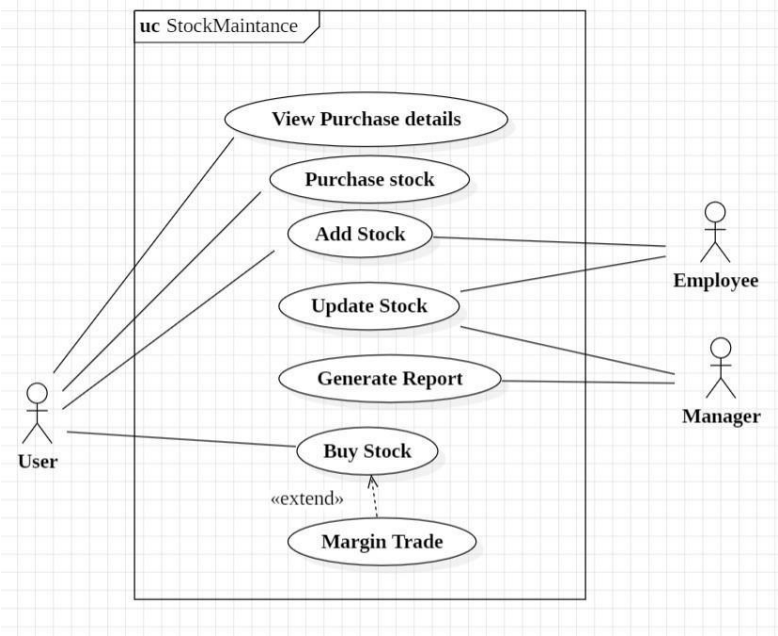


fig 4.5.1

The use case diagram depicts the primary functionalities of the stock maintenance system and the roles involved.

- Actors:
    - User: Can perform actions like viewing purchase details, purchasing stock, adding stock, updating stock, generating reports, and buying stock.
    - Employee: Collaborates with the system for stock-related tasks.
    - Manager: Focuses on generating reports and other managerial actions.
- **Use Cases**:
    - "Buy Stock" extends to "Margin Trade," indicating optional or conditional trading capabilities.
    - Functionalities are connected to the respective roles that interact with them.

Advanced Use Case Diagram

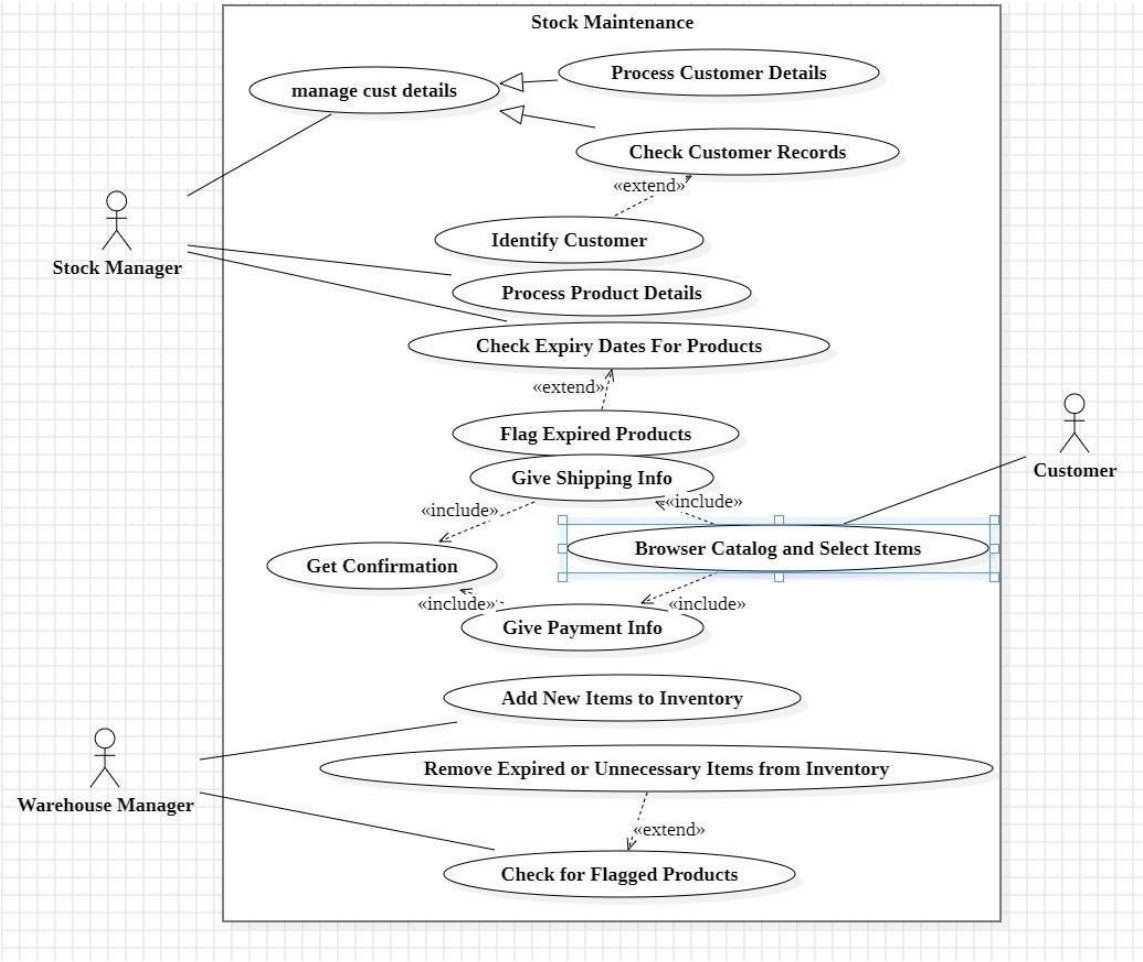


fig 4.5.2

This advanced use case diagram depicts detailed functionalities of the Stock Maintenance system involving Stock Manager, Warehouse Manager, and Customer.

- Stock Manager manages customer details, processes product details, and identifies customers.
- Warehouse Manager handles inventory by adding and removing items and flagging expired products.
- Customer interacts with the system to browse catalogs, give shipping information, and confirm payments.

4.6 Sequence Diagram

Simple Sequence Diagram

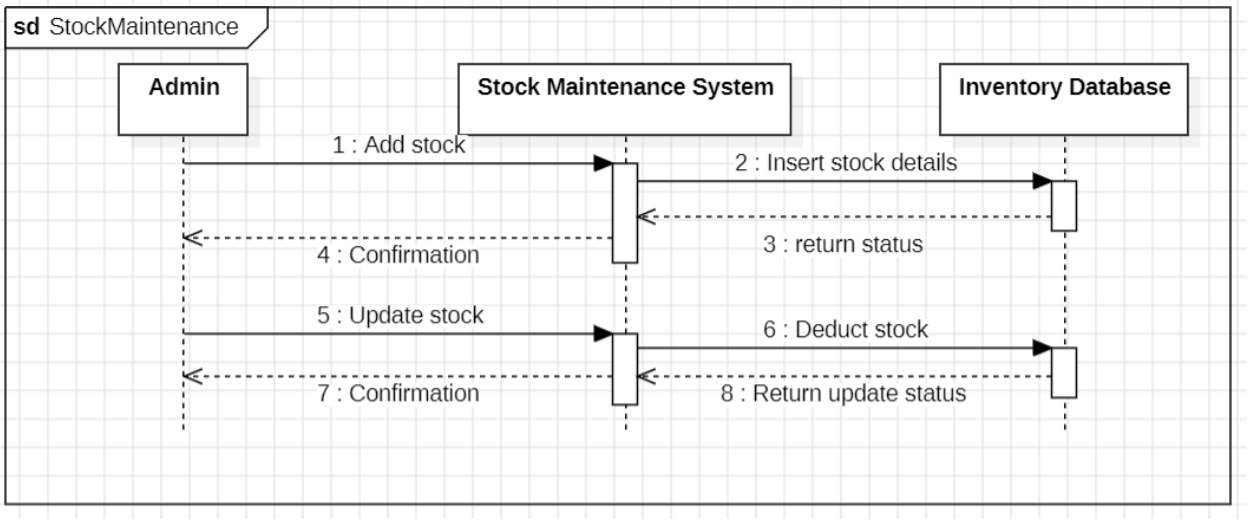


fig 4.6.1

This sequence diagram highlights the flow of interactions between the admin, stock maintenance system, and the inventory database for managing stock updates.

- Flow:
    - Admin adds or updates stock.
    - The stock maintenance system interacts with the inventory database to insert or deduct stock details.
    - Status or confirmation messages are returned at each stage to ensure consistency.

Advanced Sequence Diagram

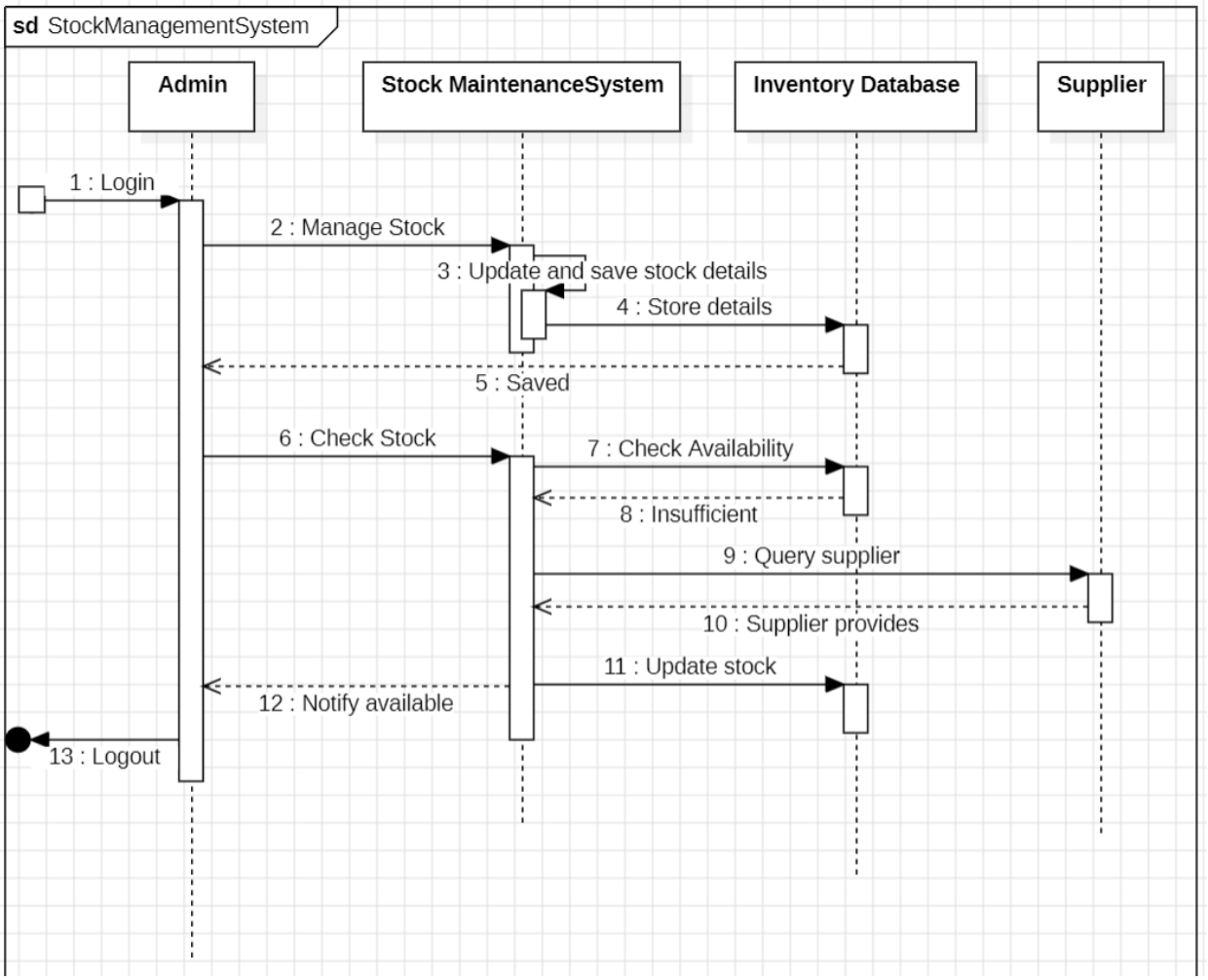


fig 4.6.2

This sequence diagram shows the flow of information among Admin, Stock Maintenance System, Inventory Database, and Supplier.

- The Admin logs in and manages stock, which is updated in the system and saved in the database.
- If stock is insufficient, the system queries the supplier, who provides additional stock.
- Notifications are sent back once the stock is updated.

4.7 Activity Diagram

Simple Activity Diagram

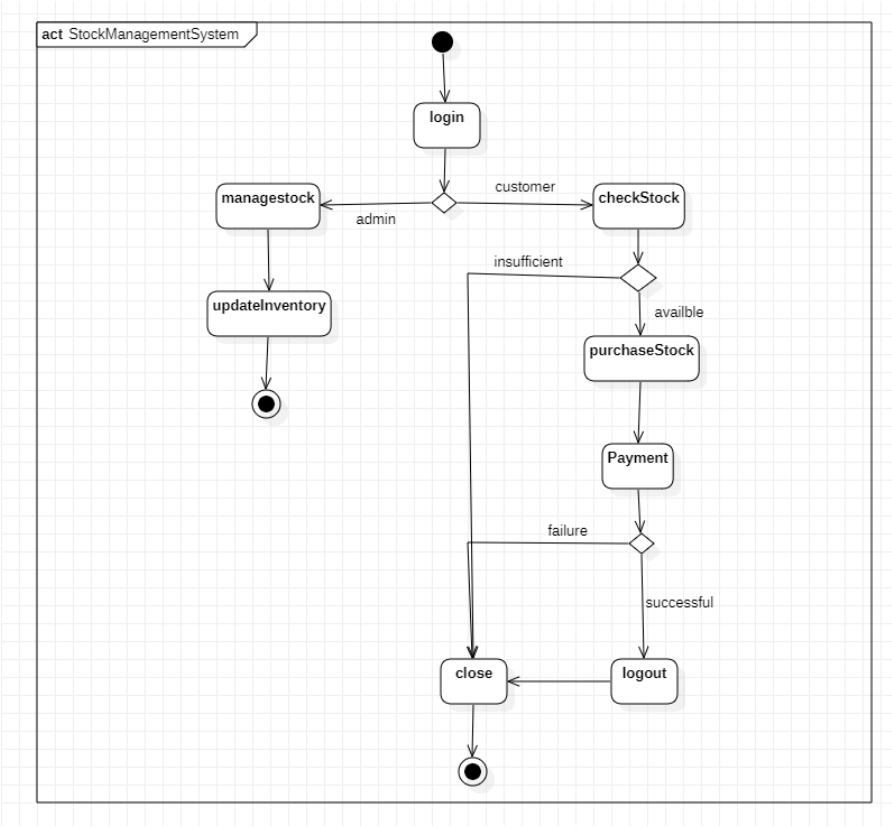


fig 4.7.1

The activity diagram outlines the overall process flow of the stock management system.

- Key Activities:
    - Login: Entry point for admin or customer.
    - Manage Stock: Admin manages inventory, leading to inventory updates.
    - Check Stock: Customers check stock availability.
    - Purchase Stock: Customers proceed with purchasing if stock is available, followed by payment.
    - Outcome:
        - Payment failure leads to closure.
        - Successful payment results in logout, completing the process.

Advanced Activity Diagram

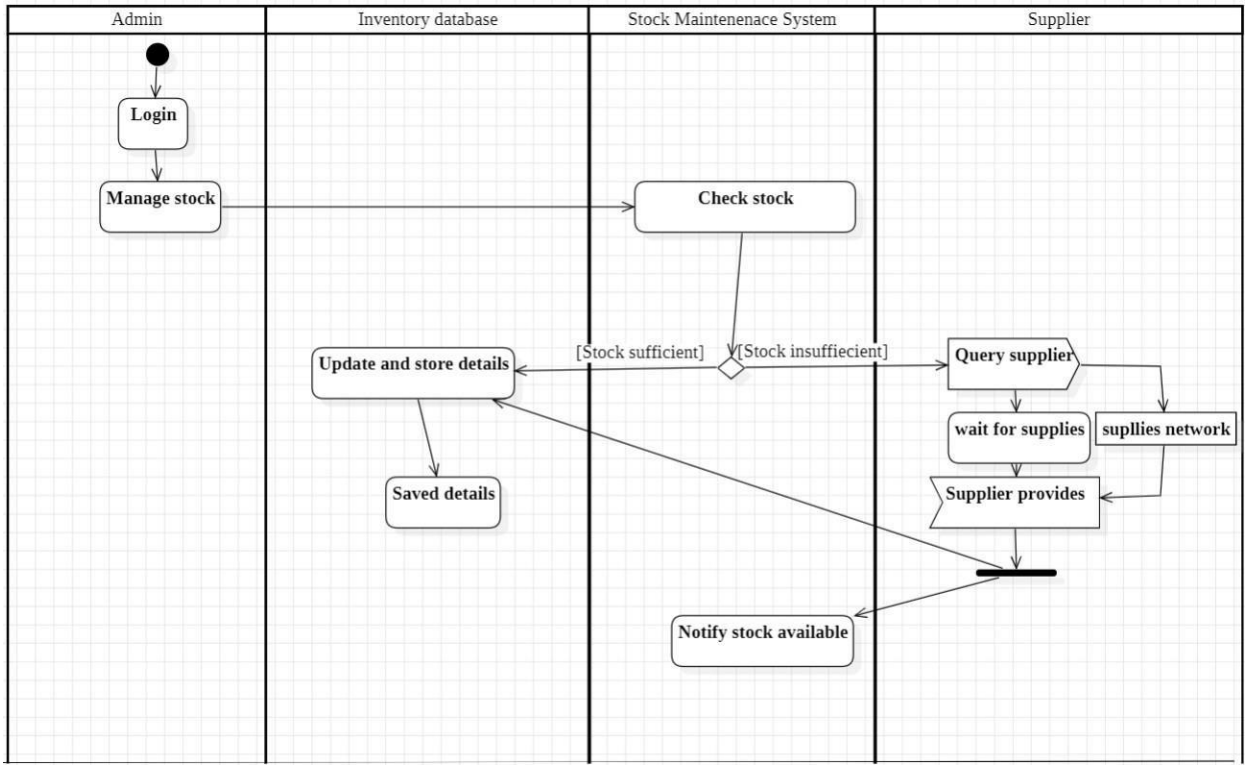


fig 4.7.2

This advanced activity diagram highlights the roles of Admin, Inventory Database, Stock Maintenance System, and Supplier in the stock management process.

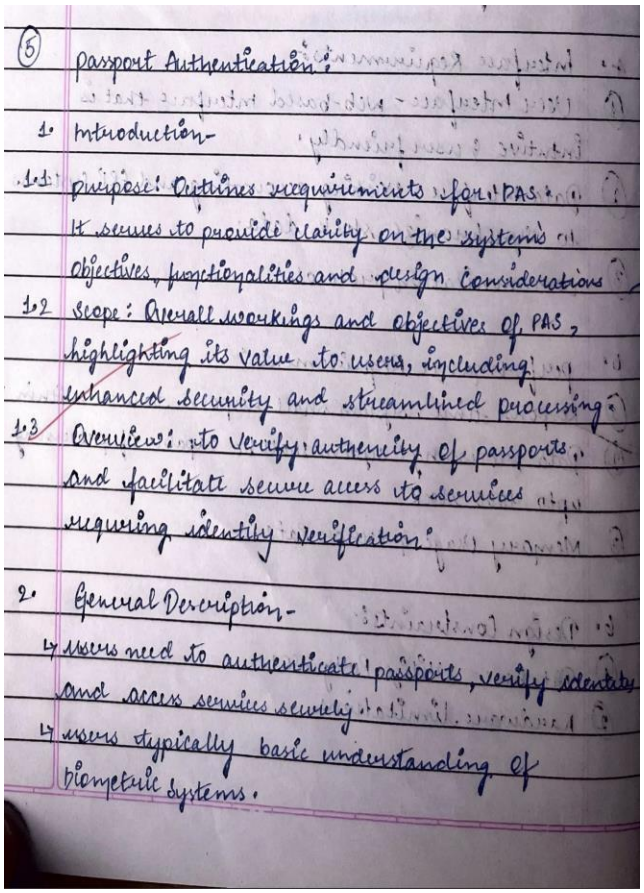
- The Admin initiates the process by managing stock.
- The Stock Maintenance System checks stock levels and interacts with the supplier if stock is insufficient.
- Stock updates and notifications are handled collaboratively between the system and database.

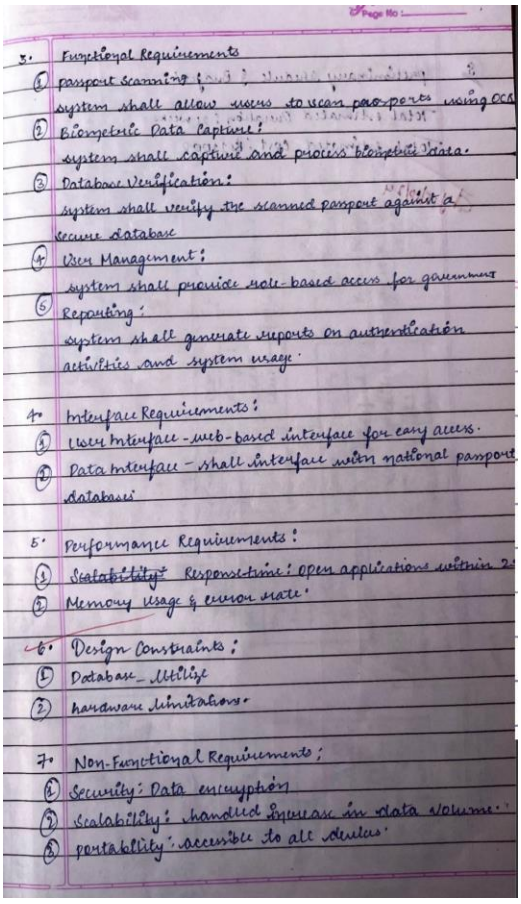
5. Passport Automation System

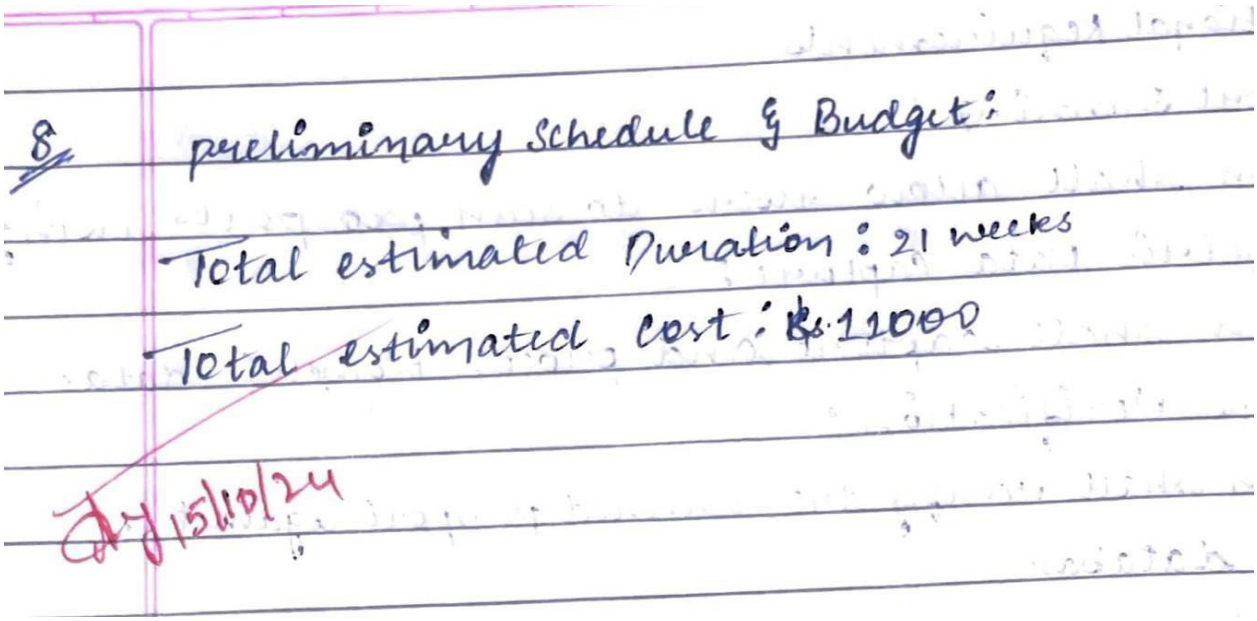
5.1 Problem Statement

Design a software system to automate passport issuance and renewal, integrating both manual verification by officials and an automated online portal. The system should handle application submissions, document verification, status tracking, and passport issuance efficiently. Each regional office will maintain its own database for managing applicant records and processing requests, while a central system oversees data synchronization and security. Applicants should be able to submit forms, upload documents, schedule appointments, and track their application status online. The system must ensure secure data handling, prevent delays through streamlined workflows, and provide transparency at every stage of the process.

5.2 Software Requirements Specification (SRS) Document







3.3 Class Diagram

Simple Class Diagram

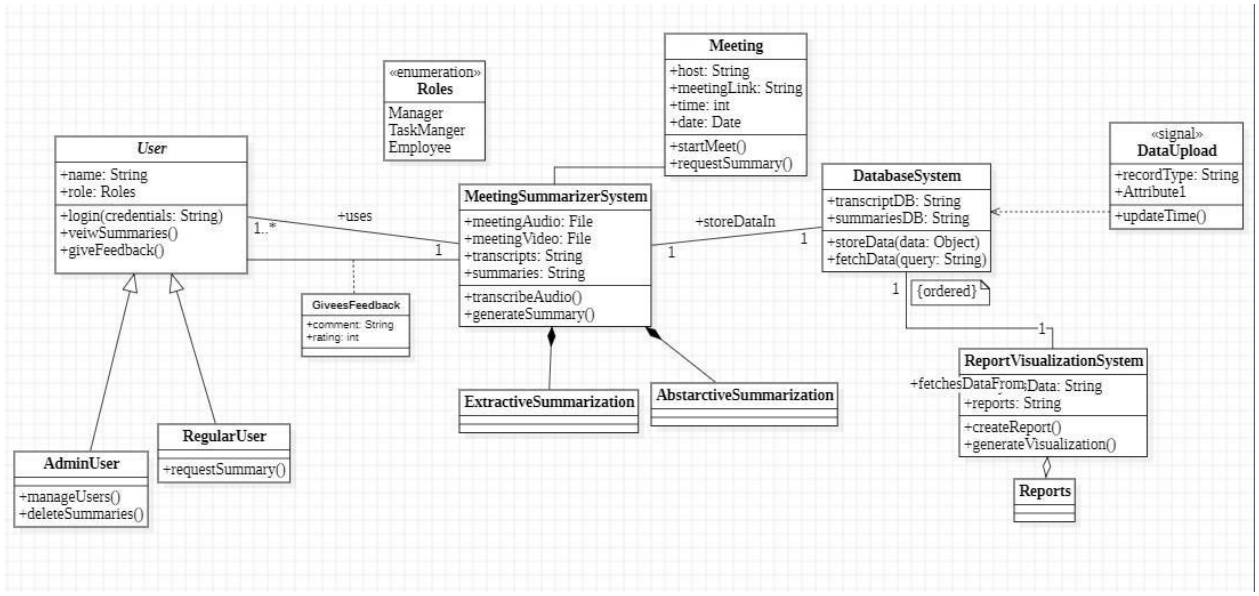


fig 5.3.1

Main Classes:

1. Applicant:
    - Attributes: applicantID, name, address, phone.
    - Methods: trackStatus(), uploadDocuments(), submitApplication().
2. Application:
    - Attributes: applicationID, status, subDate.

    - Methods: verifyDoc(), updateStatus().
3. Passport:
    - Attributes: passportID, issueDate, status.
    - Methods: issuePassport(), trackDelivery().
4. Staff:
    - Attributes: staffID, name, email.
    - Methods: verifyApplication(), processIssuance().
5. Notification (and its specializations: Email, SMS):
    - Attributes: recipient, message, sendDate.
    - Methods: send().

Key Relationships:

- Applicants submit Applications.
- Staff verify applications and issue Passports.
- Notifications update applicants.

Advanced Class Diagram

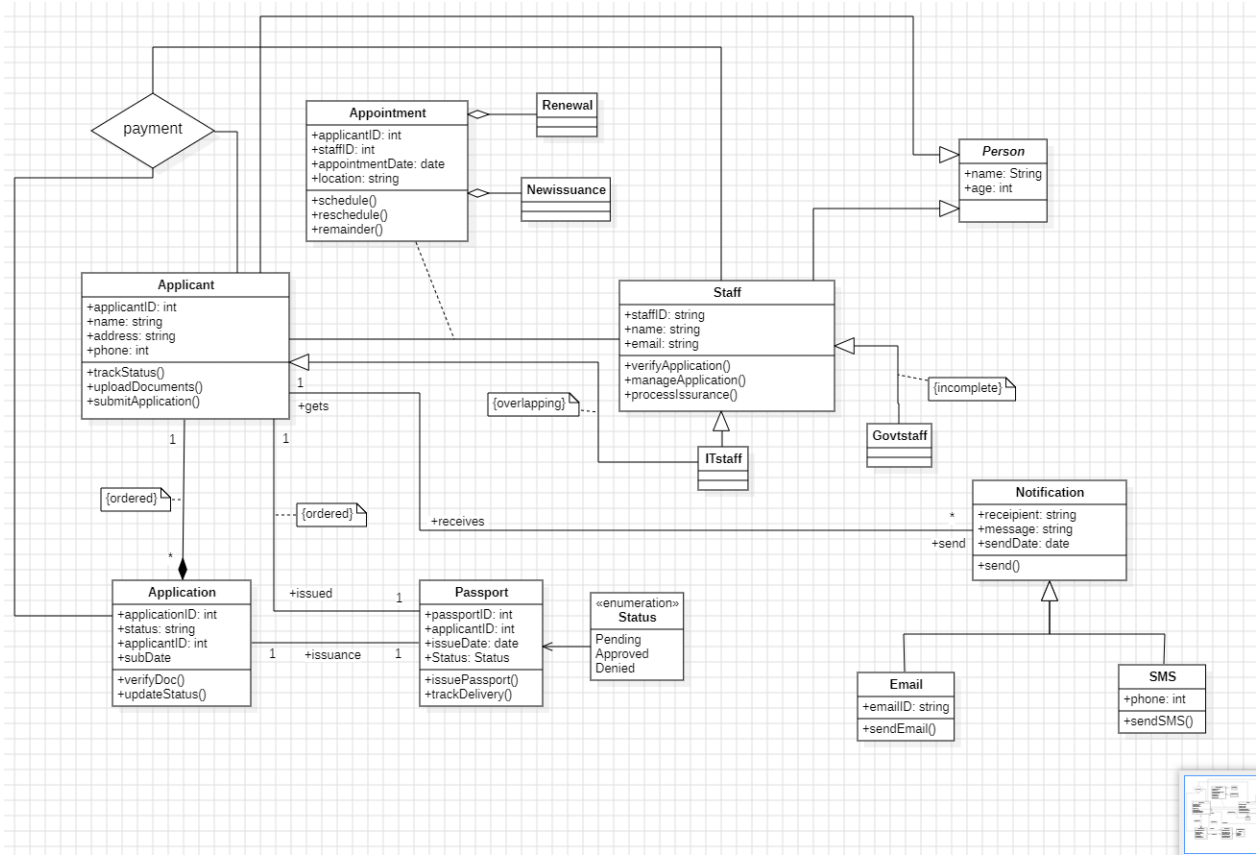


fig 5.3.2

This diagram models the structure of the passport automation system. Key classes include:

- Applicant: Represents users applying for a passport. Contains attributes such as ID, name, and address, along with methods to track status, upload documents, and submit applications.
- Application: Maintains details about the application, such as ID, status, and submission date. Methods include verifying documents and updating the status.
- Passport: Represents the issued passport with attributes like passport ID and issue date.
- Staff: Handles application verification and passport issuance.
- Notification: Facilitates sending updates through SMS or email.

5.4 State Diagram

Simple State Diagram

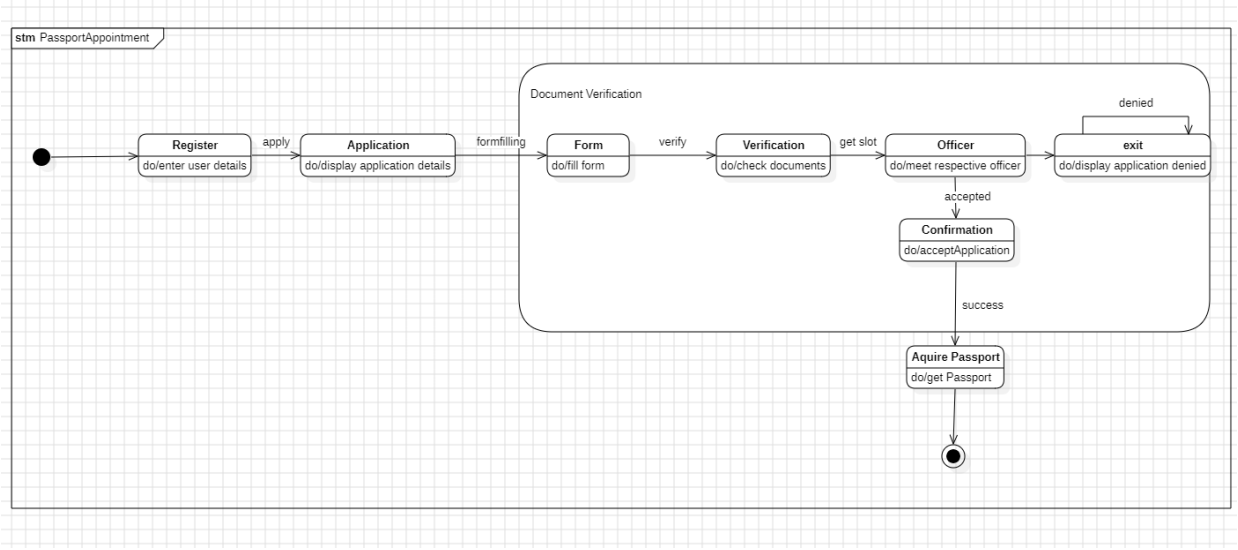


fig 5.4.1

This diagram focuses on the appointment scheduling process:

- Register: Users input their details to begin the process.
- Application: The user views application details and proceeds to fill out forms.
- Verification: Submitted documents are verified by the system or respective officers.
- Confirmation: If verified, the application is accepted; otherwise, it exits with an error.

Advanced State Diagram

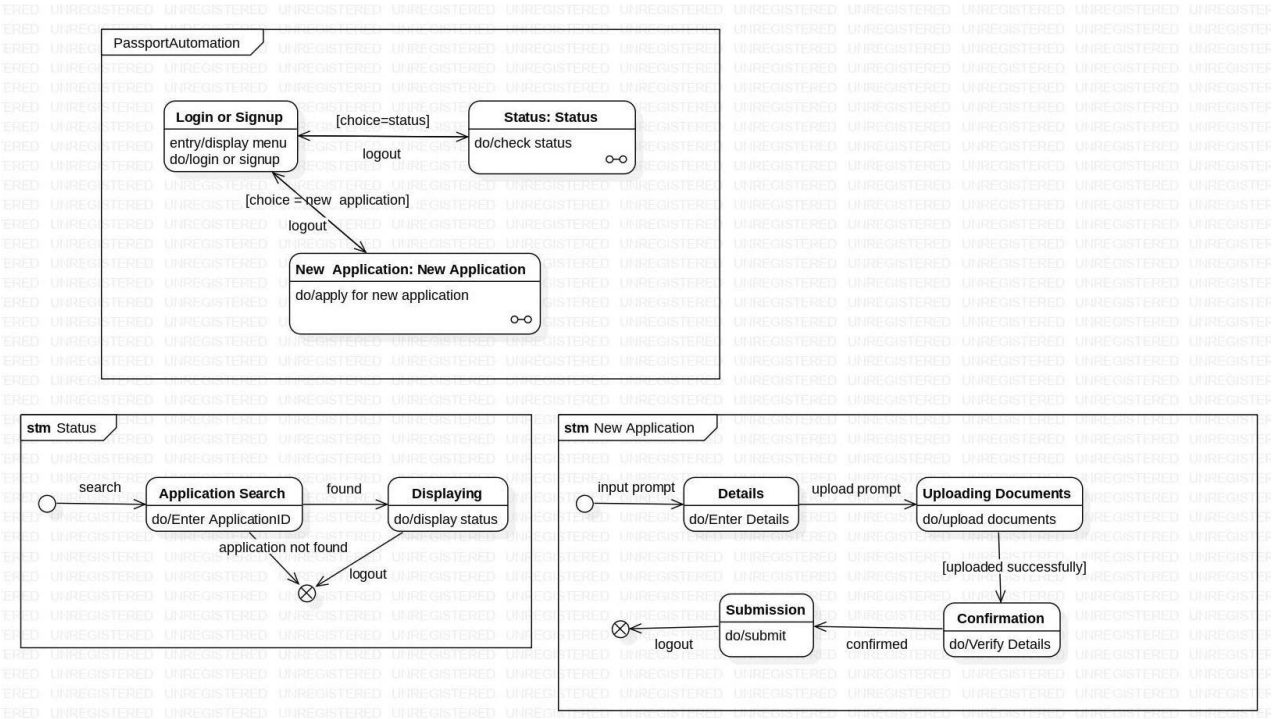


fig 5.4.2

This state diagram illustrates the dynamic behavior of the passport automation system. It depicts three major states:

- Login or Signup: Users can either log in or register for a new account.
- New Application: The process includes entering details, uploading required documents, submitting the application, and confirming details.
- Status Check: Users can track the application status by entering the application ID. The system confirms whether the application is found or not.

5.5 Use case diagram

Simple Use Case Diagram

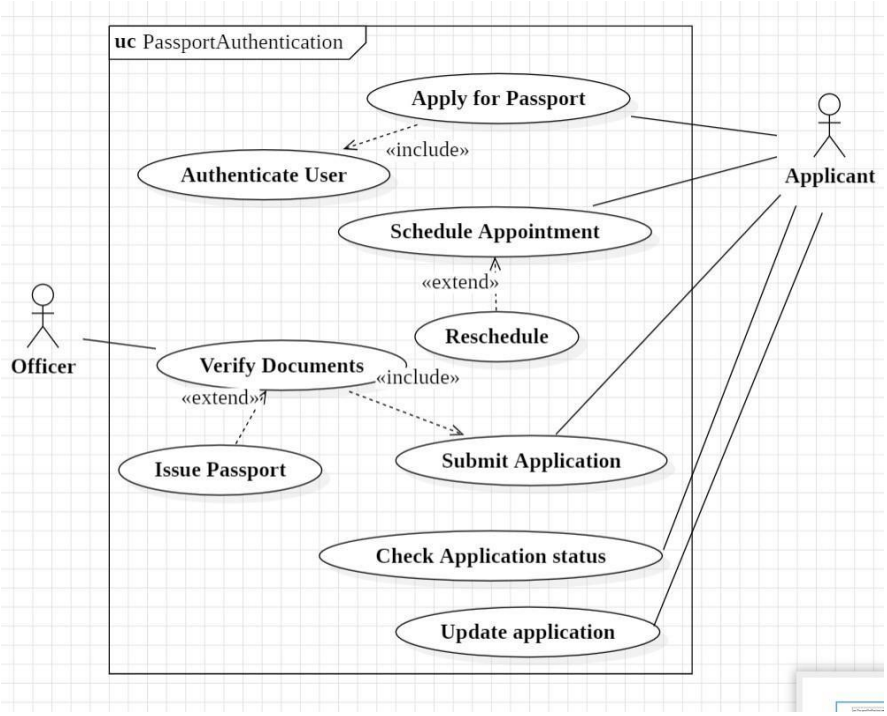


fig 5.5.1

The key actors are:

- Applicant: Responsible for applying for a passport, scheduling appointments, submitting applications, checking application status, and updating applications.
- Officer: Handles tasks like verifying documents and issuing passports.

The main use cases include:

- Apply for Passport: Initiated by the applicant and includes authentication and scheduling appointments.
- Submit Application: The applicant submits their passport application for processing.
- Verify Documents: Managed by the officer, ensuring all documents are valid before proceeding.
- Issue Passport: The final step handled by the officer after successful verification.
- Check Application Status: Allows applicants to track their application progress.
- Reschedule Appointment: Extends the scheduling use case for flexibility.

Relationships:

- The diagram uses include and extend relationships to represent dependencies and optional functionalities.

Advanced Use Case Diagram

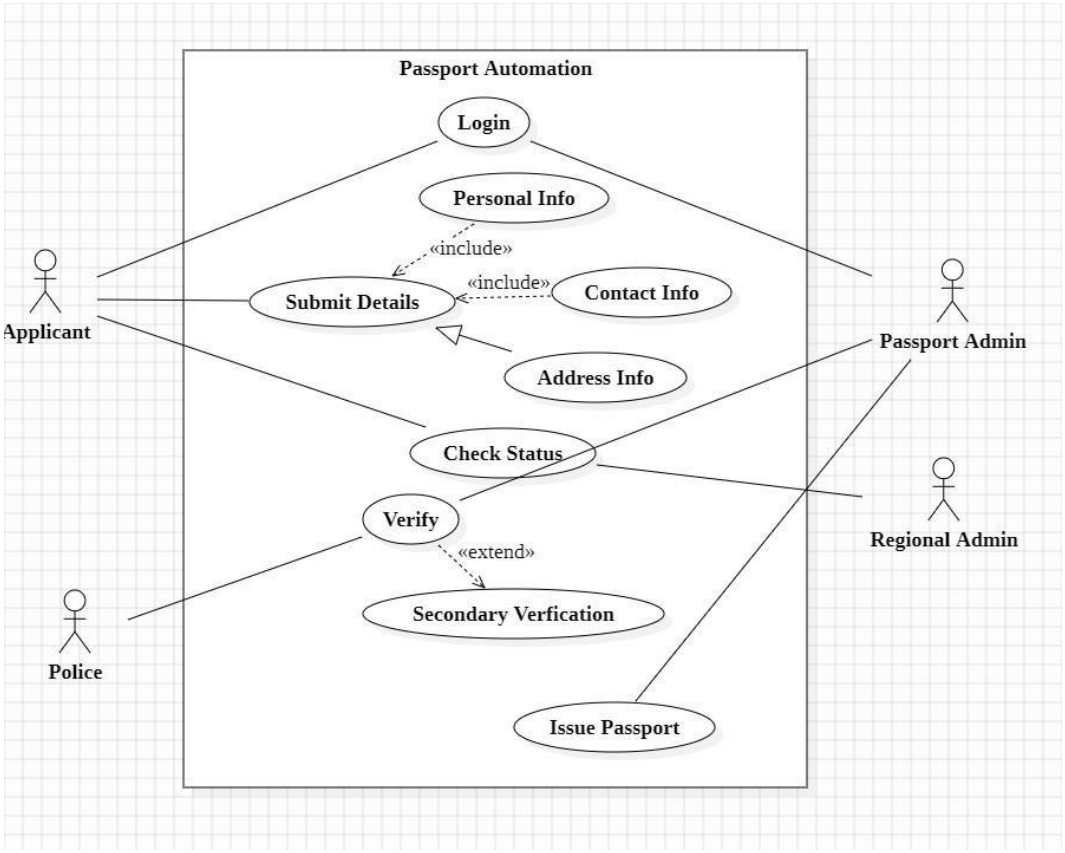


fig 5.4.2

- Actors:
    - Applicant: Logs in, submits personal/contact/address information, and checks application status.
    - Passport Admin: Verifies applications.
    - Regional Admin: Handles secondary verification and passport issuance.
    - Police: Assists in verification.
- Use Cases:
    - Core functions like login, submit details, check status, verification, and passport issuance.
- Relationships: Includes (mandatory functionality) and extends (optional or conditional steps).

Additional Elements:

- Advanced verification steps and conditions for issuing passports.
- Integration of multiple user roles like Admins and Police for secure processing.

5.6 Sequence Diagram

Simple Sequence Diagram

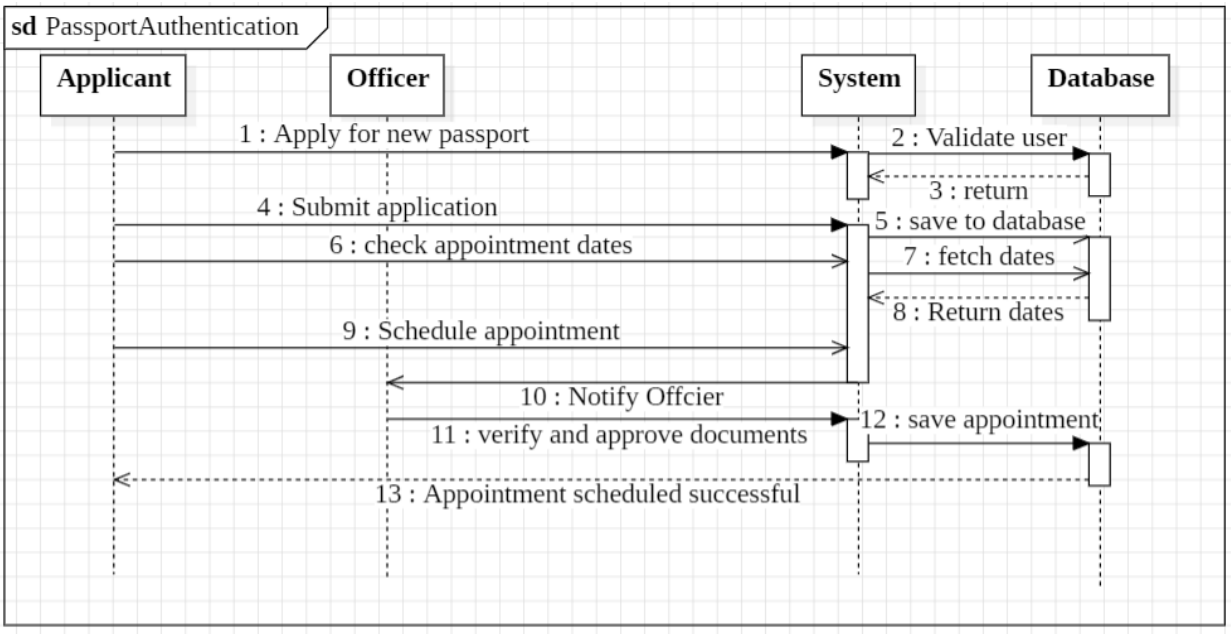


fig 5.6.2

The Sequence Diagram details the interactions between the Applicant, Officer, System, and Database during the passport application process. Key steps include:

1. The applicant applies for a passport, which triggers user validation by the system.
2. The validated user submits the application, and the data is saved in the database.
3. The applicant checks appointment dates, which are fetched and returned by the system.
4. An appointment is scheduled and saved, with the officer notified for further processing.
5. The officer verifies and approves documents, leading to a successful appointment confirmation.

Advanced Sequence Diagram

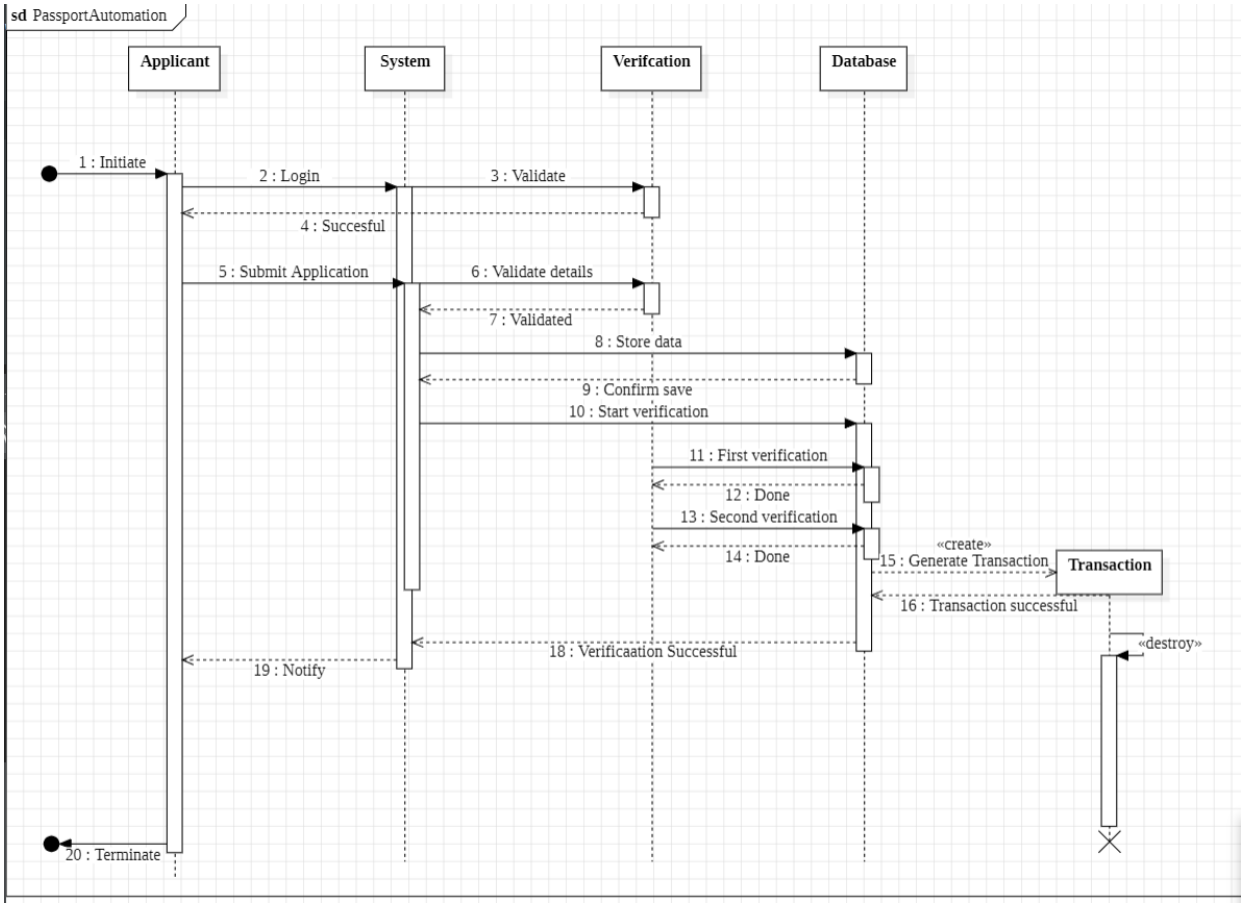


fig 5.6.2

Key Elements:

- Actors: Applicant, System, Verification module, Database, and Transaction service.
- Flow:
    1. Applicant logs in.
    2. Application is submitted and validated by the system.
    3. Details are stored in the database.
    4. Verification proceeds in two stages.
    5. Transaction is generated upon successful verification.
    6. Applicant is notified of the process status.

Additional Elements:

- Loops for repeated verification attempts.
- Conditional paths for data correction and system failure scenarios

5.7 Activity Diagram

Simple Activity Diagram

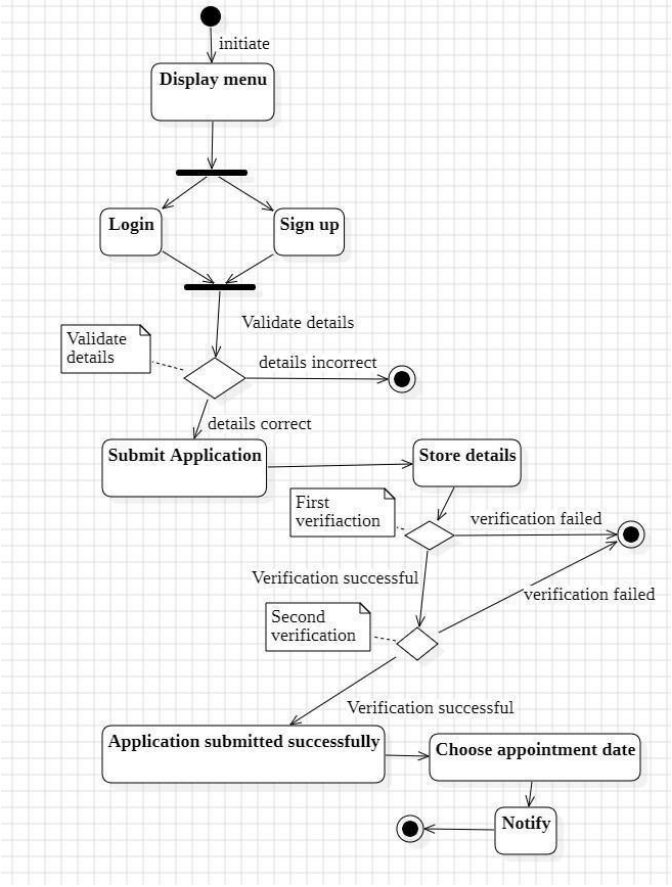


fig 5.7.1

The Activity Diagram represents the flow of activities in the Passport Authentication System, starting from the display menu:

1. Login/Sign Up: Users must either log in or sign up. Validation ensures correct user details.
2. Submit Application: Once validated, users proceed to submit their application.
3. Store Details: Application details are stored, followed by two levels of verification.
4. Verification: If both verification levels succeed, the application is marked as successfully submitted.
5. Appointment Scheduling: The user selects an appointment date and is notified accordingly

Advanced Activity Diagram

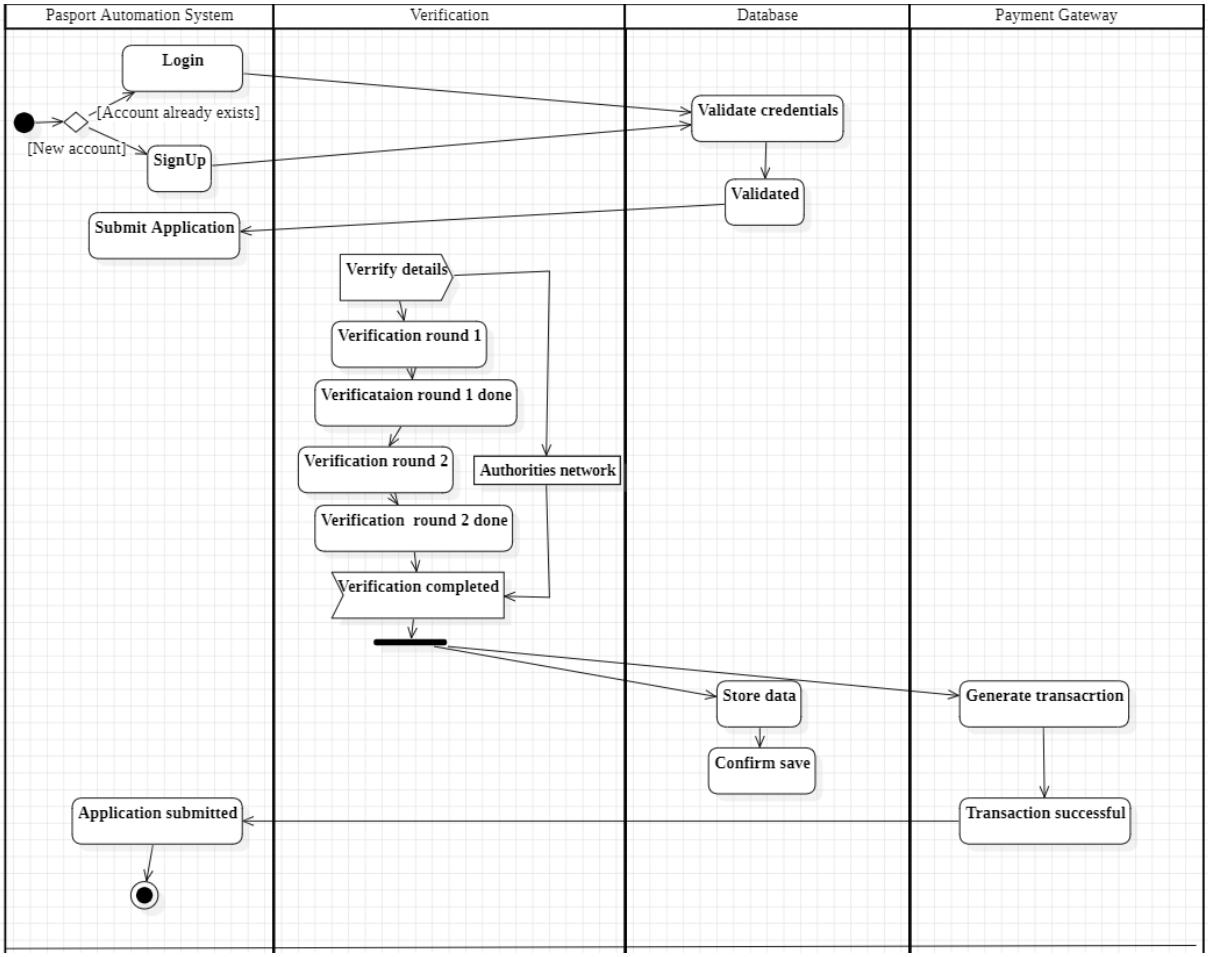


fig 5.7.2

Key Elements:

- Swimlanes: Divides actions across:
    - Passport Automation System: Login, sign-up, and application submission.
    - Verification: Two rounds of data verification.
    - Database: Credential validation and data storage.
    - Payment Gateway: Transaction generation.
- Flow:
    - If the account exists, login proceeds; otherwise, new accounts are created.
    - Submitted details undergo verification rounds before being saved.
    - Transaction completes the process.

Description: A high-level overview of process dependencies, emphasizing decision points, parallel actions, and transitions.