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:

MAKADIA RISHIT DILIPBHAI

1BM23CS177

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 Makadia Rishit Dilipbhai (1BM23CS177) during the 5th Semester August 2025-December 2025

.

Signature of the Faculty Incharge

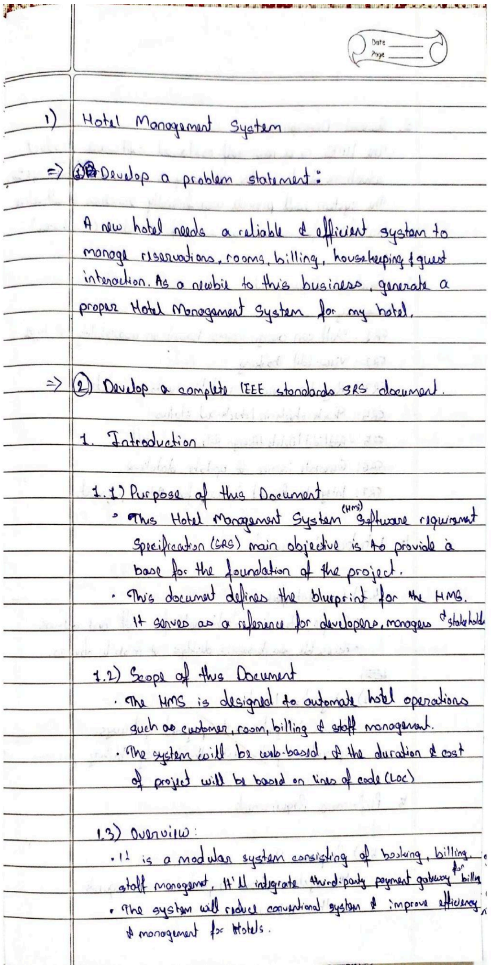
Sheetal V A
Assistant Professor

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

Table of Contents

1. Hotel Management System

SRS Document



1) Hotel Management System

=> (1) Develop a problem statement :

A new hotel needs a reliable & efficient system to manage reservations, rooms, billing, housekeeping & guest interaction. As a newbie to this business, generate a proper Hotel Management System for my hotel.

=> (2) Develop a complete IEEE standards SRS document.

1. Introduction

1.1) Purpose of this Document

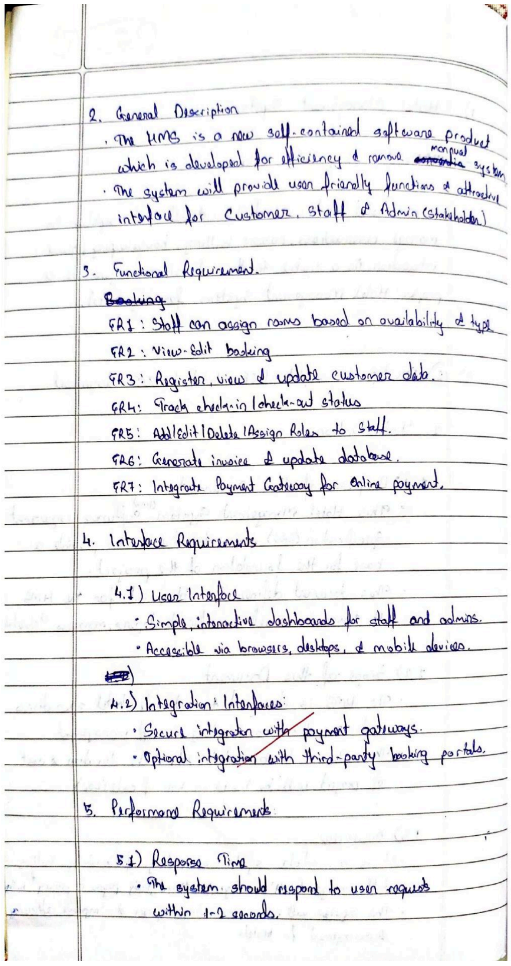
- This Hotel Management System (HMS) Software requirement Specification (SRS) main objective is to provide a base for the foundation of the project.
- This document defines the blueprint for the HMS. It serves as a reference for developers, managers & stakeholders

1.2) Scope of this Document

- The HMS is designed to automate hotel operations such as customer, room, billing & staff management.
- The system will be web-based, & the duration & cost of project will be based on lines of code (LOC)

1.3) Overview

- It is a modular system consisting of booking, billing, staff management. It'll integrate third-party payment gateway for billing
- The system will reduce conventional system & improve efficiency & management for Hotels.



2. General Description

- The HMS is a new self-contained software product which is developed for efficiency & remove manual system
- The system will provide user friendly functions & attractive interface for Customer, Staff & Admin (Stakeholder)

3. Functional Requirement.

~~Booking~~

FR1 : Staff can assign rooms based on availability & type

FR2 : View-Edit booking

FR3 : Register, view & update customer data.

FR4 : Track check-in/check-out status

FR5 : Add/Edit/Delete/Assign Roles to staff.

FR6 : Generate invoice & update database.

FR7 : Integrate Payment Gateway for online payment.

4. Interface Requirements

4.1) User Interface

- Simple, interactive dashboards for staff and admins.
- Accessible via browsers, desktops, & mobile devices.

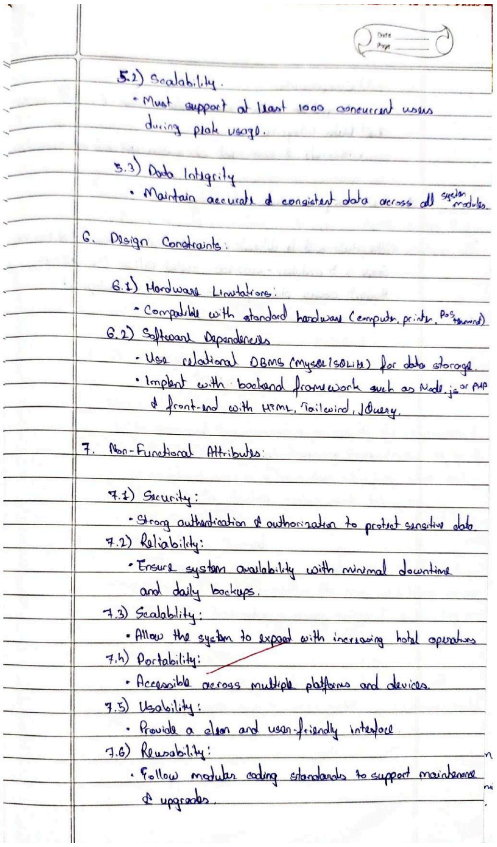
4.2) Integration/Interfaces

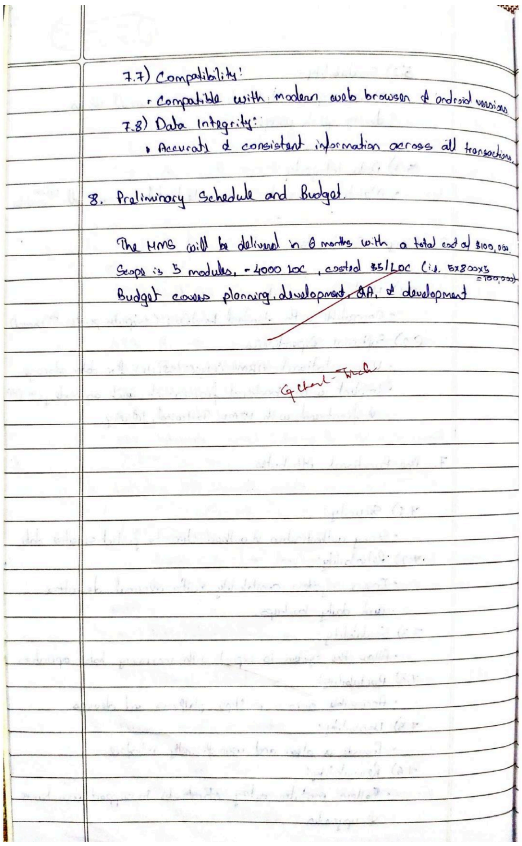
- Secure integration with payment gateways.
- Optional integration with third-party booking portals.

5. Performance Requirements

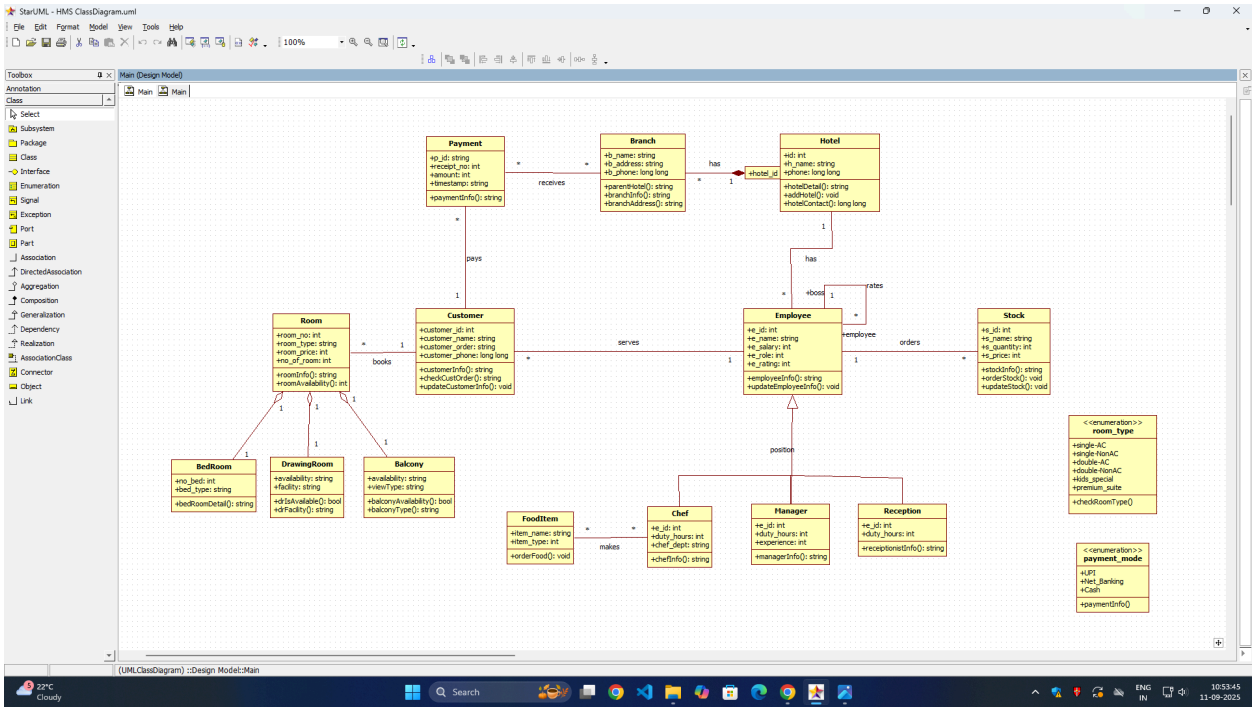
5.1) Response Time

- The system should respond to user requests within 1-2 seconds.





Class Diagram



Explanation:

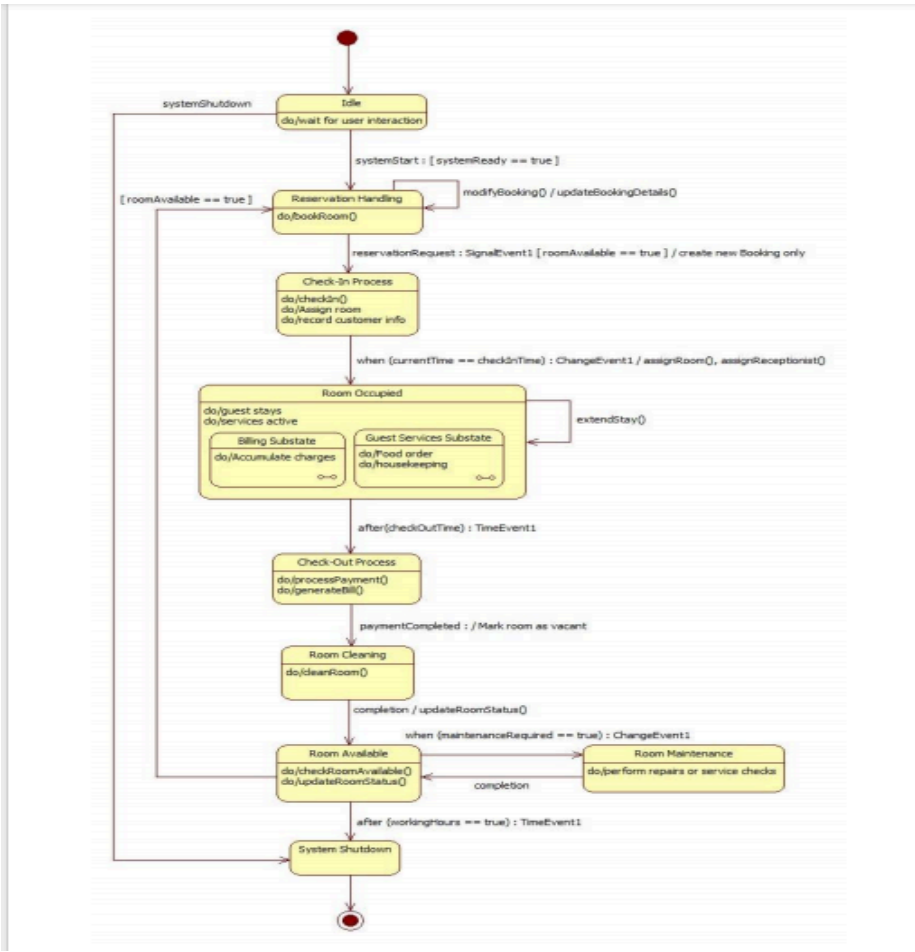
Core Components:

- Customer & Booking: Handles guest reservations and customer data management
- Room & Hotel: Manages room inventory, types, and hotel branch information
- Employee Roles: Includes specialized staff like Chef, Manager with distinct responsibilities
- Payment Processing: Handles financial transactions and billing
- Inventory Management: Tracks stock and tool items for hotel operations

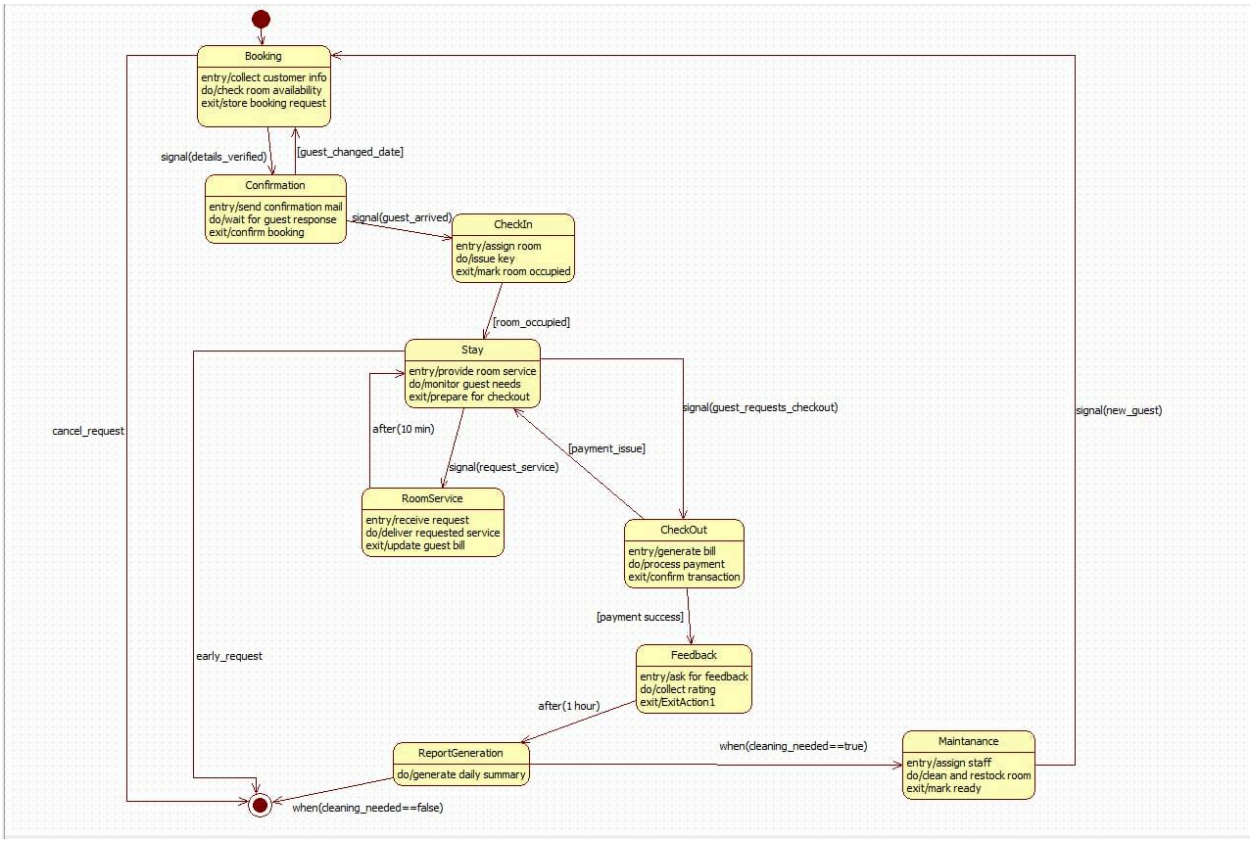
Key Features:

- Manages multiple hotel branches with shared address systems
- Handles room bookings with different room types and configurations
- Includes staff management with role-based functionality
- Processes payments and maintains customer records

State Diagram(Advanced)



State Diagram(Simple)



Simple State Diagram (Booking & Payment)

This diagram illustrates the complete lifecycle of a hotel room booking, tracking states from initial reservation through guest stay to post-checkout activities:

Booking States:

- Booking: Initial state handling customer information and room availability checks
- Confirmation: State managing booking confirmation and guest communication
- CheckIn: Room assignment and key issuance process
- Room_occupied: Active guest stay with room service provisioning
- CheckOut: Bill processing and payment transaction handling

Supporting States:

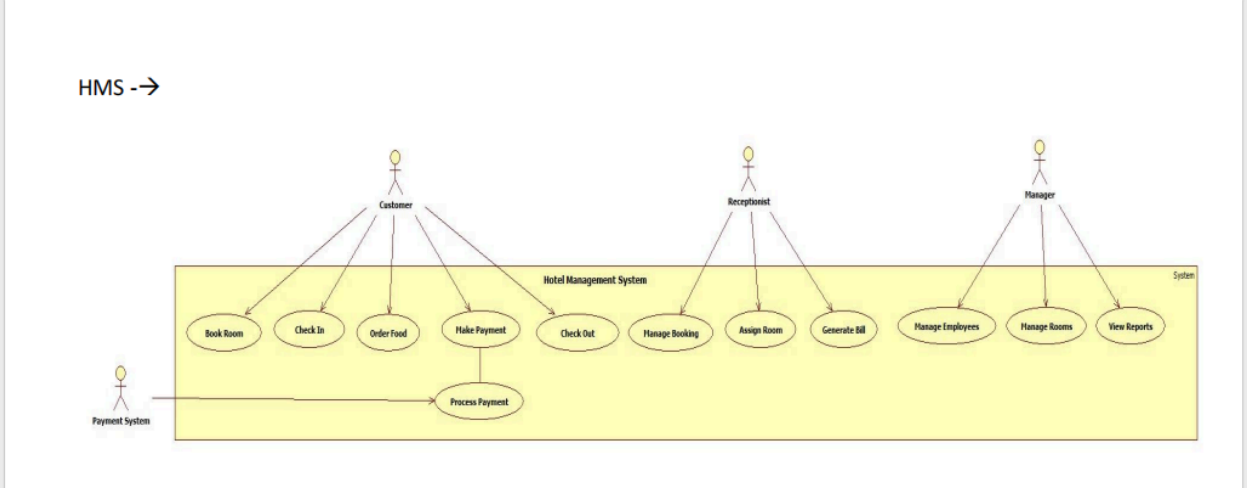
- RoomService: Handles guest service requests during occupancy
- Maintenance: Manages room cleaning and preparation for next guest
- Feedback: Collects guest ratings and reviews post-stay

Advanced State Diagram (Full Hotel Room Lifecycle)

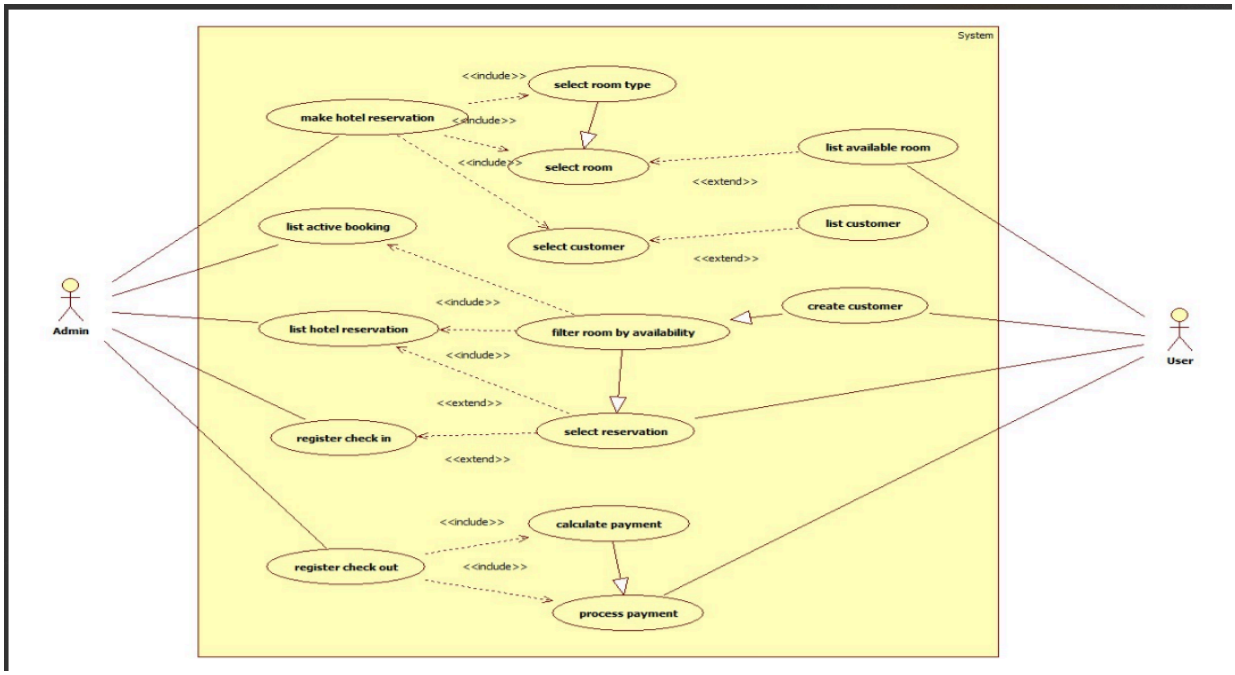
Key Transitions & Events

- State Changes: Triggered by guest actions (CheckIn, CheckOut) and system events (payment_success, cleaning_needed)
- Timed Transition: Automatic checkout processing after payment completion (after (10 min))
- Conditional Flow: Maintenance triggered based on room status (when(cleaning_needed==true))

Use Case (Simple)



Use Case (Advanced)



Simple Use Case Diagram

Primary Actors:

- Customer: Main user making reservations and checking in/out
- Receptionist: Staff member handling reservations and guest processes
- System: Automated backend managing room inventory and payments

Advanced Use Case Diagram

Key Use Cases:

- Make Hotel Reservation (<<include>> Select Room Type, Select Room)
- Register Check In (<<extend>> Select Reservation)
- Register Check Out (<<include>> Process Payment, <<extend>> Calculate Payment)

- List Available Rooms (<<include>> Filter Room by Availability)

Relationship Analysis

Include Relationships:

- Mandatory processes like selecting a room during reservation
- Payment processing as essential part of check-out
- Availability filtering for room listing

Extend Relationships:

- Optional calculation of payment details during check-out
- Customer selection as conditional extension of reservation listing
- Reservation selection extending the check-in process

Scenarios

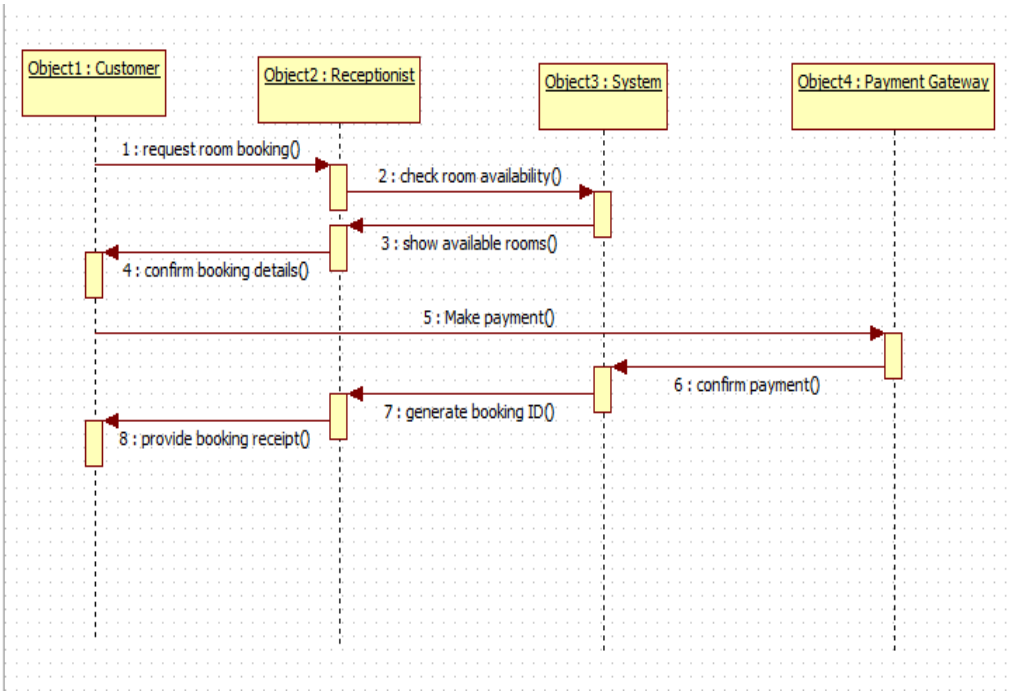
Scenario 1: Book Room

1. The customer opens the hotel booking page.
2. The customer selects the preferred room type (Single, Double, AC, Non-AC, etc.).
3. The system filters and displays all available rooms matching the selected type.
4. The customer selects one room from the displayed list.
5. The customer enters personal details such as name, contact number, and ID proof.
6. The system verifies the entered information for completeness and validity.
7. The customer confirms the booking.
8. The system generates a booking confirmation and stores the reservation details in the database.

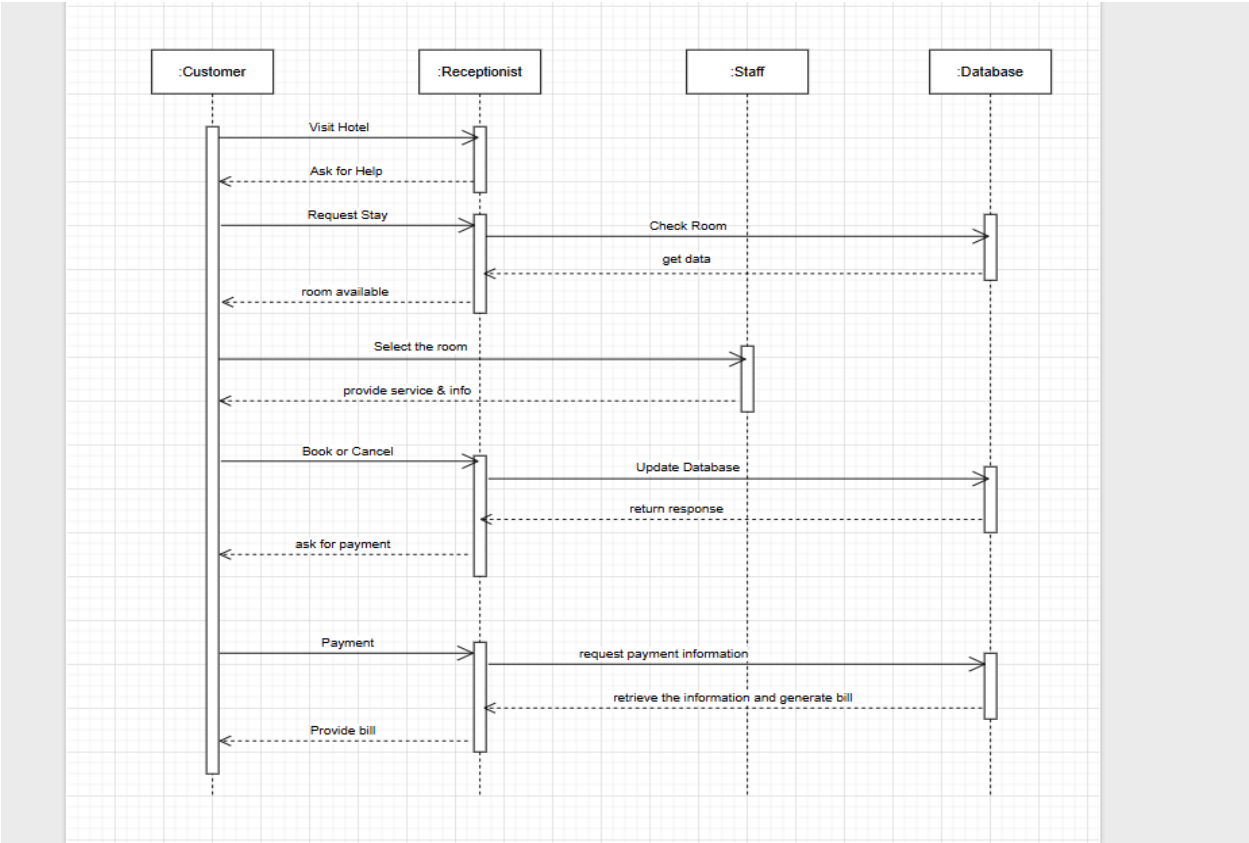
Scenario 2: Process Payment

1. The customer selects the **"Make Payment"** option.
2. The system displays the total bill amount.
3. The customer chooses a payment method such as UPI, card, or net banking.
4. The customer enters the required payment details.
5. The payment gateway validates the information and processes the transaction.
6. Upon successful payment, the gateway sends a confirmation message to the hotel system.
7. The system marks the booking as **"Paid"** and generates a payment receipt for the customer.

Sequence Diagram(Simple)



Sequence diagram(Advance)



Simple Sequence Diagram (Customer–Receptionist–System–Payment Gateway)

Initial Phase:

- Customer initiates the process by visiting the hotel and requesting a stay
- Receptionist checks room availability by querying the Database

Booking Phase:

- Database returns available room data to Receptionist
- Receptionist provides room options and service information to Customer
- Customer selects a room and confirms booking (Book or Cancel decision)

Payment & Completion:

- Receptionist requests payment information after booking confirmation

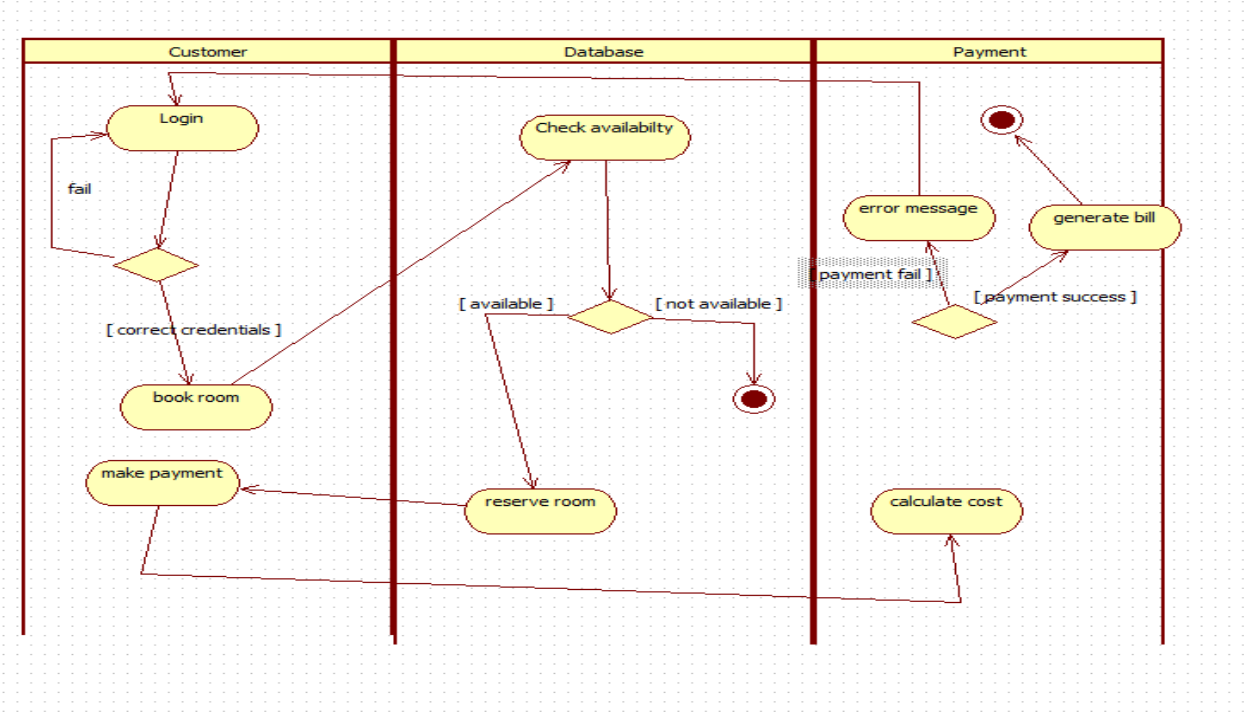
- Customer provides payment details
- Receptionist processes payment and generates bill through Database
- Final bill is provided to Customer to complete the transaction

Advanced Sequence Diagram (Customer–Receptionist–Staff–Database)

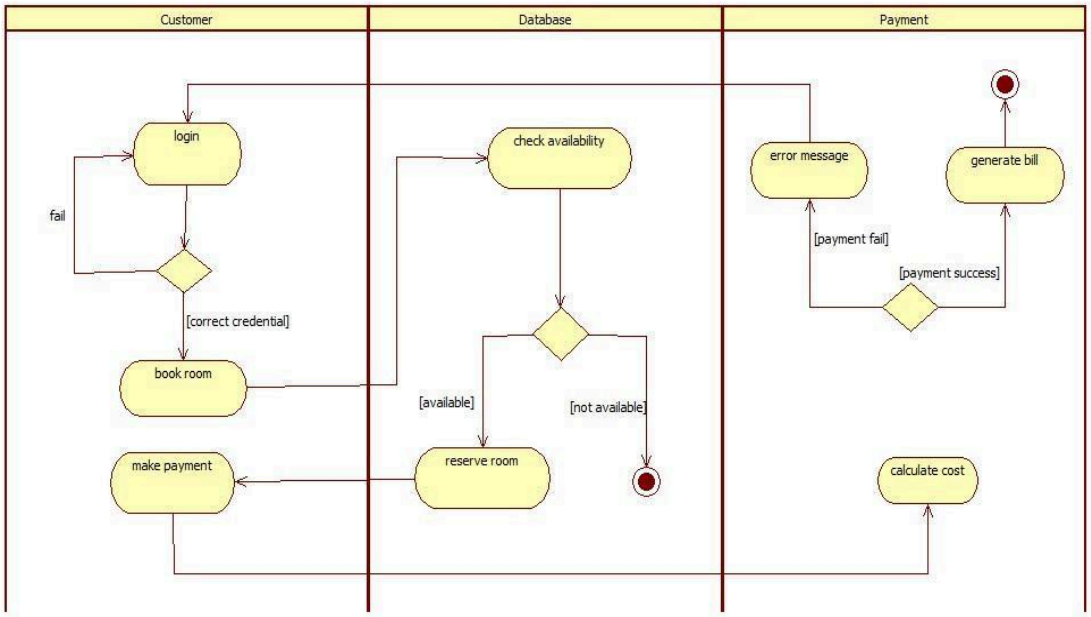
Interaction Patterns

- Synchronous Communication: Direct request-response between Receptionist and Database
- User-Driven Flow: Customer decisions (Book/Cancel) determine process direction
- Data Consistency: Real-time database updates maintain accurate room availability

Activity Diagram(Simple)



Activity Diagram(Advanced)



Simple Activity Diagram

This activity diagram shows the hotel booking process. A customer logs in, books a room if available, makes a payment, and receives a bill if successful. If anything fails (login, availability, or payment), an error message is shown.

- Login: The customer enters their details. If the login fails, they try again; if it succeeds, they move forward.
- Book Room: The system checks room availability. If no rooms are available, the process ends. If a room is available, it is reserved.
- Make Payment: The customer pays for the booking. If the payment fails, an error message appears. If it succeeds, the system generates the final bill.

Advanced Activity Diagram

Swimlanes & Control Flow:

- Customer: Initiates actions (login, booking, payment)
- Database: Handles data checks (credentials, availability)
- Payment: Processes financial transactions

Decision Nodes & Guards:

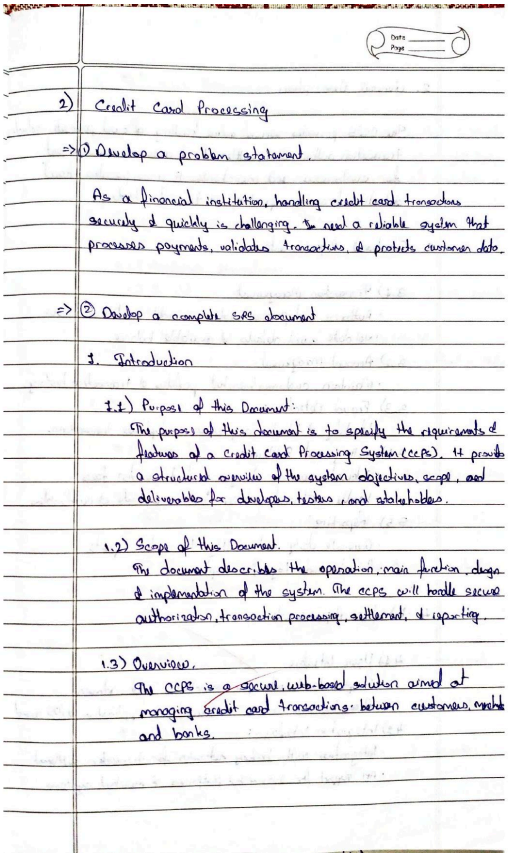
- Credential validation ([correct credential]/[fail])
- Room availability check ([available]/[not available])
- Payment verification ([payment success]/[payment fail])

Parallel Processing:

- Simultaneous room reservation and cost calculation
- Integrated payment processing with bill generation
- Error handling across all transaction stages

2.Credit Card Processing

SRS Document



2) Credit Card Processing

=> ① Develop a problem statement.

As a financial institution, handling credit card transactions securely & quickly is challenging. We need a reliable system that processes payments, validates transactions, & protects customer data.

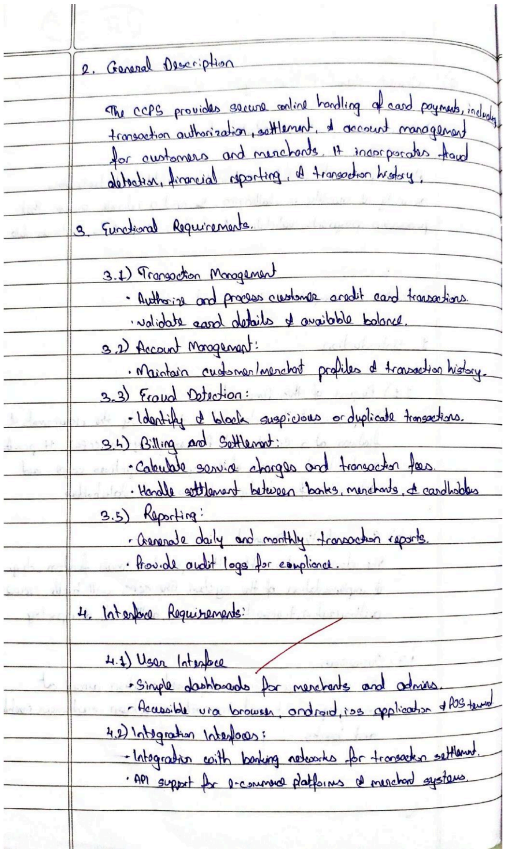
=> ② Develop a complete SRS document

1. Introduction

1.1) Purpose of this Document:
The purpose of this document is to specify the requirements & features of a Credit Card Processing System (CCPS). It provides a structured overview of the system objectives, scope, and deliverables for developers, testers, and stakeholders.

1.2) Scope of this Document.
This document describes the operation, main function, design & implementation of the system. The CCPS will handle secure authorization, transaction processing, settlement, & reporting.

1.3) Overview.
The CCPS is a ~~secure~~, web-based solution aimed at managing credit card transactions between customers, merchants and banks.



2. General Description

The CCPS provides secure online handling of card payments, including transaction authorization, settlement, & account management for customers and merchants. It incorporates fraud detection, financial reporting, & transaction history.

3. Functional Requirements.

3.1) Transaction Management
- Authorize and process customer credit card transactions.
- Validate card details & available balance.

3.2) Account Management:
- Maintain customer/merchant profiles & transaction history.

3.3) Fraud Detection:
- Identify & block suspicious or duplicate transactions.

3.4) Billing and Settlement:
- Calculate service charges and transaction fees.
- Handle settlement between banks, merchants, & cardholders.

3.5) Reporting:
- Generate daily and monthly transaction reports.
- Provide audit logs for compliance.

4. Interface Requirements:

4.1) User Interface
- Simple dashboards for merchants and admins.
- Accessible via browser, android, ios application & POS terminal

4.2) Integration Interfaces:
- Integration with banking networks for transaction settlement.
- API support for e-commerce platforms & merchant systems.

5. Performance Requirements

- The system should authorize transactions within 2 seconds
- capable of processing at least 10,000 transaction an hour
- Ensure accurate transaction record across all modules.

6. Design Constraints.

6.1) Hardware Limitations:
- Should run on standard financial servers & POS terminals

6.2) Software Dependencies:
- Use a relational DBMS (eg. Oracle/mySQL)
- Implement secure frameworks such as Java Spring Boot or Node.js

7. Non-Functional Attributes:

7.1) Security:
- Enforce strong encryption, authentication, & ~~Assigned~~ compliance

7.2) Reliability:
- Ensure 99.9% uptime with redundant backups

7.3) Scalability:
- Support growth in transaction volume & new merchants

7.4) Portability:
- Accessible via multiple platforms & devices.

7.5) Reusability:
- Modern system design for upgrades & enhancements

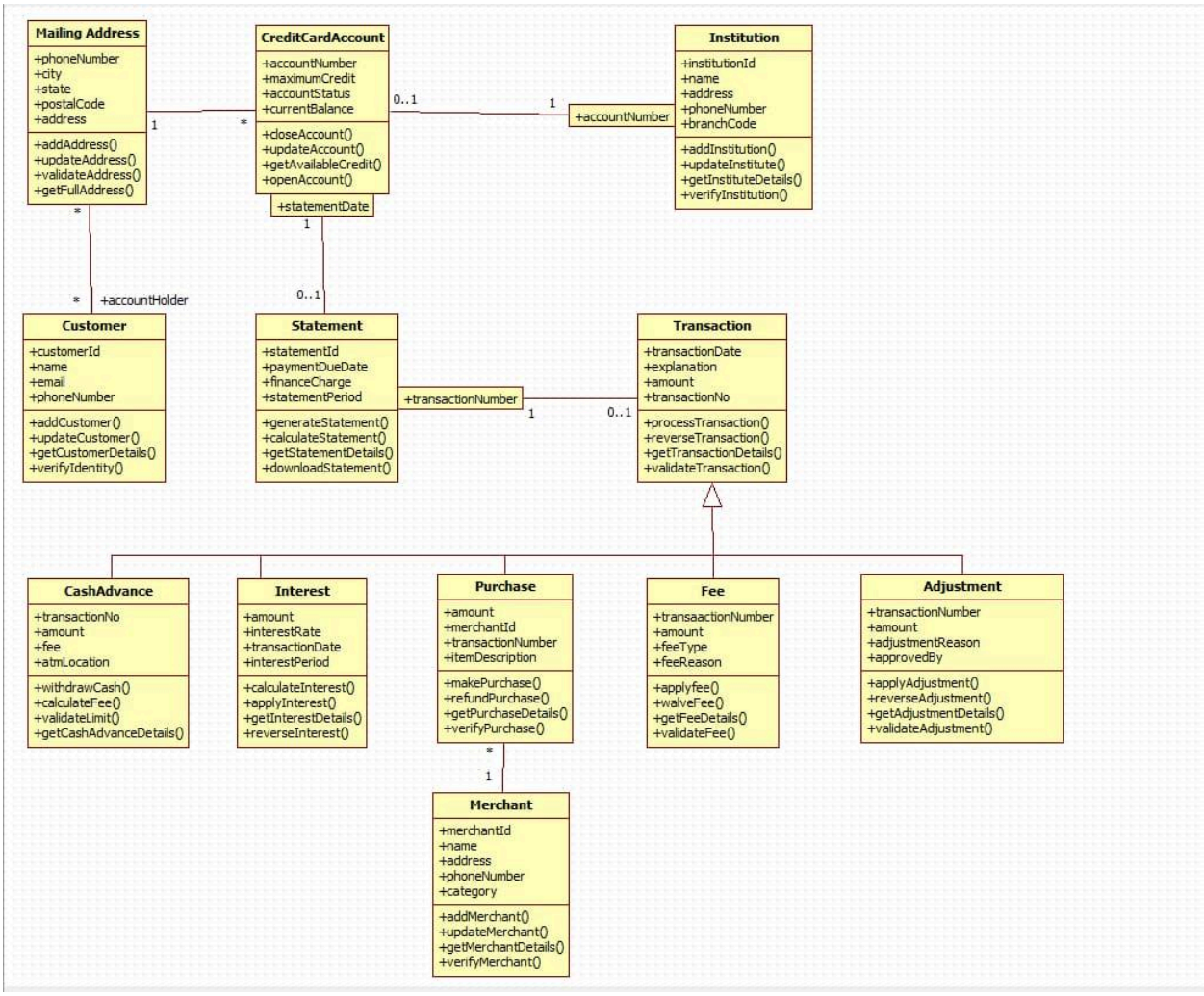
7.6) Compatibility:
- Compatible with major web browsers & payment APIs

7.7) Data Integrity:
- Maintain accurate & consistent financial records.

8. Preliminary Schedule and Budget

The app will be developed in 8 months, covering 5 modules with 20000 LoC/module, at $1/LoC.
The total budget is ₹100,000 including planning, development, integration, testing & development.

Class Diagram



Explanation

Core Architecture:

- Customer Class: Central entity with identity management and verification
- CreditCardAccount: Manages credit limits, balances, and account status
- Transaction: Processes financial operations with validation mechanisms
- Institution: Represents banking entities with branch management

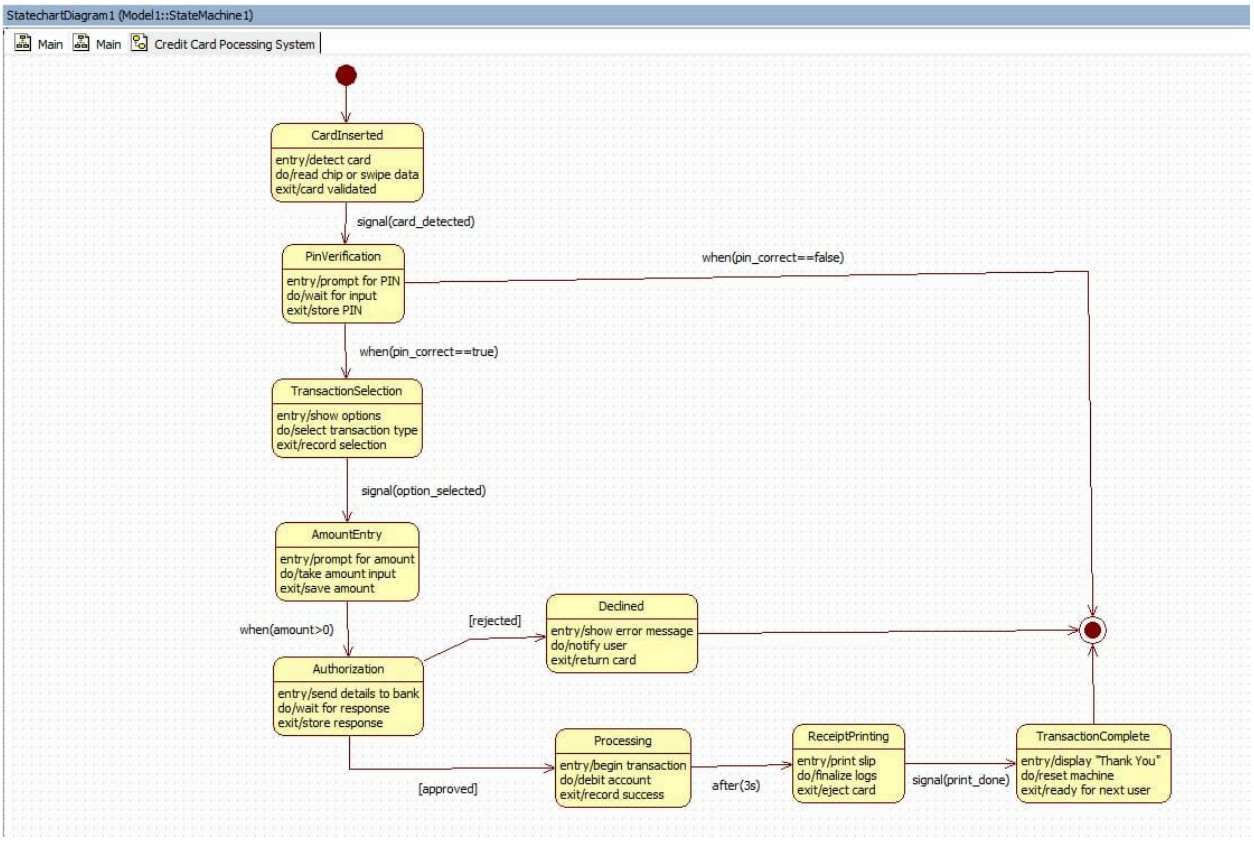
Key Relationships:

- 1-to-many: Customer to MailingAddress (multiple addresses)
- 1-to-many: CreditCardAccount to Statement (periodic statements)
- 1-to-1: Transaction to Account (individual transactions)

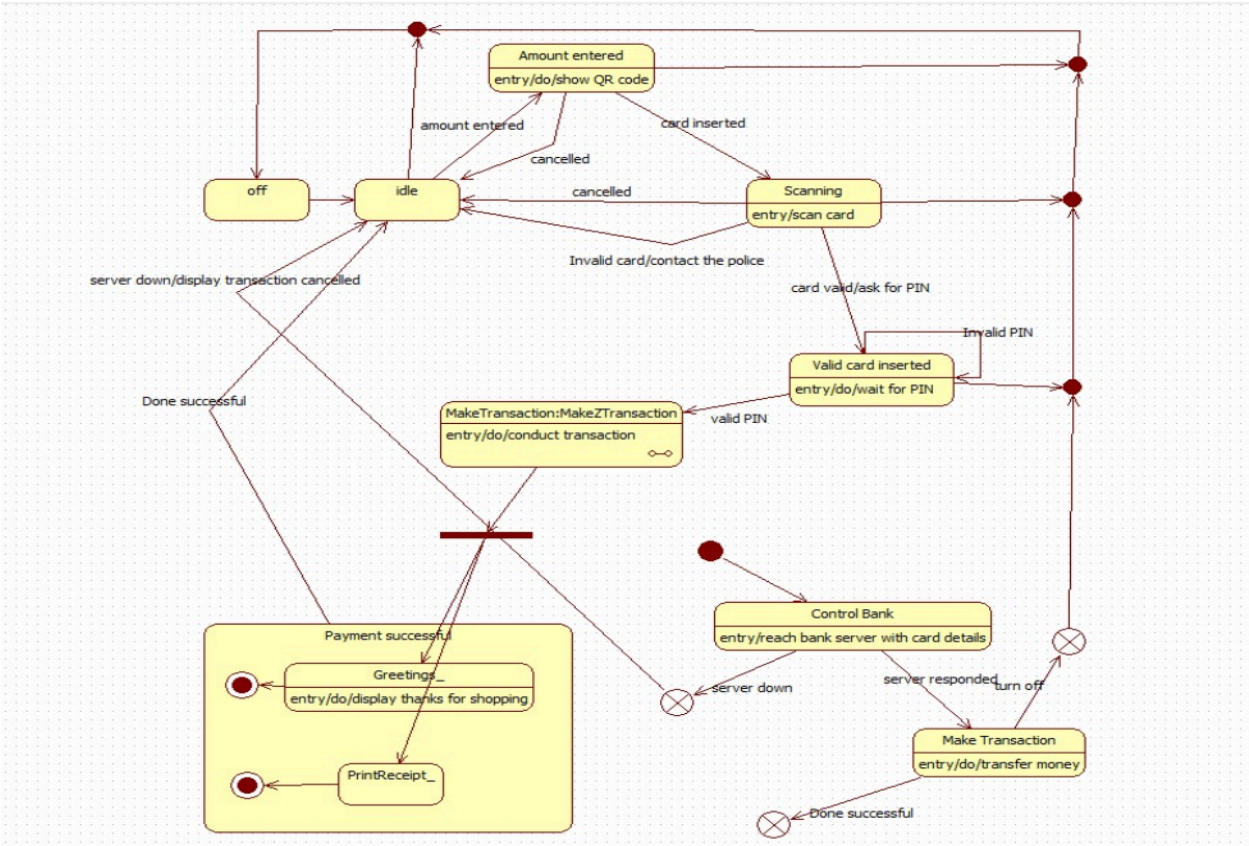
Business Logic:

- Address validation and geolocation services
- Credit availability calculations and limit management
- Statement generation with finance charge computations
- Institutional verification and branch code validation

State Diagram(Simple)



State Diagram(Advanced)



Simple State Diagram (Card Lifecycle & Payment Flow)

State Machine Workflow:

- Authentication States: CardInserted → PINVerification (secure validation)
- Transaction Setup: TransactionSelection → AmountEntry (user input phases)
- Processing Core: Authorization → [approved]/[rejected] (bank decision gate)
- Completion Cycle: Processing → ReceiptPrinting → TransactionComplete (success path)

Advanced State Diagram (ATM/Payment Terminal Transaction Flow)

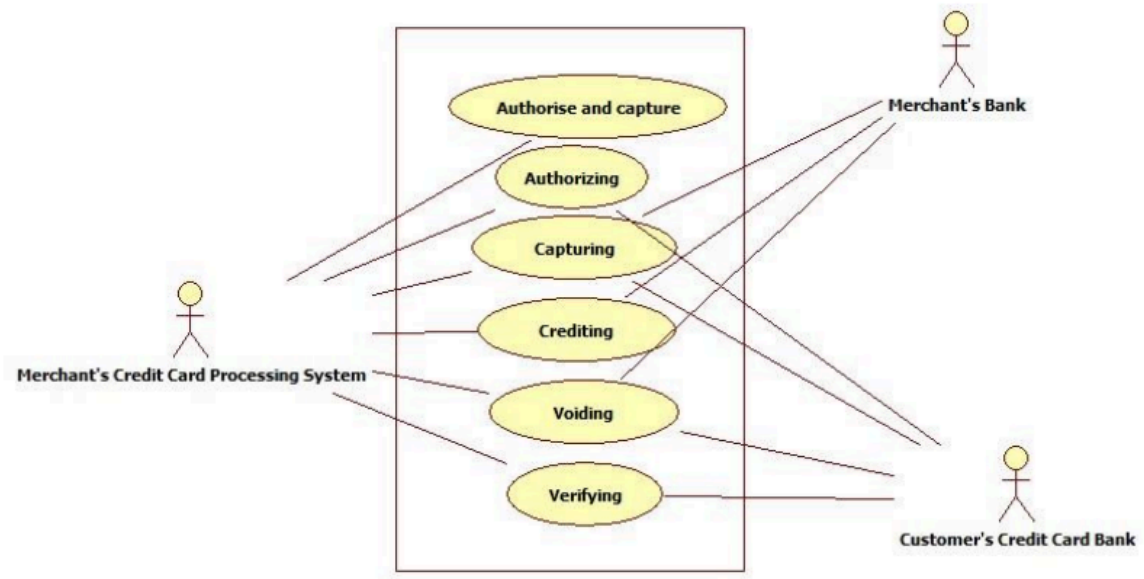
Key Transitions:

- Conditional guards: [PIN_correct==true], [amount>0]
- Timed events: after(3s) for processing delay
- System signals: card_detected, print_done triggering state changes

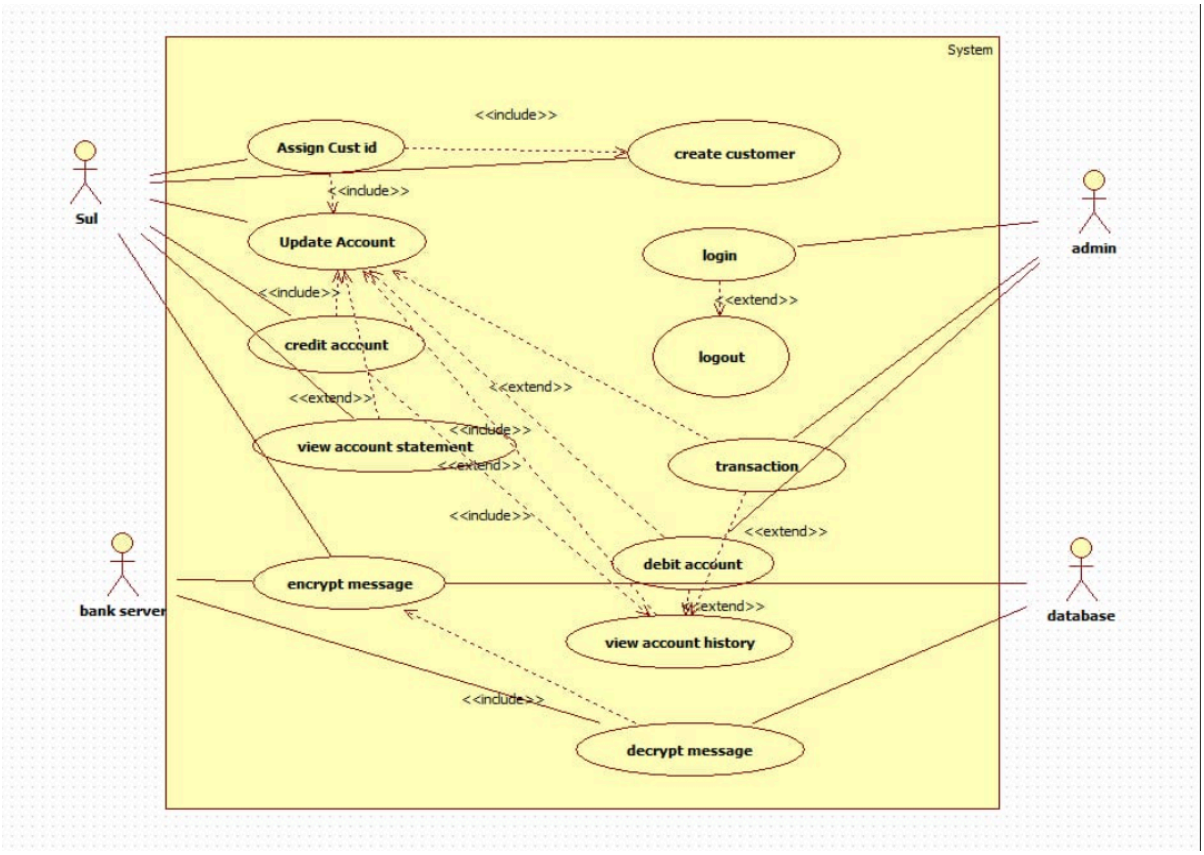
Error Handling:

- Dedicated Declined state for failed transactions
- Graceful degradation with user notification and card return
- System reset mechanism for next customer readiness

Use Case(Simple)



Use Case(Advanced)



Simple Use Case Diagram (Credit Card Processing System)

This use case diagram shows a banking system where customers can create accounts, login, view statements, and perform transactions. The system includes security features like encrypted messaging and allows administrators to manage accounts through a central database.

Actors & Core Functions:

- Customer: Performs login, transactions, views account history/statements
- Admin: Manages customer accounts and database operations
- System: Handles encryption/decryption and account processing

Advanced Use Case Diagram (Banking Customer Management System)

Key Relationships:

- Include Dependencies: Mandatory functions like updating accounts during customer creation
- Extend Relationships: Optional features like logout extending login, debit extending transactions
- Security Layer: Encrypted messaging integrated with account viewing functions

System Features:

- Account lifecycle management (create/update/credit/debit)
- Secure authentication with encrypted communication
- Comprehensive transaction history and statement generation
- Administrative oversight and database management

Scenarios

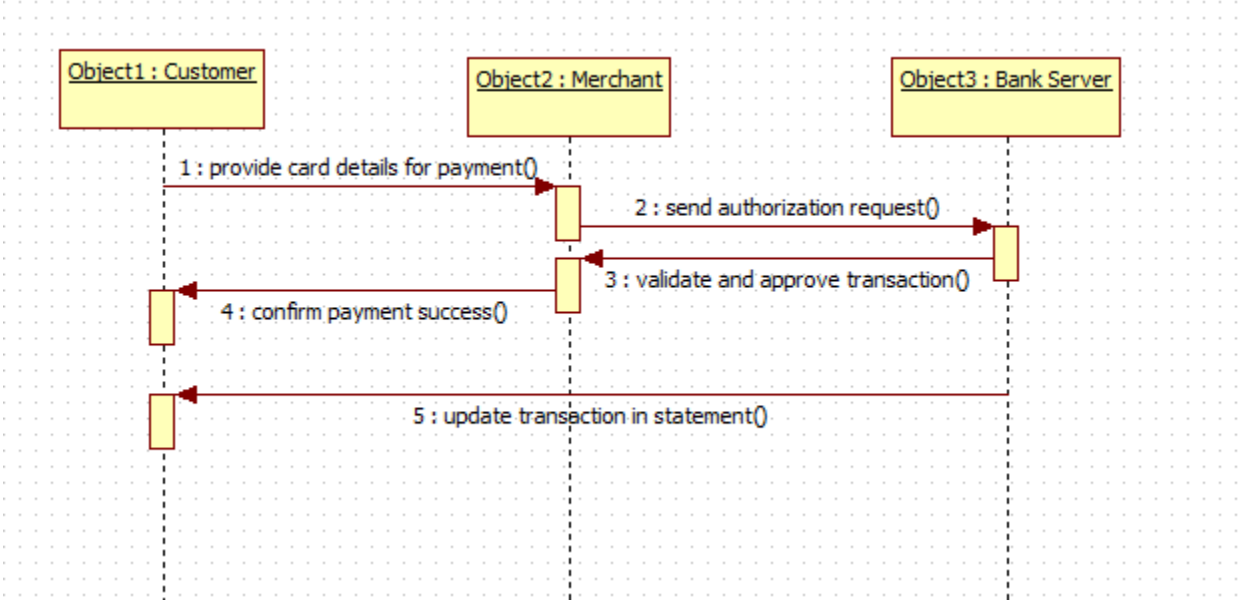
Scenario 1: Make Payment

1. The customer selects the **"Make Payment"** option.
2. The system displays the total bill amount.
3. The customer chooses a payment method (UPI, card, net banking, etc.).
4. The system sends the payment request to the payment gateway.
5. The customer enters the required payment details.
6. The payment gateway validates the information and processes the transaction.
7. On successful payment, the system updates the payment status.
8. A receipt is generated and displayed to the customer.

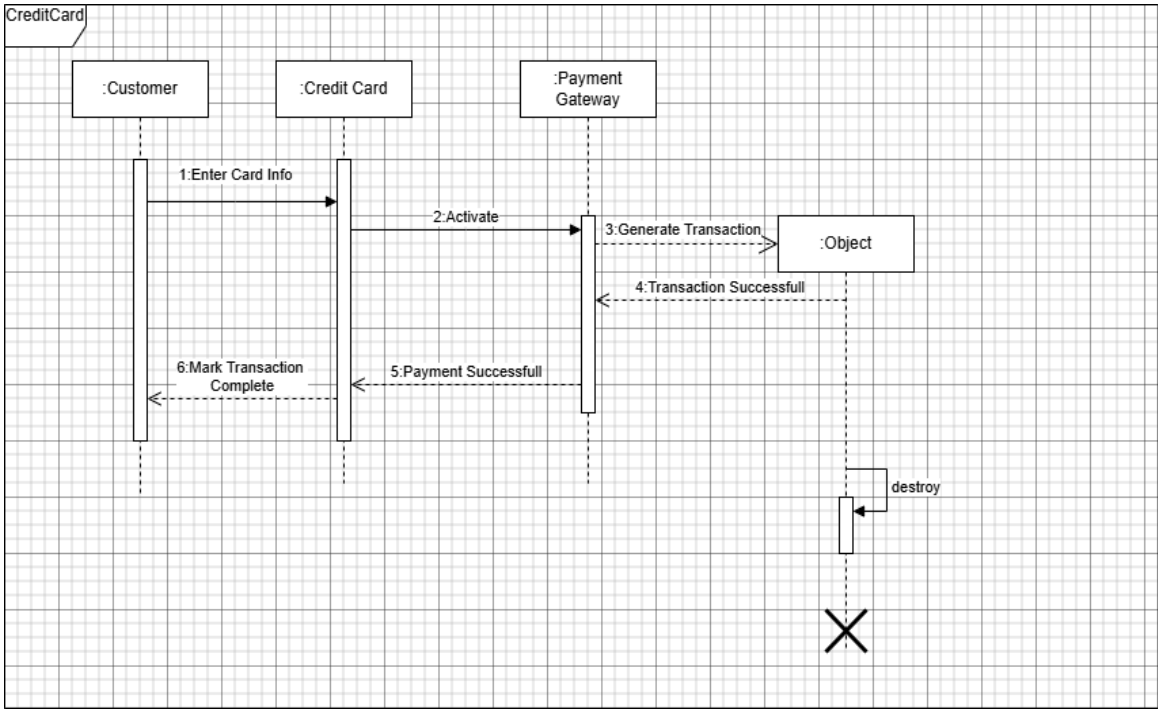
Scenario 2: Create Customer Account

1. The admin selects the **"Create Customer"** option.
2. The system displays a customer registration form.
3. The admin enters the name, ID proof, contact details, and address.
4. The system validates the entered information.
5. A unique customer ID is generated automatically.
6. The system creates the customer account and stores it in the database.
7. A confirmation message is shown indicating successful registration.

Sequence Diagram(Simple)



Sequence Diagram(Advanced)



Simple Sequence Diagram (Customer – Credit Card – Payment Gateway)

This sequence diagram shows a credit card payment process. The customer enters card information, which activates the payment system. A transaction is generated and processed, leading to successful payment confirmation. Finally, the transaction is marked complete and the session ends.

Process Flow:

- Initiation Phase: Customer input triggers card activation
- Processing Phase: Transaction generation and payment authorization
- Completion Phase: Success confirmation and resource cleanup

Advanced Sequence Diagram (Customer – Merchant – Bank Server)

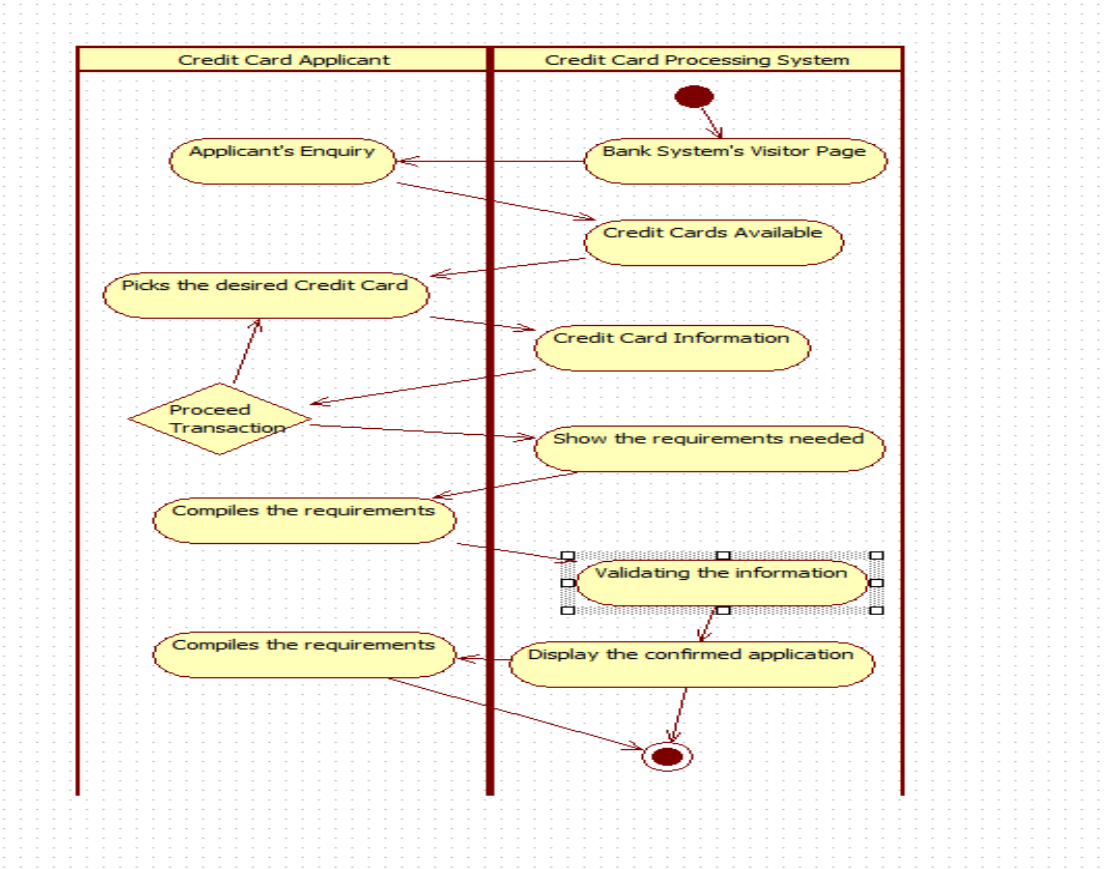
Key Interactions:

- Synchronous message passing between system components
- Object lifecycle management with creation and destruction events
- Sequential processing with dependency chains between steps

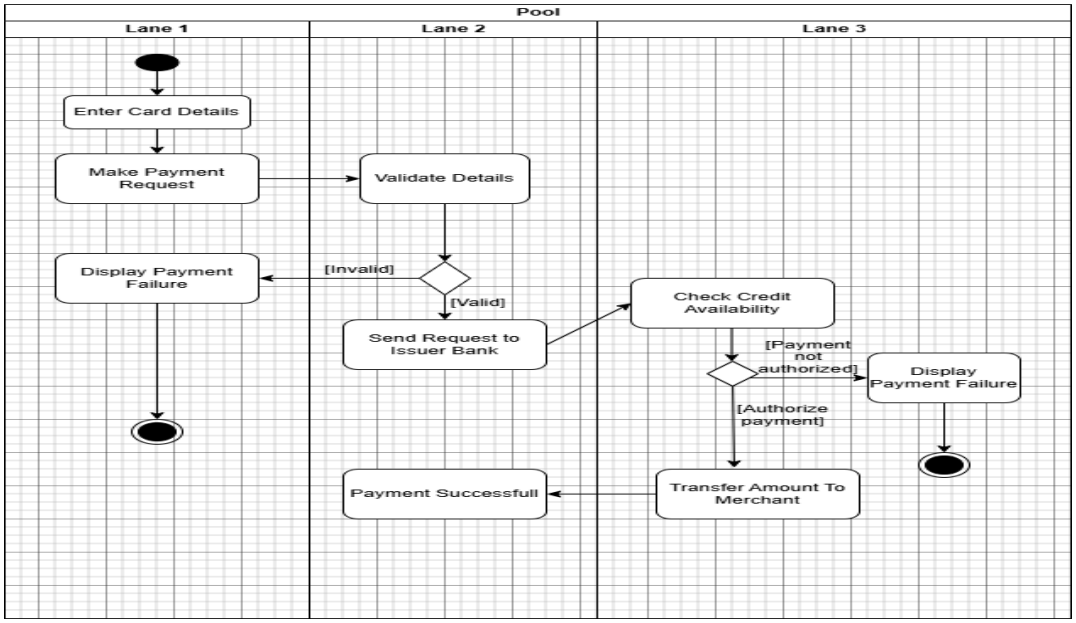
Technical Features:

- Transaction state management (generate → complete)
- Payment verification before transaction finalization
- System resource cleanup after process completion

Activity Diagram(Simple)



Activity Diagram(Advanced)



Simple Activity Diagram (Credit Card Payment Workflow)

This activity diagram shows the credit card payment process across three participants. The customer enters card details and makes a payment request. The system validates details and checks credit availability. If approved, the bank transfers funds to the merchant; if rejected, payment failure is displayed.

Advanced Activity Diagram (Credit Card Application & Processing System)

Swimlanes & Responsibilities:

- Lane 1 (Customer): Initiates payment request and views results
- Lane 2 (Payment System): Validates details and handles authorization
- Lane 3 (Bank): Checks credit availability and processes fund transfer

Decision Flow:

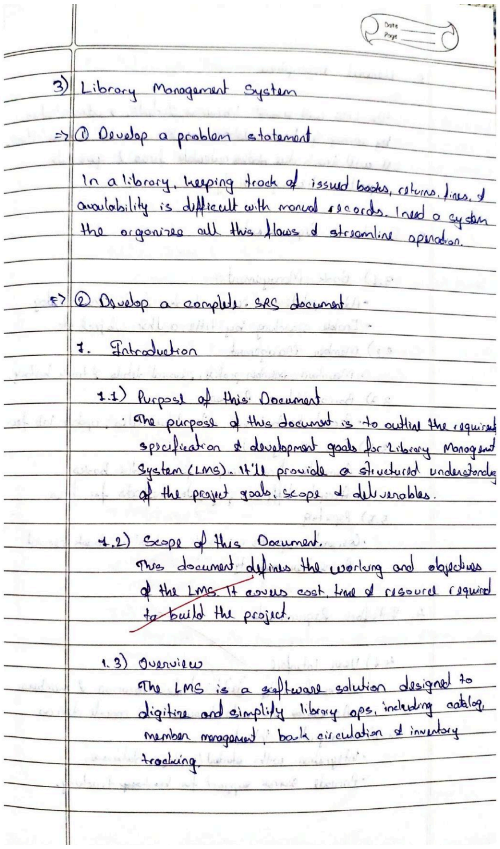
- Validation Gate: [Invalid] → Display Payment Failure
- Authorization Gate: [Payment not authorized] → Display Payment Failure
- Success Path: Valid & Authorized → Transfer Amount → Payment Successful

Business Logic:

- Sequential dependency: Validation → Credit Check → Authorization
- Failure handling at multiple checkpoints
- End-to-end transaction lifecycle management

3.Library Management System

SRS Document



3) Library Management System

=> ① Develop a problem statement

In a library, keeping track of issued books, returns, fines, & availability is difficult with manual records. I need a system the organizes all this flows & streamline operation.

=> ② Develop a complete SRS document

1. Introduction

1.1) Purpose of this Document

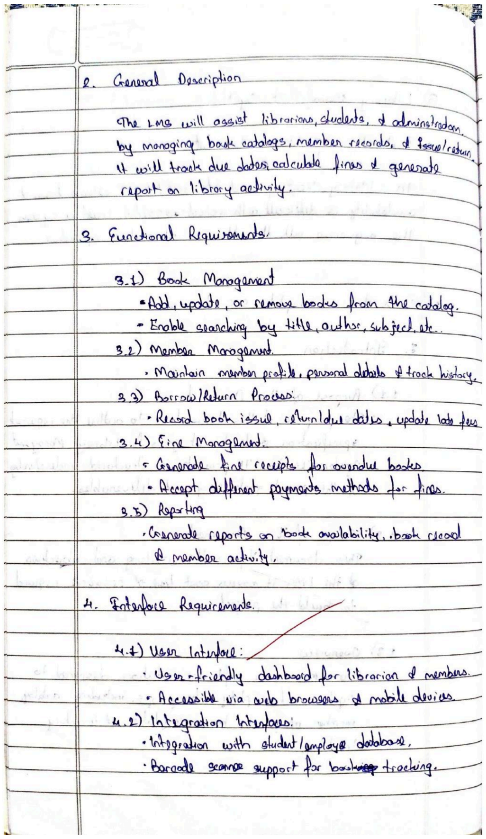
- The purpose of this document is to outline the required specification & development goals for Library Management System (LMS). It'll provide a structured understanding of the project goals, scope & deliverables.

1.2) Scope of this Document.

This document defines the working and objectives of the LMS. It covers cost, time & resource required to build the project.

1.3) Overview

The LMS is a software solution designed to digitize and simplify library ops, including catalog, member management, book circulation & inventory tracking.



2. General Description

The LMS will assist librarians, students, & administrators by managing book catalogs, member records, & issue/return. It will track due dates, calculate fines & generate report on library activity.

3. Functional Requirements

3.1) Book Management

- Add, update, or remove books from the catalog.
- Enable searching by title, author, subject, etc.

3.2) Member Management

- Maintain member profile, personal details & track history.

3.3) Borrow/Return Process:

- Record book issue, return date, & update late fees

3.4) Fine Management.

- Generate fine receipts for overdue books
- Accept different payments methods for fines.

3.5) Reporting

- Generate reports on book availability, book record & member activity.

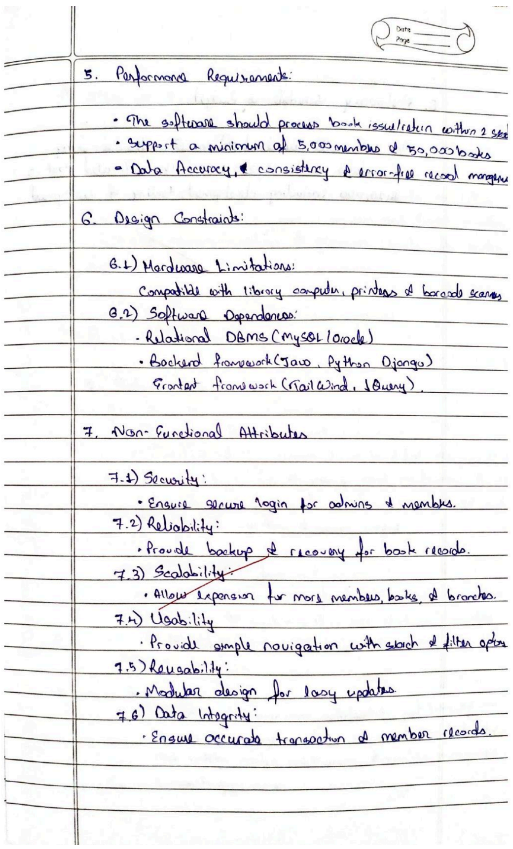
4. Interface Requirements

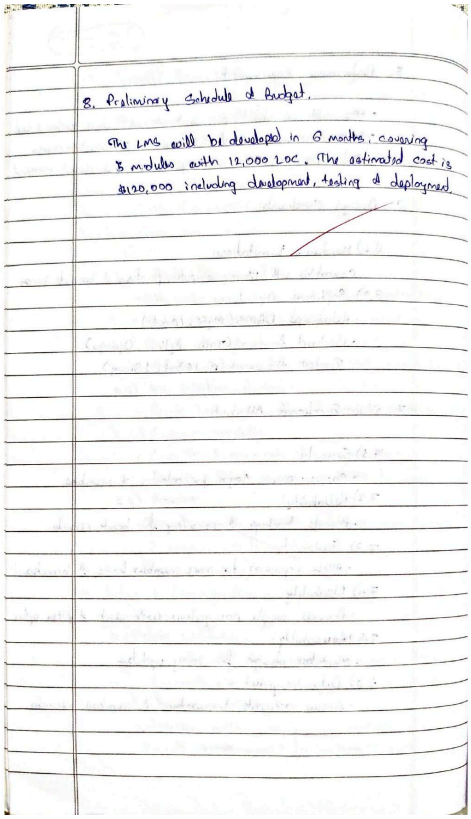
4.1) User Interface:

- User-friendly dashboard for librarian & members.
- Accessible via web browsers & mobile devices.

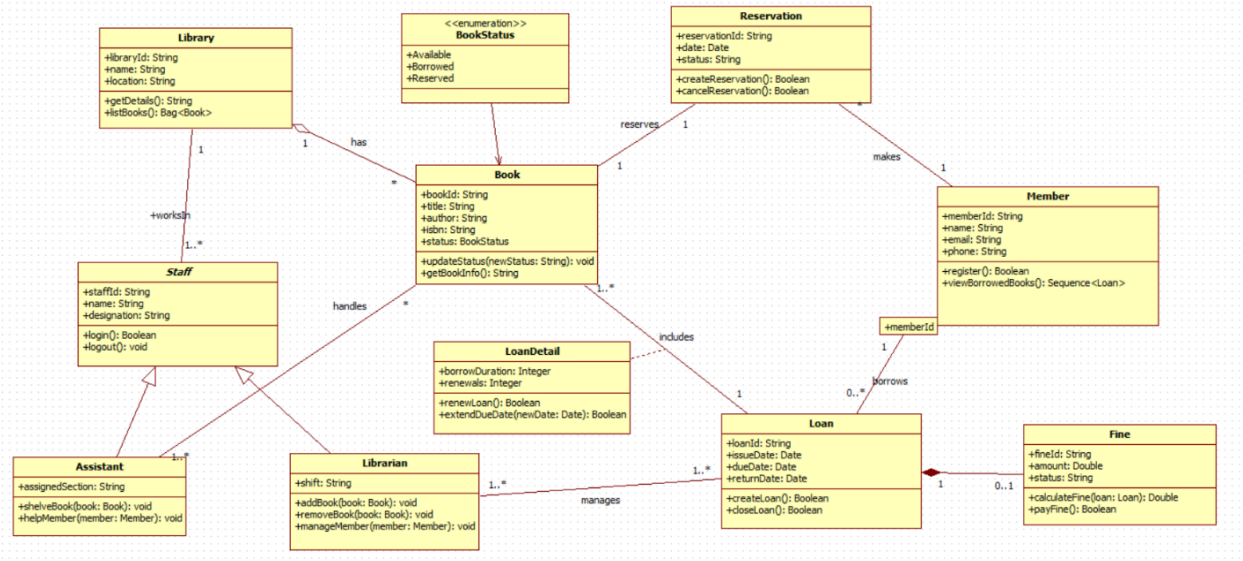
4.2) Integration Interfaces:

- Integration with student/employee database.
- Barcode scanner support for book tracking.





Class Diagram



Core Class Structure:

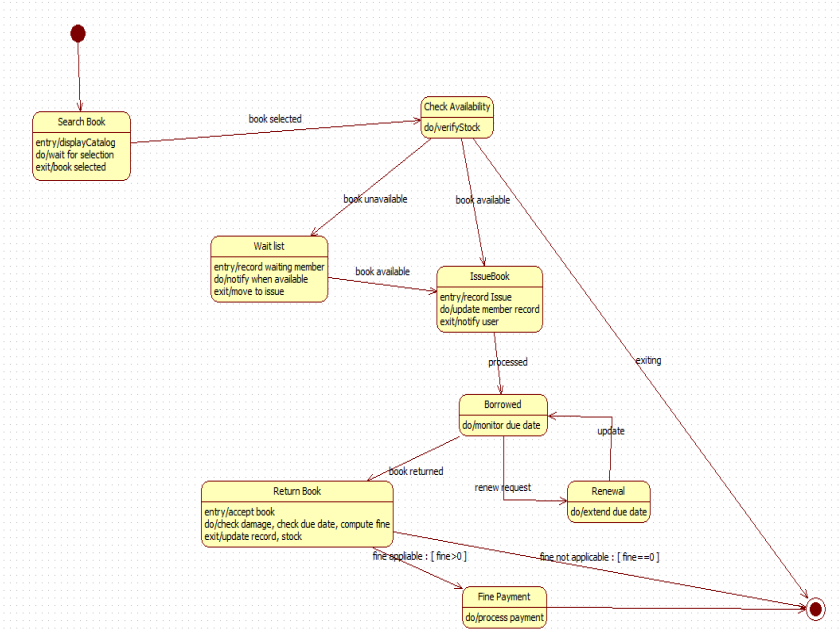
- Book Hierarchy: Base Book class specialized into Journals, Magazines, SubjectText
- User Management: Member superclass with Student/Faculty subclasses
- Transaction Processing: Librarian handles book issuance, returns, and fine calculation
- Financial Tracking: Fine class manages late return penalties and payment records

Key Relationships:

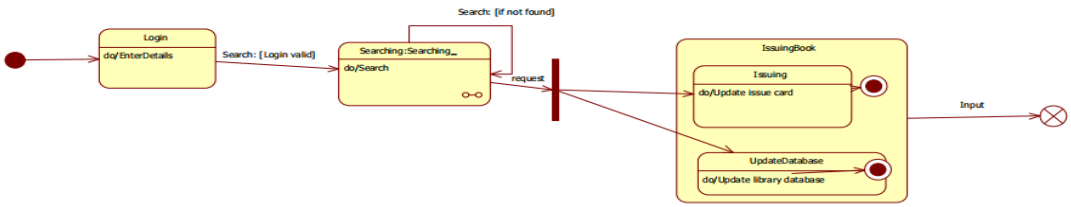
- Inheritance: Book → Journals/Magazines/SubjectText (catalog specialization)
- Association: Member → Transaction → Book (borrowing lifecycle)
- Dependency: Librarian → Fine → Member (penalty enforcement)

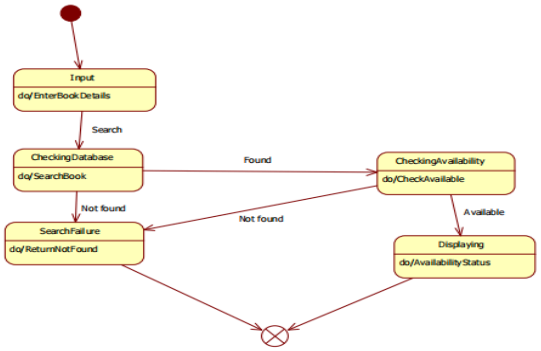
The system also keeps track of borrowing activities using the Transaction class, which stores the transaction ID, book ID, issue date, and due date. When members return books late, the Fine class is used to generate bills, update fines, and store payment information. Members pay these fines, and the librarian or system records them for future reference. Overall, the class diagram shows how different components—books, members, librarians, transactions, and fines—work together to ensure smooth operation of the library.

State Diagram(Simple)



State Diagram(Advanced)





Simple State Diagram

State Flow: Library Book Issue

- Login → Search Book
- If Not Found → Show Error → End
- If Found → Check Availability
- If Unavailable → Show Status → End
- If Available → Issue Book → Update Records → End

Advanced State Diagram

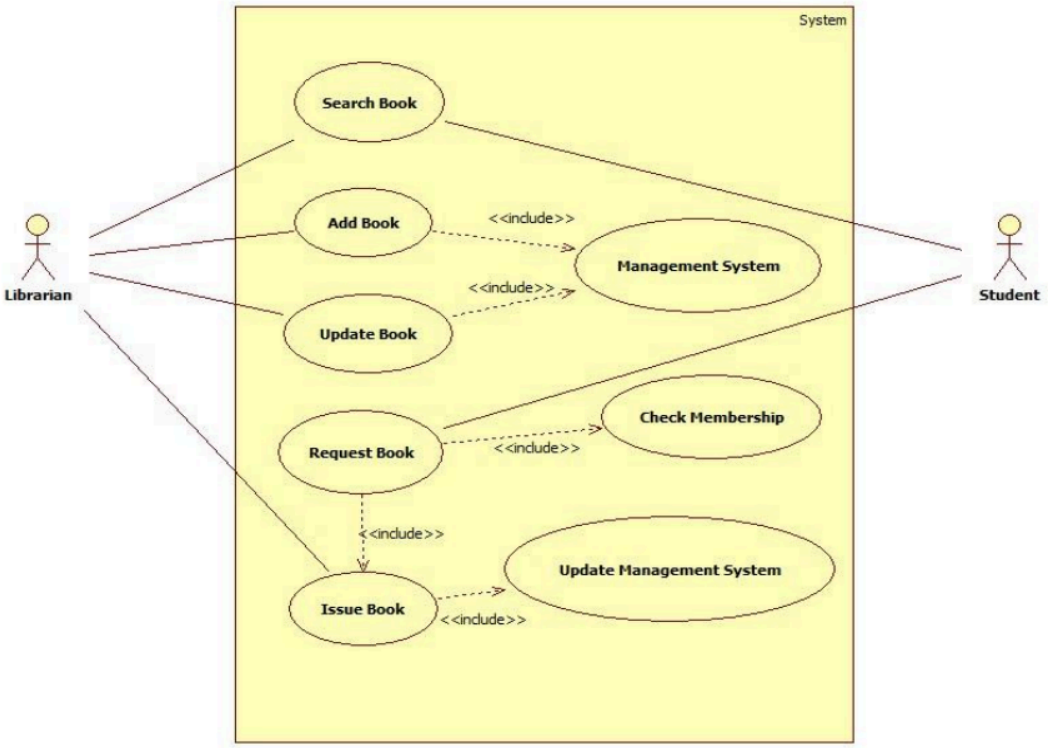
Diagram 1: Login & Book Issuing

- Authentication Flow: Login validation → Search initiation
- Transaction Processing: Successful search triggers book issuing workflow
- Data Persistence: Real-time database updates for issue cards and library records
- Decision Logic: Conditional branching based on search results and login validity

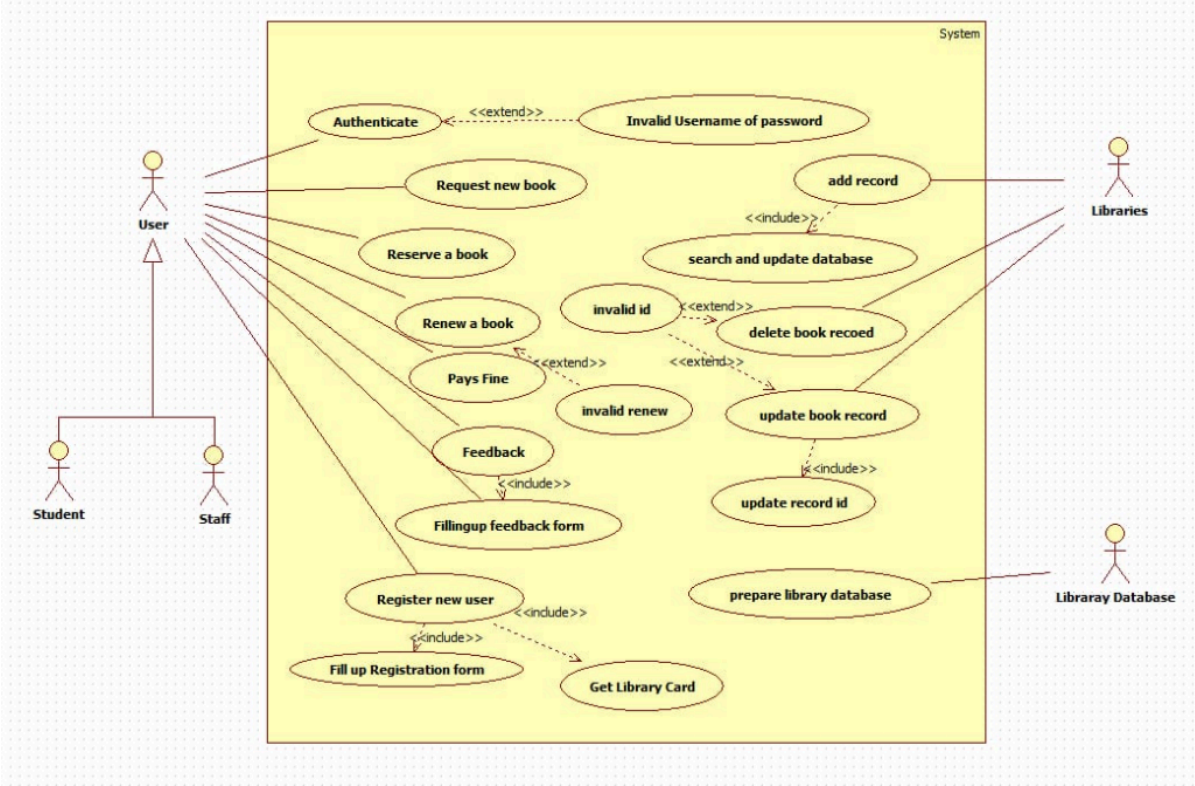
Diagram 2: Book Search Process

- Search Initiation: Book details input triggers database query
- Validation Layers:
    - Database existence check ([Found]/[Not found])
    - Availability verification ([Available]/[Not available])
- Output States:
    - Success path: Availability status display
    - Failure path: "Not found" return message
- System Actions: Automated database checking with status reporting

UseCase(Simple)



UseCase(Advanced)



Simple Use Case Diagram

Actors and Primary Goals:

- User: Interacts with core lending services (Reserve a Book, Renew a Book, Pay Fine) and provides system feedback.
- Librarian: Manages system data and user accounts (Register New User, Update Book Record, Prepare Library Database).

Advanced Use Case Diagram

Critical Relationships:

- «extend» Relationship: The "Invalid Username or Password" use case extends several primary user functions, providing a security and exception-handling mechanism.
- «include» Relationship: The "Add Record" use case mandatorily includes "Prepare Library Database," ensuring data integrity during updates.

System Integrity Features:

- Authentication check is a cross-cutting concern, extending multiple user-initiated use cases.
- Database update processes are modular, separating the preparation of data from the final record addition.

Scenario 1: Issue Book

1. The librarian selects the **“Issue Book”** option.
2. The librarian enters the **member ID** and **book ID**.
3. The system checks the member’s status and borrowing limit.
4. The system verifies whether the book is available for issue.
5. If all conditions are valid, the system updates the issue record and marks the book as **borrowed**.
6. The librarian hands the book to the member.

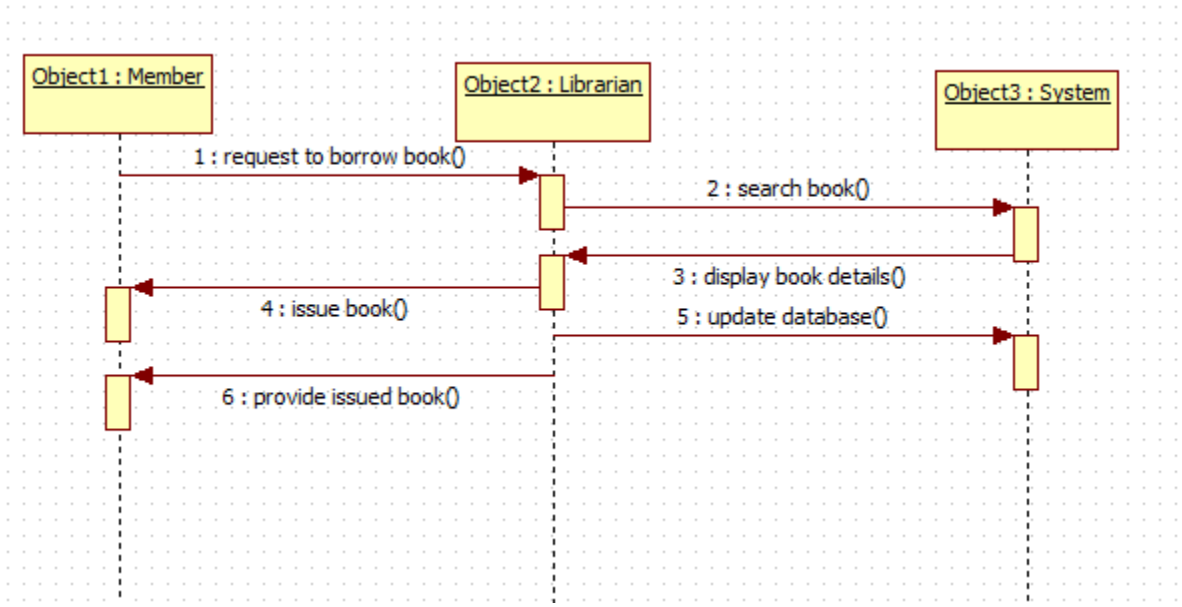
Scenario 2: Search Book

1. The student selects the **“Search Book”** option.
2. The system prompts the student to enter the book title, author name, or book ID.
3. The student enters the required search details.

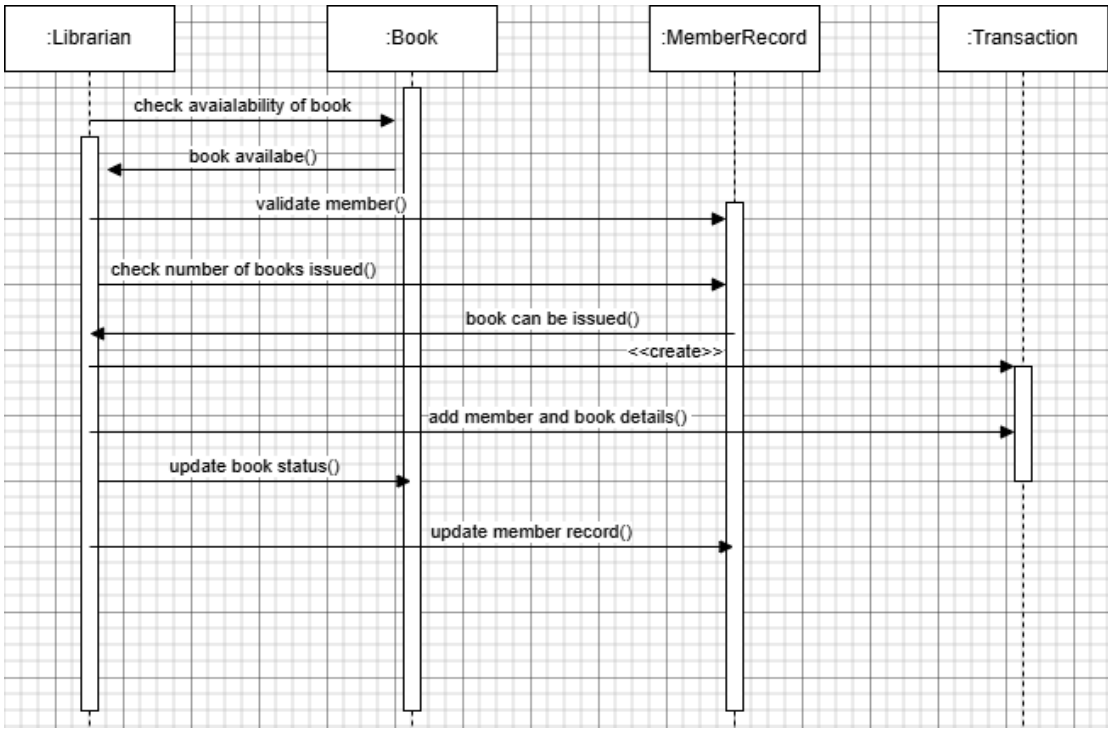
4. The system searches the database for matching records.

5. The system displays the result along with the book's availability status (**Available / Issued / Not Found**).

Sequence Diagram(Simple)



Sequence Diagram(Advanced)



Simple Sequence Diagram (Member – Librarian – System)

This sequence diagram shows the process of issuing a book in a library system. The Librarian checks if the book is available, validates the member, and checks how many books the member has already borrowed. If all checks pass, the book status is updated, the member's record is modified, and the transaction is completed.

Process Flow & Object Interaction:

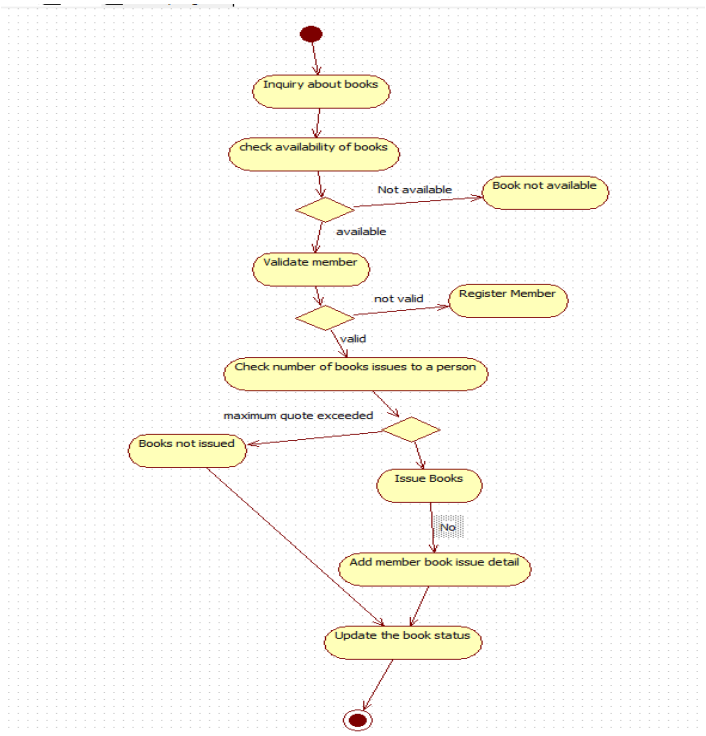
- Initialization Phase:
    - Librarian initiates the process by checking book availability
    - System returns book available() status
    - Member validation is performed through validate member()
- Validation Phase:
    - System checks borrowing limits via check number of books issued()
    - All validations must pass before proceeding to issuance
- Execution Phase:
    - Book object confirms issuable status with book can be issued()
    - Member and book details are added to the transaction
    - Member record is created/updated in the system
    - Book status is updated to reflect the issuance

Advanced Sequence Diagram (Librarian – Book – MemberRecord – Transaction)

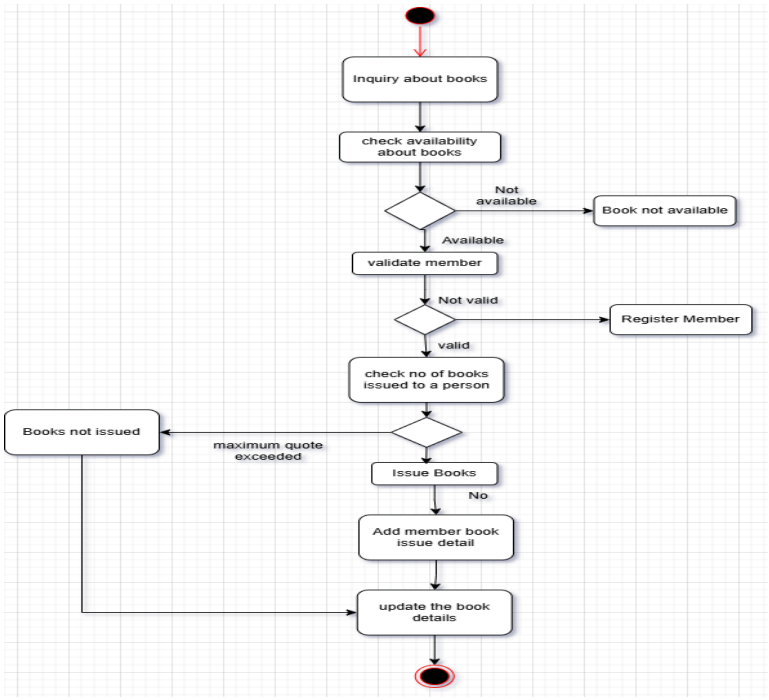
Key Technical Features:

- Sequential dependency between validation checks
- Object creation for member record tracking
- State changes across multiple system entities
- Coordinated updates between Book and MemberRecord objects

Activity Diagram(Simple)



Activity Diagram(Advanced)



Simple Activity Diagram

This activity diagram shows the library book issuing workflow. It starts with a book inquiry and checks availability. If available, the member is validated. A valid member's book quota is checked, and if within limit, the book is issued, member details are updated, and book records are modified.

- Sequential flow with multiple decision gates
- Exception handling for unavailable books and invalid members
- Member registration sub-process within main workflow
- Data updates to both member records and book inventory
- Quota management to enforce borrowing limits

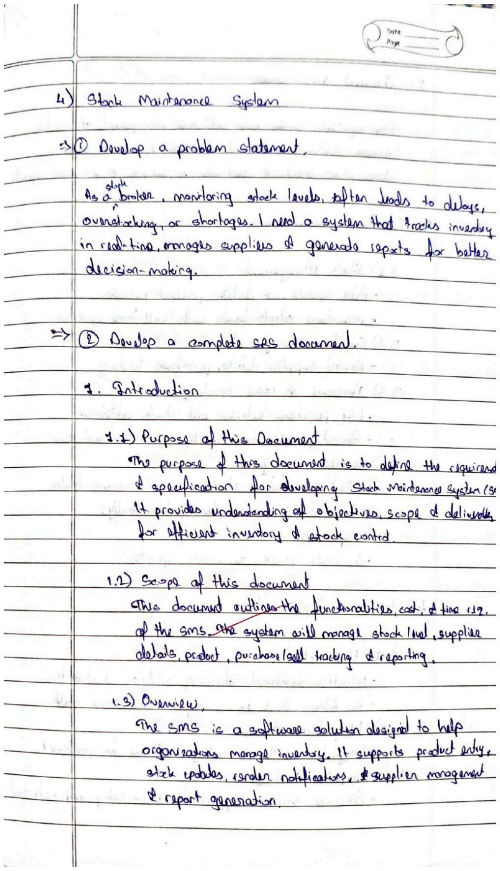
Advanced Activity Diagram

Swimlanes & Control Flow:

- Initialization:
    - Activity starts with "Inquiry about books"
    - System performs "check availability about books"
- Decision Nodes:
    - Book Availability: [Not available] → End process / [Available] → Continue
    - Member Validation: [Not valid] → "Register Member" / [Valid] → Continue
    - Quota Check: [Maximum quota exceeded] → End process / [Books not issued] → Continue
- Execution Path:
    - "Issue Books" → "Add member book issue detail" → "Update the book details"

4. Stock Maintenance System

SRS Document:



4) Stock Maintenance System

=> ① Develop a problem statement.

As a ~~stock~~ broker, monitoring stock levels often leads to delays, overstocking, or shortages. I need a system that tracks inventory in real-time, manages suppliers & generates reports for better decision-making.

=> ② Develop a complete SRS document.

1. Introduction

1.1) Purpose of this Document

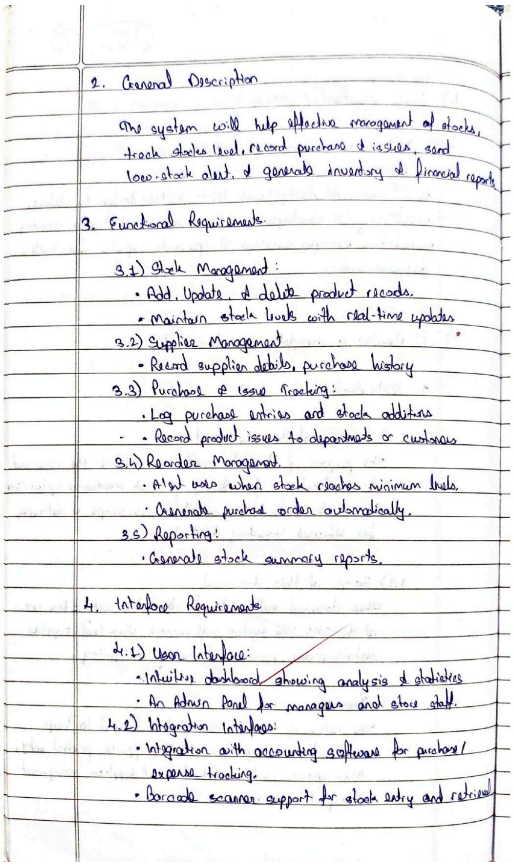
The purpose of this document is to define the requirement & specification for developing Stock maintenance system (SMS). It provides understanding of objectives, scope & deliverables for efficient inventory & stock control.

1.2) Scope of this document

This document outlines the functionalities, cost, & time req. of the SMS. The system will manage stock level, supplier details, product, purchase/sell tracking & reporting.

1.3) Overview

The SMS is a software solution designed to help organizations manage inventory. It supports product entry, stock updates, reorder notifications, & supplier management & report generation.



2. General Description

The system will help effective management of stocks, track stocks level, record purchase & issues, send low-stock alert, & generate inventory & financial reports.

3. Functional Requirements

3.1) Stock Management:
- Add, Update, & delete product records.
- Maintain stock levels with real-time updates

3.2) Supplier Management
- Record supplier details, purchase history

3.3) Purchase & issue tracking:
- Log purchase entries and stock additions
- Record product issues to departments or customers

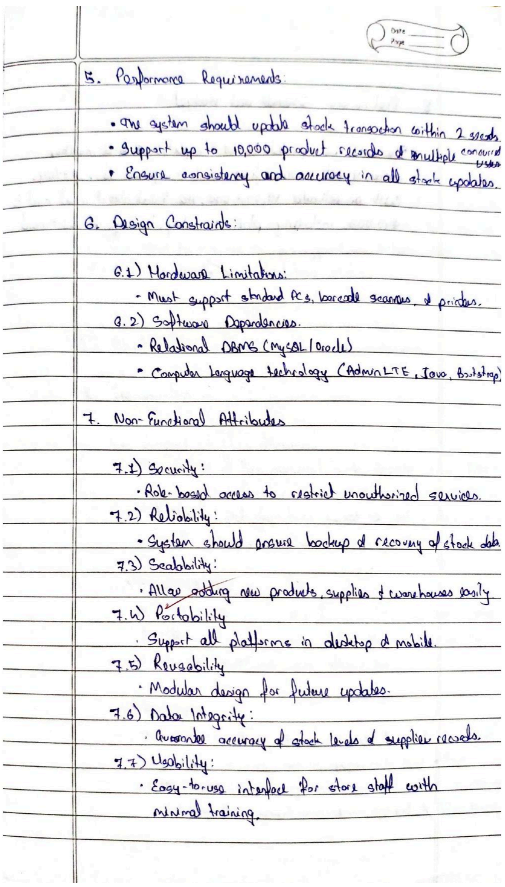
3.4) Reorder Management.
- Alert user when stock reaches minimum levels.
- Generate purchase order automatically.

3.5) Reporting:
- Generate stock summary reports.

4. Interface Requirements

4.1) User Interface:
- Intuitive dashboard showing analysis & statistics
- An Admin Panel for managers and store staff.

4.2) Integration Interfaces:
- Integration with accounting software for purchase/ expense tracking.
- Barcode scanner support for stock entry and retrieval



5. Performance Requirements

- The system should update stock transaction within 2 secs
- Support up to 10,000 product records & multiple concurrent users
- Ensure consistency and accuracy in all stock updates.

6. Design Constraints:

6.1) Hardware Limitations:
- Must support standard PCs, barcode scanners, & printers.

6.2) Software Dependencies:
- Relational DBMS (MySQL/Oracle)
- Computer language technology (AdminLTE, Java, Bootstrap)

7. Non-Functional Attributes

7.1) Security:
- Role-based access to restrict unauthorized services.

7.2) Reliability:
- System should ensure backup & recovery of stock data

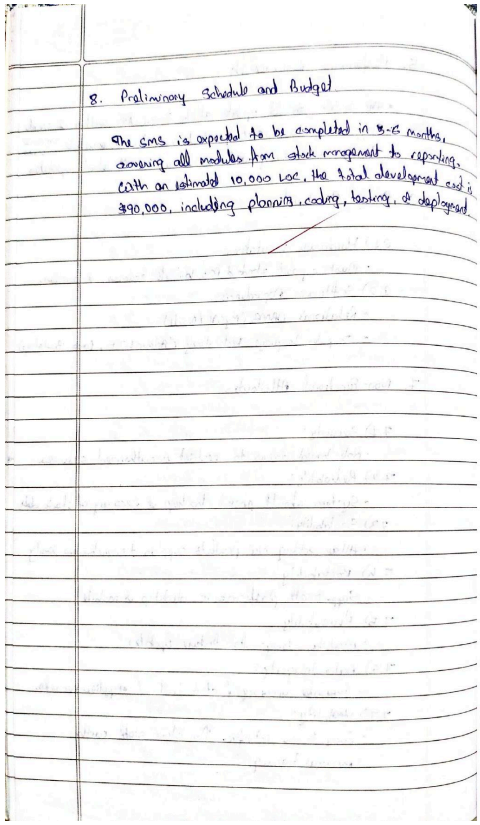
7.3) Scalability:
- Allow adding new products, supplies & warehouses easily

7.4) Portability
- Support all platforms in desktop & mobile.

7.5) Reusability
- Modular design for future updates.

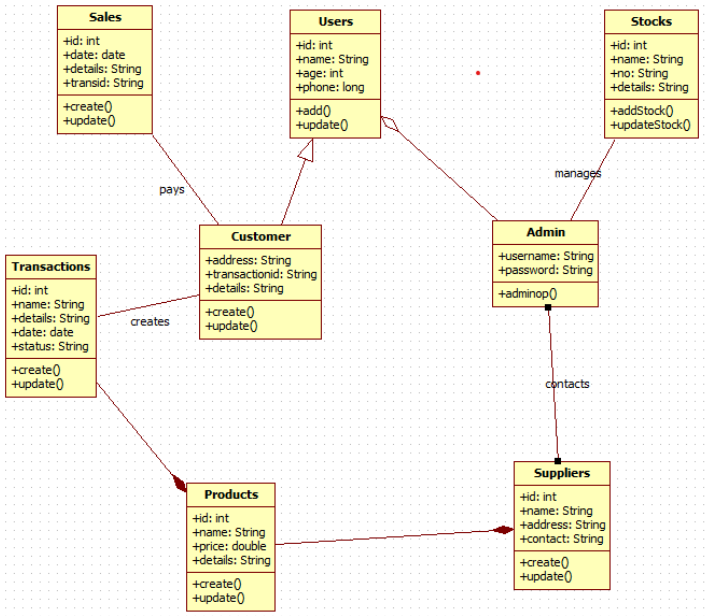
7.6) Data Integrity:
- Guarantee accuracy of stock levels & supplier records.

7.7) Usability:
- Easy-to-use interface for store staff with minimal training.



8. Preliminary Schedule and Budget

The SMS is expected to be completed in 3-6 months, covering all modules from stock management to reporting. With an estimated 10,000 LOC, the total development cost is $90,000, including planning, coding, testing, & deployment



Class Diagram

Explanation:

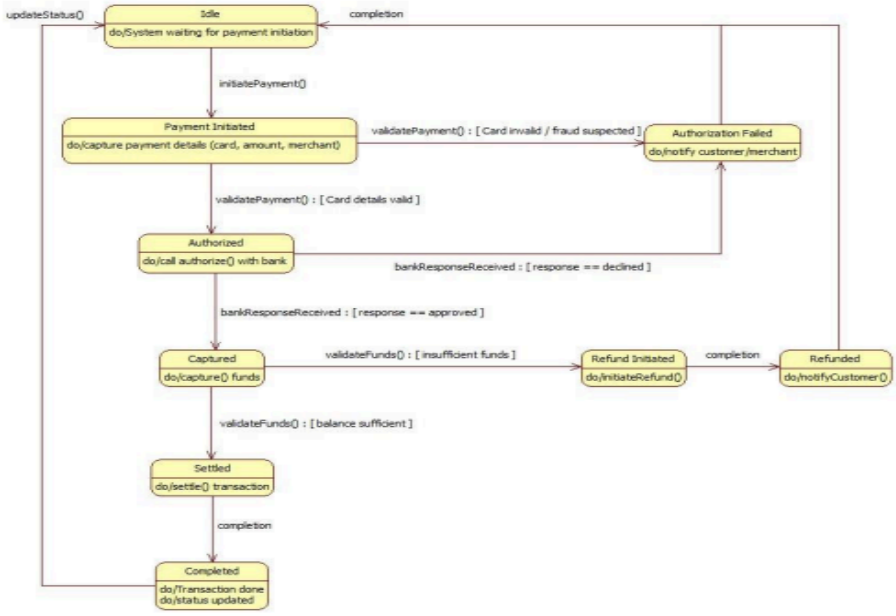
Core Entities:

- Person (Base Class): Contains organization, address with login() and updateProfile() methods
- Customer & Employee: Specialized person classes with role-specific attributes
- Stock: Manages inventory with performanceRating, score and updateRating() capabilities
- Purchase: Handles transactions with quantity, purchaseDate and sale() operations

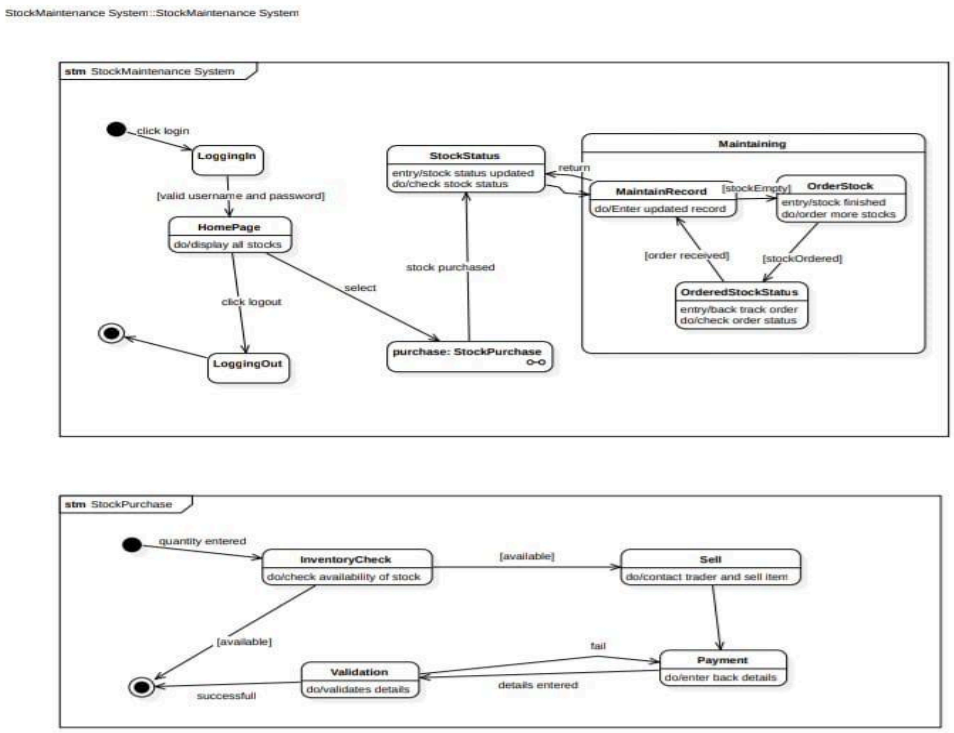
Key Features:

- Inheritance hierarchy with Person as superclass
- Stock performance tracking with rating system
- Purchase transaction management
- Role-based differentiation between customers and employees
- Profile management and authentication system

State Diagram



Advanced State Diagram



Simple State Diagram:

These state diagrams show a stock maintenance system workflow. Users login, view stocks, and can purchase items. The system checks inventory availability, processes payments, and handles stock ordering when inventory is low. It also tracks order status and maintains records.

Advanced State Diagram:

Diagram 1: Main System Workflow

- Authentication Flow: Login → HomePage → Logout
- Stock Management:
    - Purchase process triggers StockPurchase state
    - Record maintenance with real-time updates

- Order Management:
    - OrderedStockStatus tracks order progress
    - Automatic reordering when [stockEmpty]
    - Status transitions: [stockOrdered] → [stockReceived]

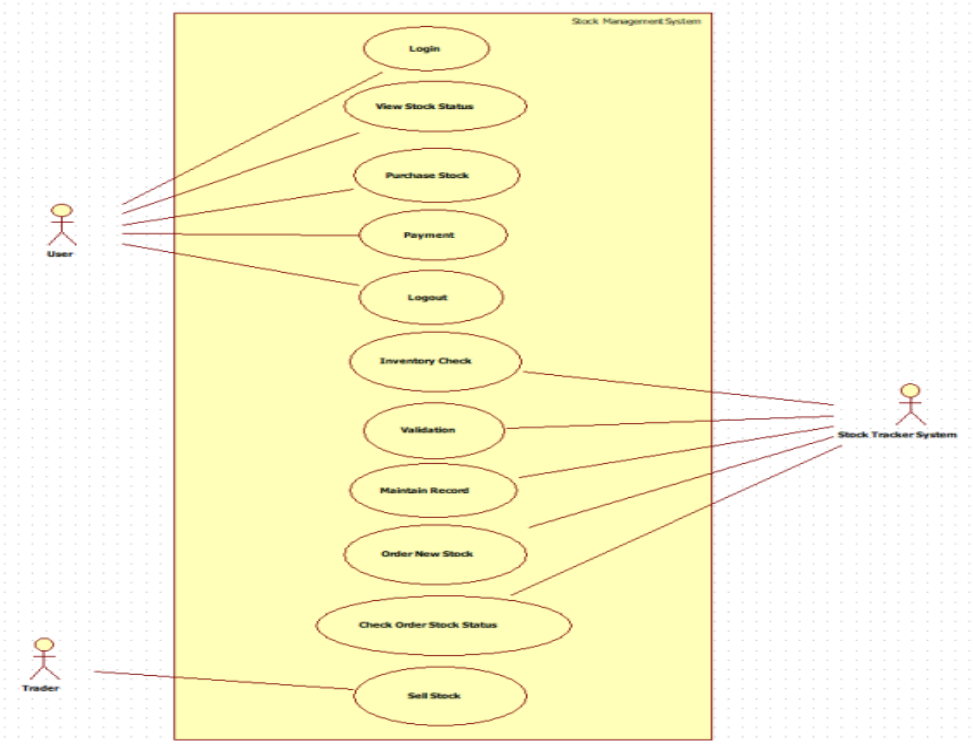
Diagram 2: Purchase Process

- Inventory Validation:
    - Quantity entry → InventoryCheck → [available]/[not available]
    - Dual validation layers for stock availability
- Transaction Processing:
    - Payment state with detail validation
    - Sell state coordinates with traders
    - Fail state handles transaction errors

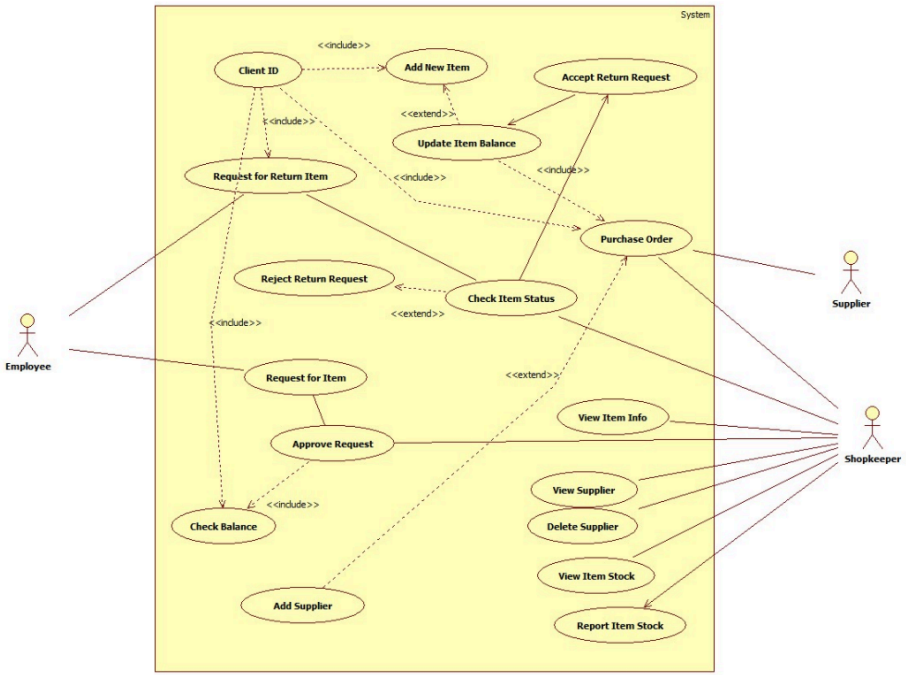
Key Features:

- Real-time inventory monitoring
- Automated stock replenishment
- Multi-layer validation for transactions
- Order tracking and status management
- Trader integration for sales execution

Simple Use Case:



Advanced Use Case:



Scenario 1: Successful Purchase and Stock Update (Happy Path)

- Purchase department raises a purchase requisition for 500 units of Item-X
- Supplier accepts PO and delivers 500 units with proper invoice
- Warehouse staff receives goods, performs quality check → Passed
- Staff scans/enters GRN (Goods Receipt Note) into the system
- System automatically increases stock level of Item-X by 500
- Stock value updated in inventory ledger
- Finance receives GRN copy and processes supplier payment
- Re-order level alert for Item-X automatically turns off
- Dashboard shows updated stock: Item-X = 1,200 units

Scenario 2: Stock Rejection and Return (Exception Scenario)

- Supplier delivers 300 units of Item-Y, but 80 units found damaged
- Warehouse staff performs quality check → 80 units Failed
- Staff creates Partial GRN for 220 units only
- System adds only 220 units to stock, blocks 80 units
- Return Material Authorization (RMA) generated automatically
- Rejected 80 units returned to supplier with Debit Note
- Stock level of Item-Y updated correctly (only +220)
- Supplier acknowledges return, issues Credit Note
- Inventory report reflects accurate stock and pending returns
- Low-stock alert remains active since actual receipt is below requirement

Simple Use case:

This use case diagram shows a stock management system with Storekeeper and Supplier actors. Storekeepers manage items, handle return requests, check stock, and generate reports. Suppliers fulfill purchase orders and provide item information. The system includes item balance updates and return request processing.

System Features:

- Inventory lifecycle management (addition to deletion)
- Return request workflow with approval/rejection
- Real-time stock monitoring and balance checks
- Supplier relationship management
- Comprehensive reporting capabilities

Advanced Use Case:

Actors & Core Functions:

Storekeeper

- Manages inventory (Add New Item, Update Item Balance, View Item Stock)
- Processes returns (Access Return Request, Reject Return Request)
- Generates reports (Report Item Stock)
- Handles supplier data (Add Supplier, View Supplier, Delete Supplier)

Supplier

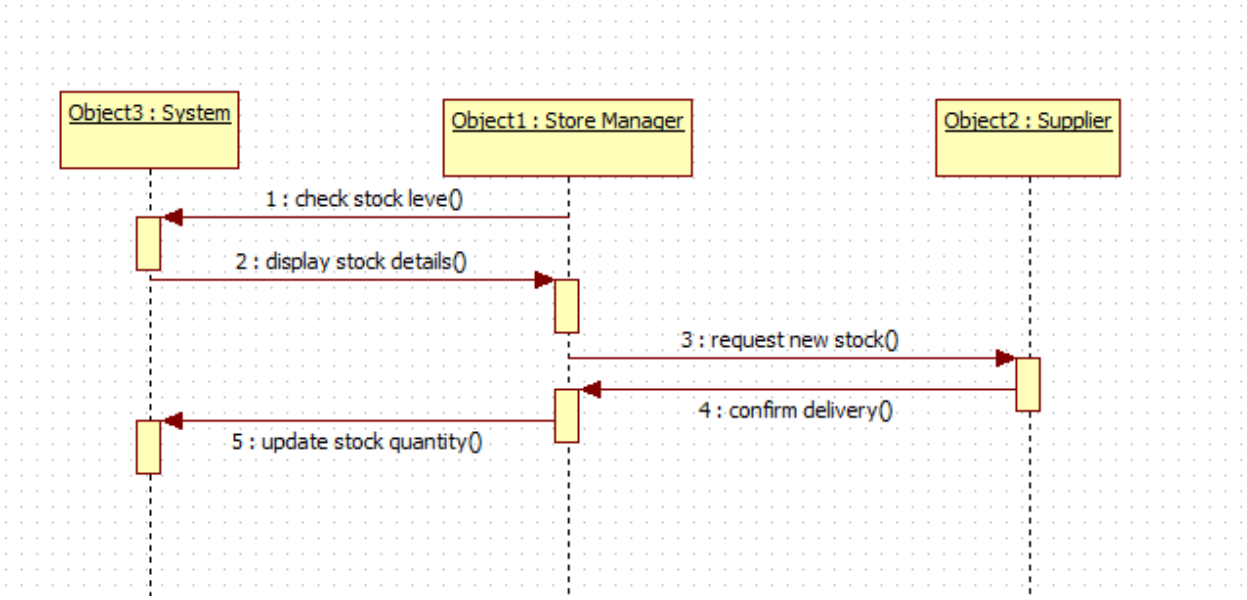
- Fulfills Purchase Orders
- Provides product information (View Item Info)

Key Relationships:

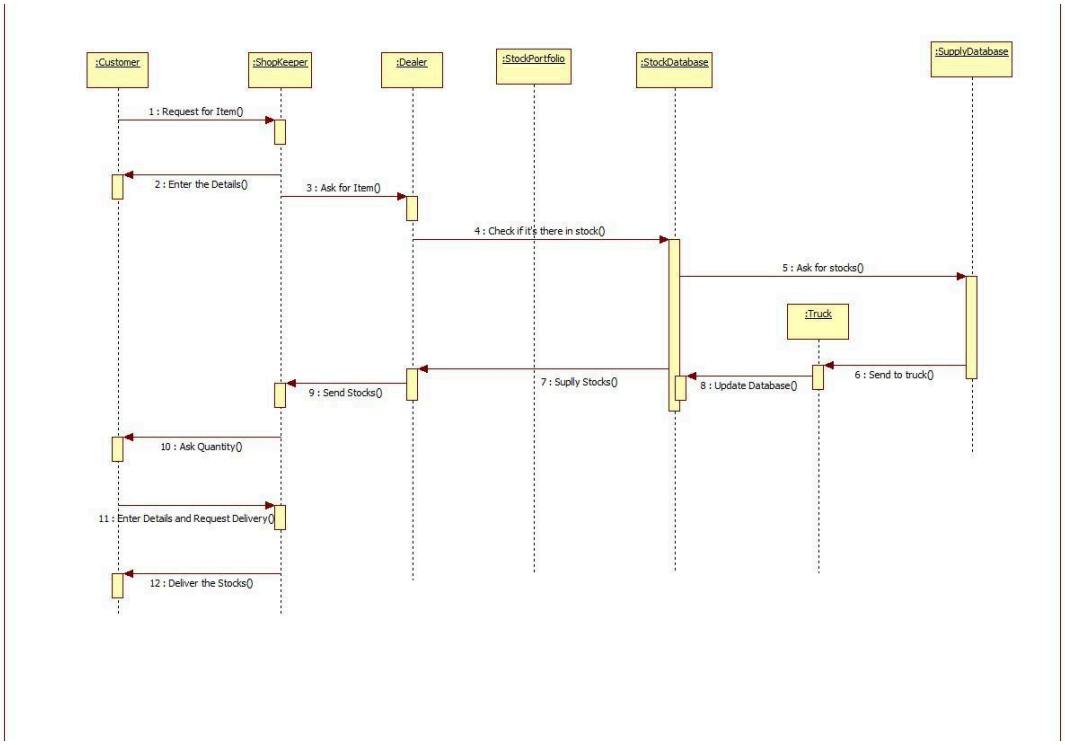
- «include»: Mandatory dependencies like Client ID verification during item addition

- «extend»: Optional functionalities like return request handling extending main processes

Simple Sequence Diagram:



Advanced Sequence Diagram:



Simple Sequence:

This sequence diagram shows a stock maintenance process where a customer requests an item. The system checks stock availability, arranges delivery from the warehouse, updates the database, and completes the delivery to the customer.

Key Technical Features:

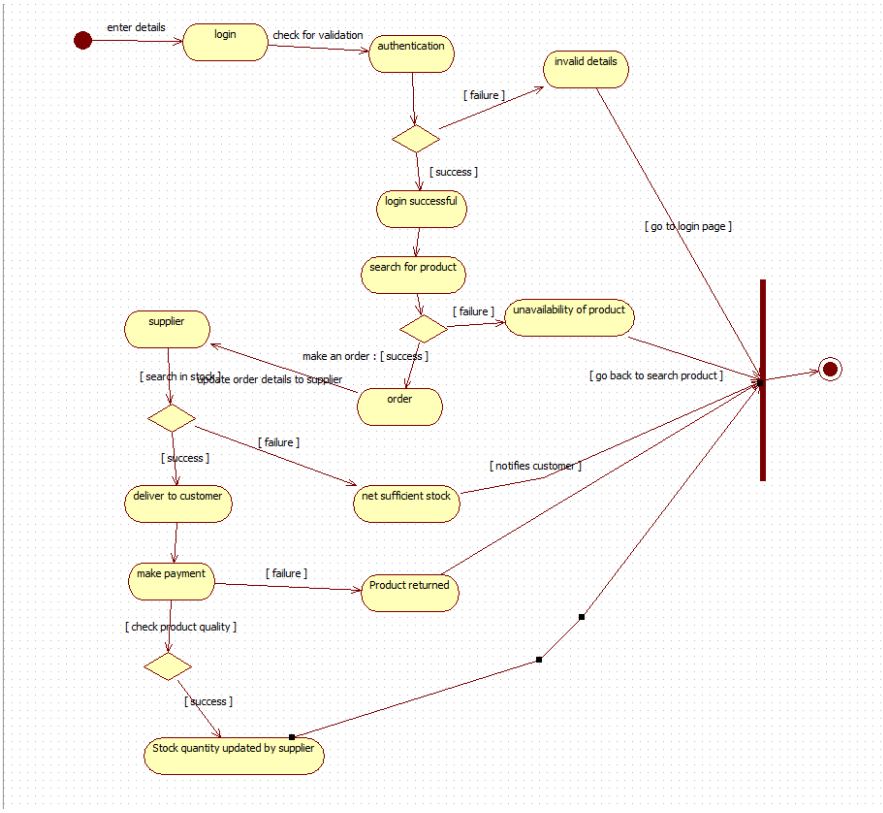
- Sequential dependency between request and fulfillment stages
- Real-time inventory checking and database synchronization
- Logistics integration for stock delivery coordination
- Customer interaction throughout the entire process lifecycle

Advanced Sequence Diagram:

Process Flow & Interactions:

- Initial Request Phase:
    - Customer initiates Request for Item()
    - System prompts for Enter the Details()
    - Stock check performed via Check if it's there in stock()
- Inventory Management Phase:
    - Warehouse handles Ask for stocks()
    - Logistics arranged through Send to truck()
    - Database updated via Update Database()
- Delivery Execution Phase:
    - Quantity confirmation through Ask Quantity()
    - Delivery details finalized via Enter Details and Request Delivery()
    - Process completes with Deliver the Stocks()

Simple Activity Diagram:



Simple Activity:

This activity diagram shows the stock ordering process. Users login, search for products, and place orders. The system checks stock availability, processes payments, and handles deliveries. It also manages product returns and updates inventory quantities with suppliers.

Key Features:

- Multi-layer validation (authentication, stock availability)
- Supplier integration for inventory management
- Customer notification system
- Return process integration
- Real-time stock quantity updates

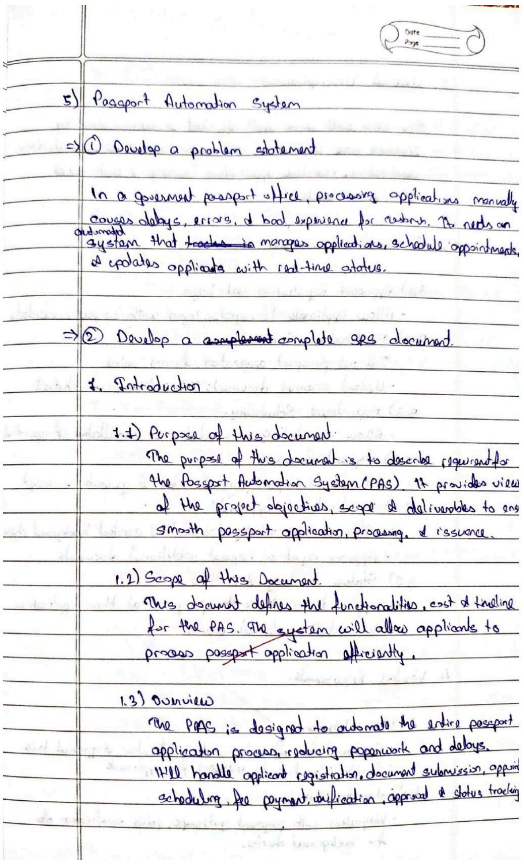
Advanced Activity:

Workflow Stages & Decision Points:

- Authentication Phase:
    - Login → Validation Check → [success]/[failure]
    - Failed authentication leads to "Invalid Data" termination
- Product Search & Ordering:
    - Search for product → [availability]/[unavailability]
    - Successful search enables order placement
    - Customer notification upon order confirmation
- Inventory & Fulfillment:
    - Stock check → [sufficient stock]/[insufficient stock]
    - Supplier coordination for order updates
    - Delivery execution to customer
- Payment & Returns:
    - Payment processing after successful delivery
    - Return handling for unsatisfactory products
    - Real-time inventory updates from suppliers

5. Passport Automation System

SRS Document:



5) Passport Automation System

=> (1) Develop a problem statement

In a government passport office, processing applications manually causes delays, errors, & bad experience for customers. We needs an automated system that ~~tracks~~ to manages applications, schedule appointments, & updates applicants with real-time status.

=> (2) Develop a ~~complement~~ complete SRS document.

1. Introduction

1.1) Purpose of this document

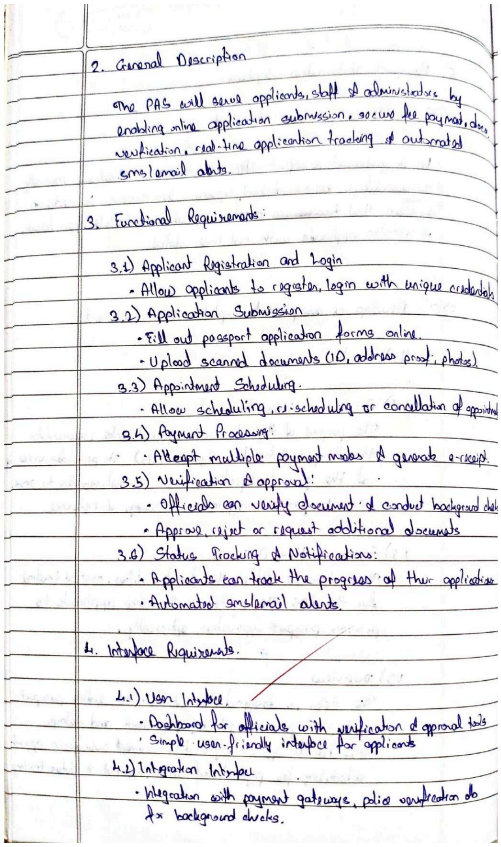
The purpose of this document is to describe requirements for the Passport Automation System (PAS). It provides view of the project objectives, scope & deliverables to ens smooth passport application, processing, & issuance.

1.2) Scope of this Document

This document defines the functionalities, cost & timeline for the PAS. The system will allow applicants to process passport application efficiently.

1.3) Overview

The PAS, is designed to automate the entire passport application process, reducing paperwork and delays. Will handle applicant registration, document submission, appoin scheduling, fee payment, verification, approval & status tracking



2. General Description

The PAS will serve applicants, staff & administrators by enabling online application submission, secure fee payment, doc verification, real-time application tracking & automated sms/email alerts.

3. Functional Requirements:

3.1) Applicant Registration and Login

- Allow applicants to register, login with unique credentials

3.2) Application Submission

- Fill out passport application forms online.
- Upload scanned documents (ID, address proof, photos)

3.3) Appointment Scheduling.

- Allow scheduling, re-scheduling or cancellation of appointm

3.4) Payment Processing:

- Accept multiple payment modes & generate e-receipt

3.5) Verification & approval:

- Officials can verify document & conduct background chec
- Approve, reject or request additional documents

3.6) Status Tracking & Notifications:

- Applicants can track the progress of their application
- Automated sms/email alerts.

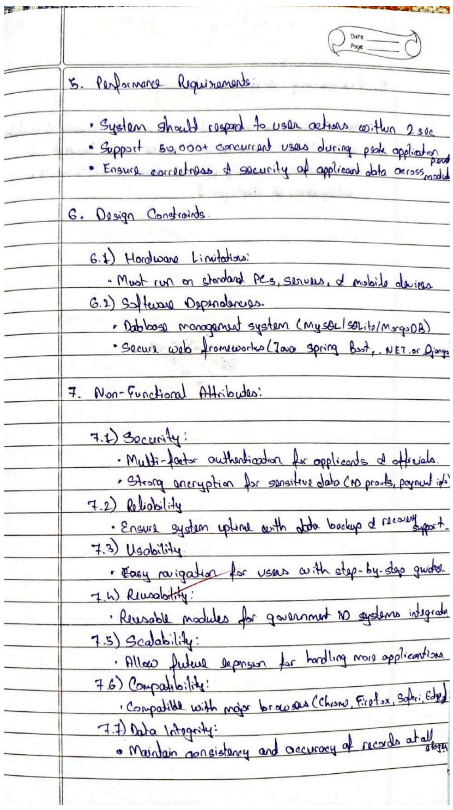
4. Interface Requirements.

4.1) User Interface.

- Dashboard for officials with verification & approval tools
- Simple user-friendly interface for applicants

4.2) Integration Interface

- Integration with payment gateways, police verification db & background checks.



5. Performance Requirements:

- System should respond to user actions within 2 sec
- Support 50,000+ concurrent users during peak application period
- Ensure correctness & security of applicant data across modules

6. Design Constraints

6.1) Hardware Limitations:
- Must run on standard PCs, servers, & mobile devices

6.2) Software Dependencies:
- Database management system (MySQL/SQLite/MongoDB)
- Secure web frameworks (Java Spring Boot, .NET or Django)

7. Non-Functional Attributes:

7.1) Security:
- Multi-factor authentication for applicants & officials.
- Strong encryption for sensitive data (ID proofs, payment info)

7.2) Reliability
- Ensure system uptime with data backup & recovery support.

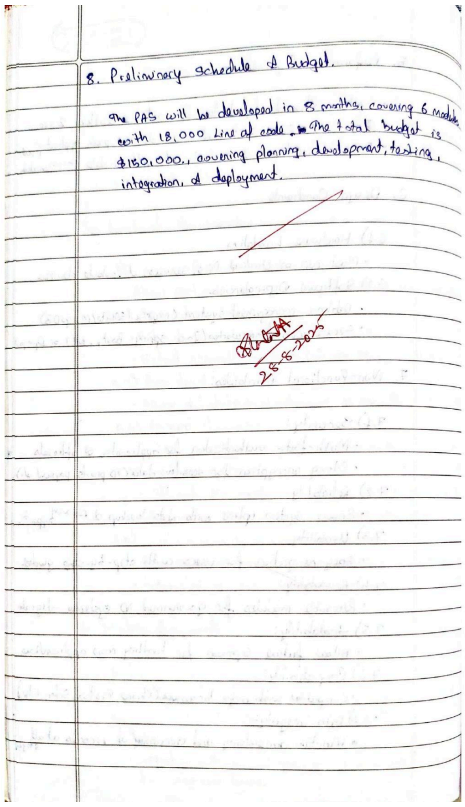
7.3) Usability
- Easy navigation for users with step-by-step guides.

7.4) Reusability:
- Reusable modules for government ID systems integration

7.5) Scalability:
- Allow future expansion for handling more applications

7.6) Compatibility:
- Compatible with major browsers (Chrome, Firefox, Safari, Edge)

7.7) Data Integrity:
- Maintain consistency and accuracy of records at all stages

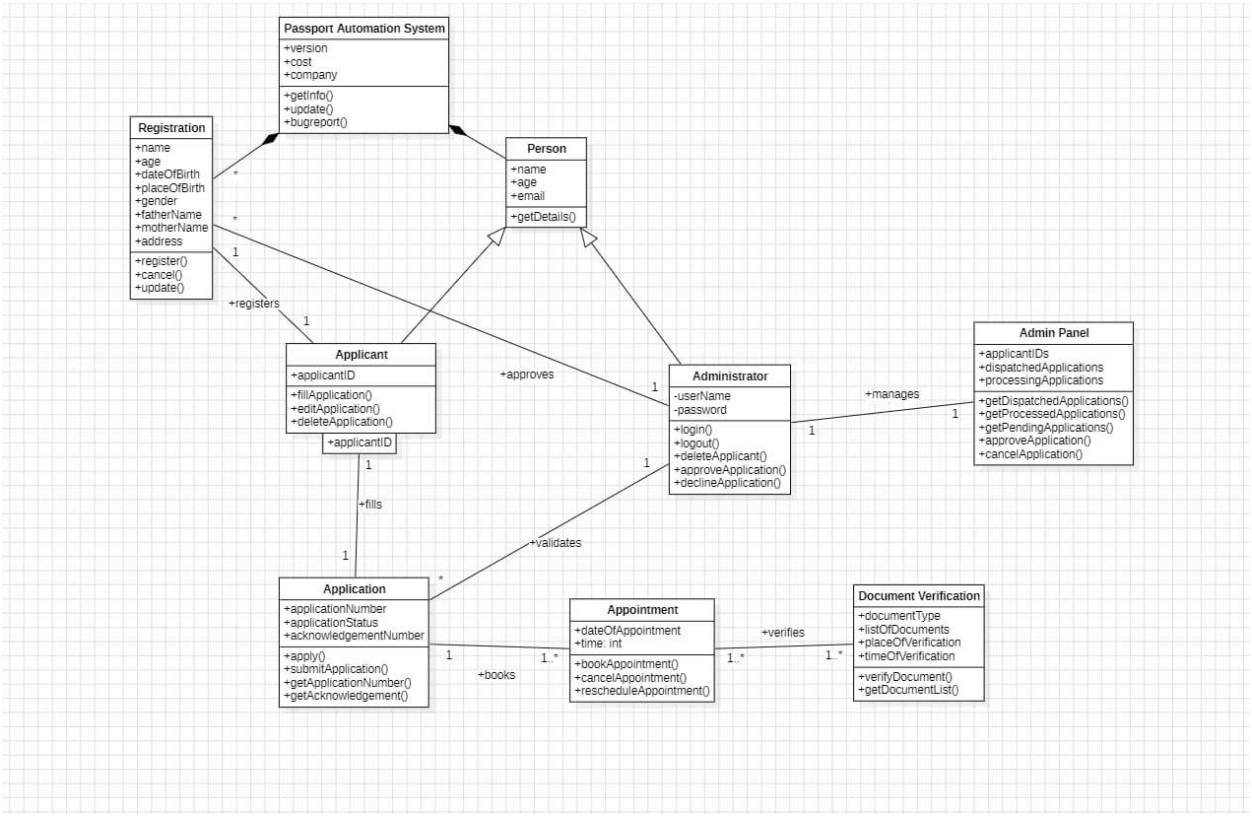


8. Preliminary Schedule & Budget.

The PAS will be developed in 8 months, covering 6 modules with 18,000 line of code. The total budget is ₹150,000., covering planning, development, testing, integration, & deployment.

25-8-2025

Class Diagram



Explanation:

This class diagram models a Passport Automation System with Applicant, Admin, and Application classes. Applicants register personal details, fill applications, and schedule appointments. Admins verify documents, process applications, and handle approvals. The system manages the entire passport issuance workflow.

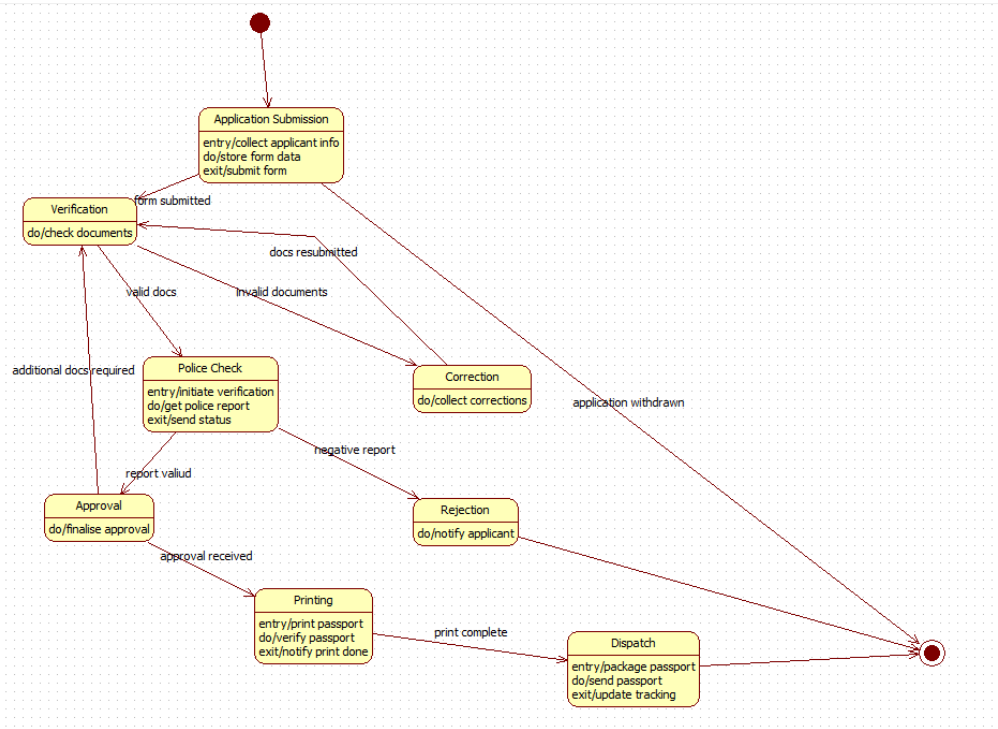
Class Structure & Relationships:

- Core Entities:
    - Person/Applicant: Manages personal data (name, age, address) and application handling
    - Application: Tracks application status, numbers, and submission processes
    - Admin/Issuator: Handles authentication, application approval/rejection, and management
    - Appointment & Verification: Manages scheduling and document validation

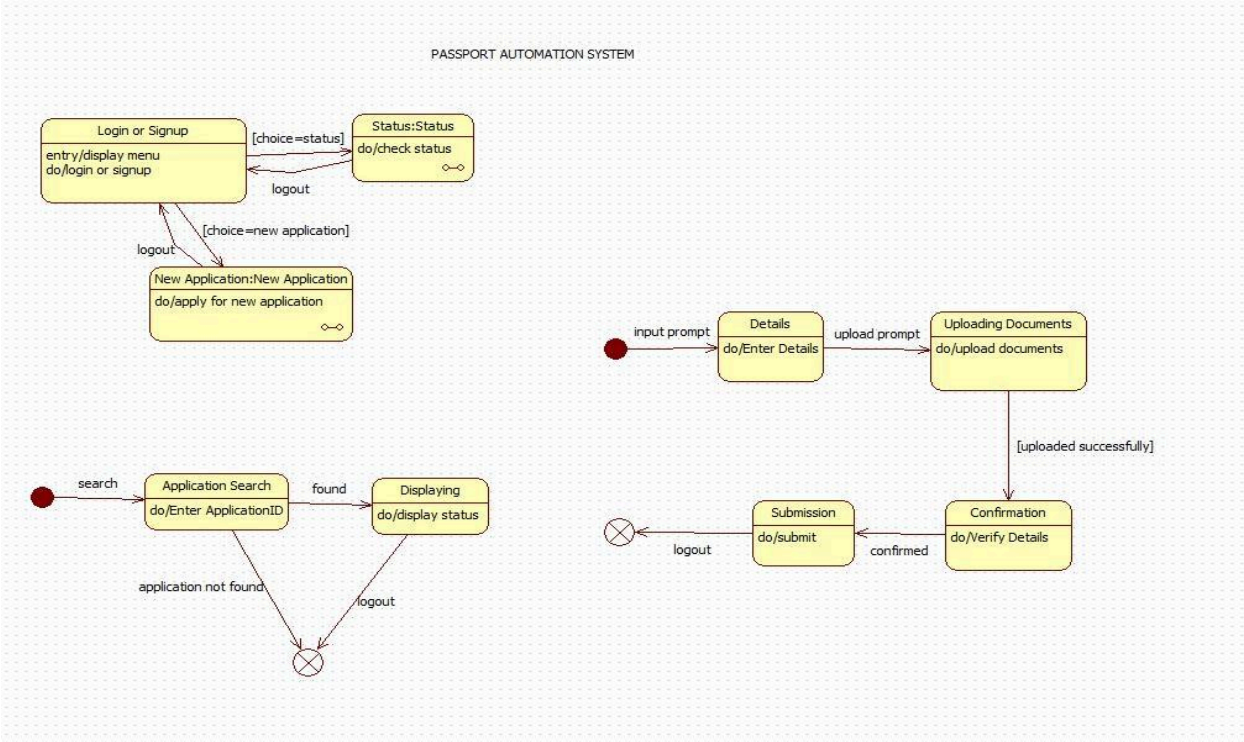
Key Functionalities:

- Application Lifecycle: Registration → Submission → Verification → Approval/Rejection
- Admin Operations: Application processing, status tracking, bulk management
- Document Management: Verification with document type classification
- Appointment Scheduling: Date and time coordination for verification

Simple State Diagram:



Advanced State Diagram:



Simple State:

State Machine Workflow:

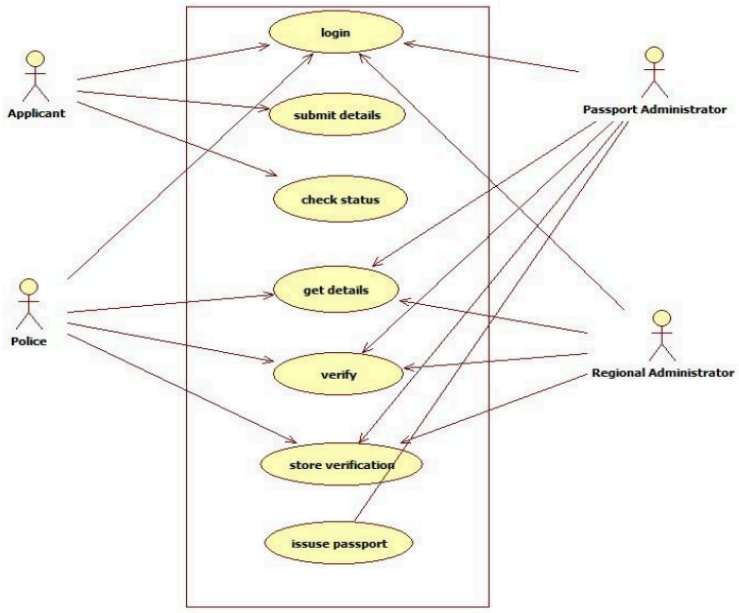
- Submission & Verification Phase:
    - ApplicationSubmission: Collects and stores applicant data
    - Verification: Document validation with transitions for [valid_docs], [invalid_docs], or [additional_docs_requested]
    - Correction: Handles document rectification when needed
- Security & Approval Phase:
    - PoliceCheck: Background verification with [report_clear] or [negative_report] outcomes
    - Approval: Final authorization with 2-day processing time
    - Rejection: Applicant notification for failed applications
- Production & Delivery Phase:
    - Printing: Passport production with quality verification

- Dispatch: Packaging and shipping with tracking updates

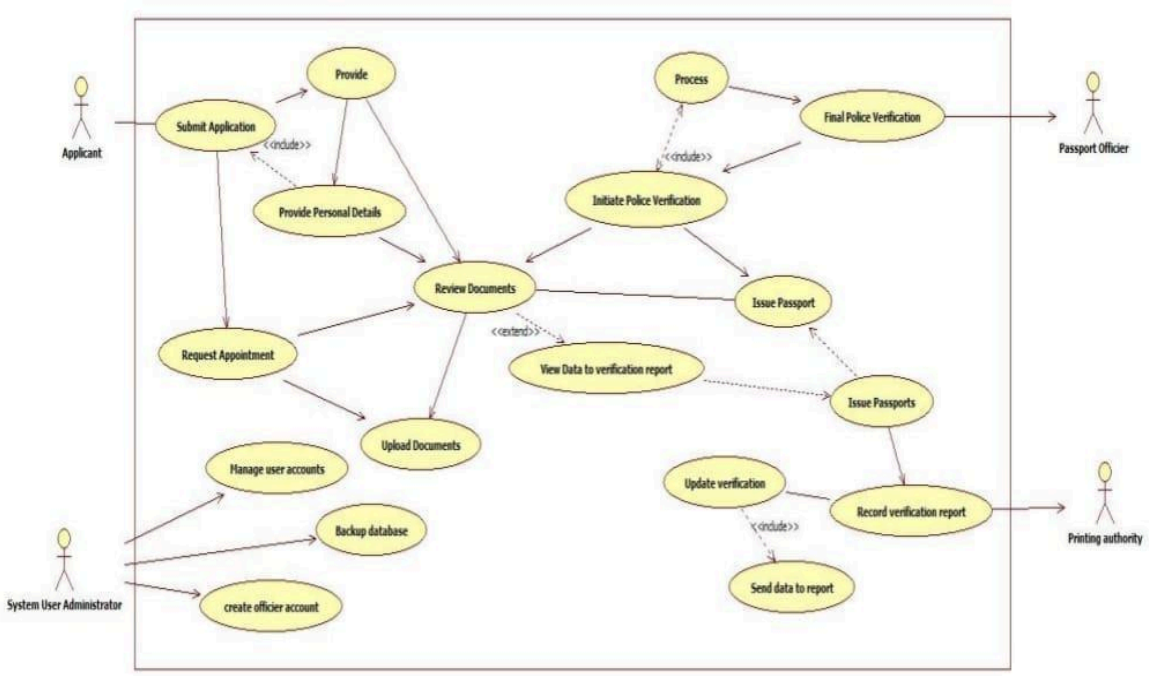
Advanced State:

- Timed transitions (after (2 days))
- Conditional branching based on verification results
- Signal-based state progression (signal(print_complete))
- Exception handling for withdrawn applications
- Quality control checkpoints throughout the process

Simple Use Case:



Advanced Use Case:



Scenario 1: Main Success Scenario (Happy Path)

1. The applicant registers and logs into the passport portal.
2. The applicant fills the application form correctly and uploads all required documents clearly.
3. The applicant successfully completes the online fee payment.
4. A PSK appointment is booked and attended on the scheduled date.
5. The officer verifies original documents and captures biometrics without any issues.
6. Police verification is carried out and the report is submitted as clear/no adverse record.
7. The regional passport authority reviews and approves the application.
8. The passport is printed and dispatched through Speed Post.
9. The applicant tracks the application through the portal and receives the passport within 25–30 days.

Scenario 2: Alternative Scenario (Rejection due to Adverse Police Report)

1. The applicant completes registration, fills the application form, and pays the required fees.
2. The applicant attends the PSK appointment where documents and biometrics are successfully verified.
3. A police officer visits the applicant's address for background verification.
4. During verification, the system detects an existing criminal case or FIR against the applicant.
5. The police submit an adverse verification report.
6. The regional authority reviews the case and rejects the application.
7. An SMS and email notification is sent to the applicant with the reason for rejection.
8. The status on the portal is updated as "Rejected."

Simple Use Case:

This use case diagram shows a passport automation system with Applicant, Administrator, and System actors. Applicants submit applications, provide documents, and request appointments. Administrators process applications, conduct police verification, and issue passports. The system manages data and generates reports.

Actors & Core Functions:

- Applicant:
    - Submit Application (includes providing personal details)
    - Upload Documents
    - Request Appointment
    - Review Application Status
- Administrator:
    - Process Applications
    - Initiate Police Verification

    - Issue Passport
    - Update Application Status
    - Verify Documents
- System:
    - Generate Verification Reports
    - Manage Data
    - Track Application Progress

Advanced Use Case:

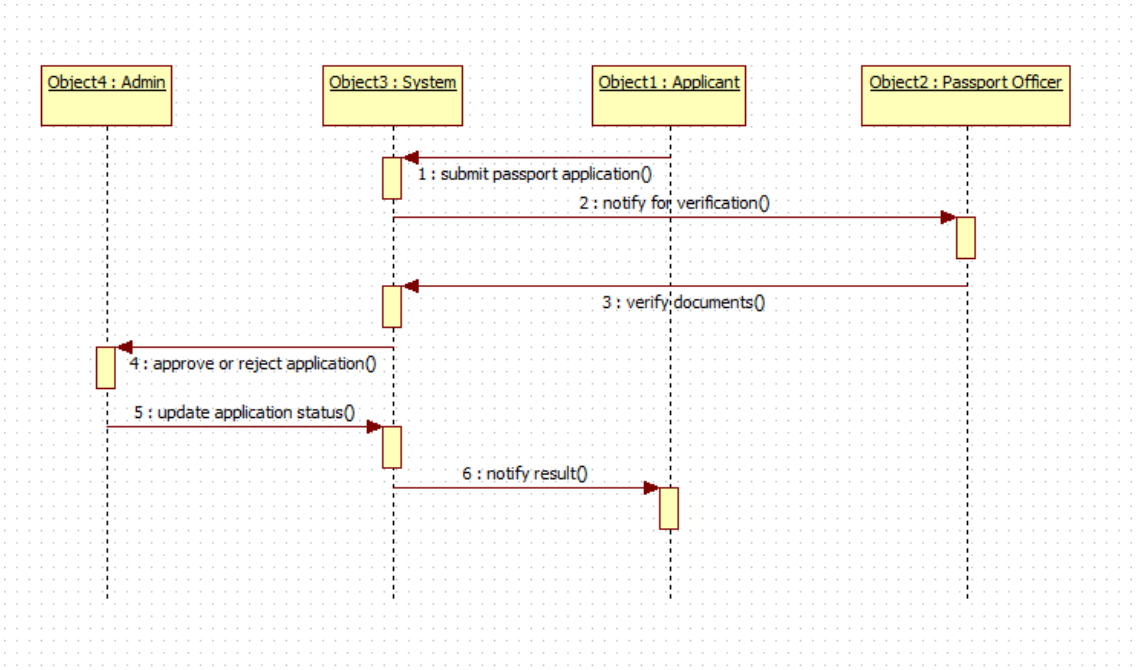
Key Relationships:

- «include»: Mandatory dependencies like document review during application processing
- «extend»: Optional functionalities that enhance core processes

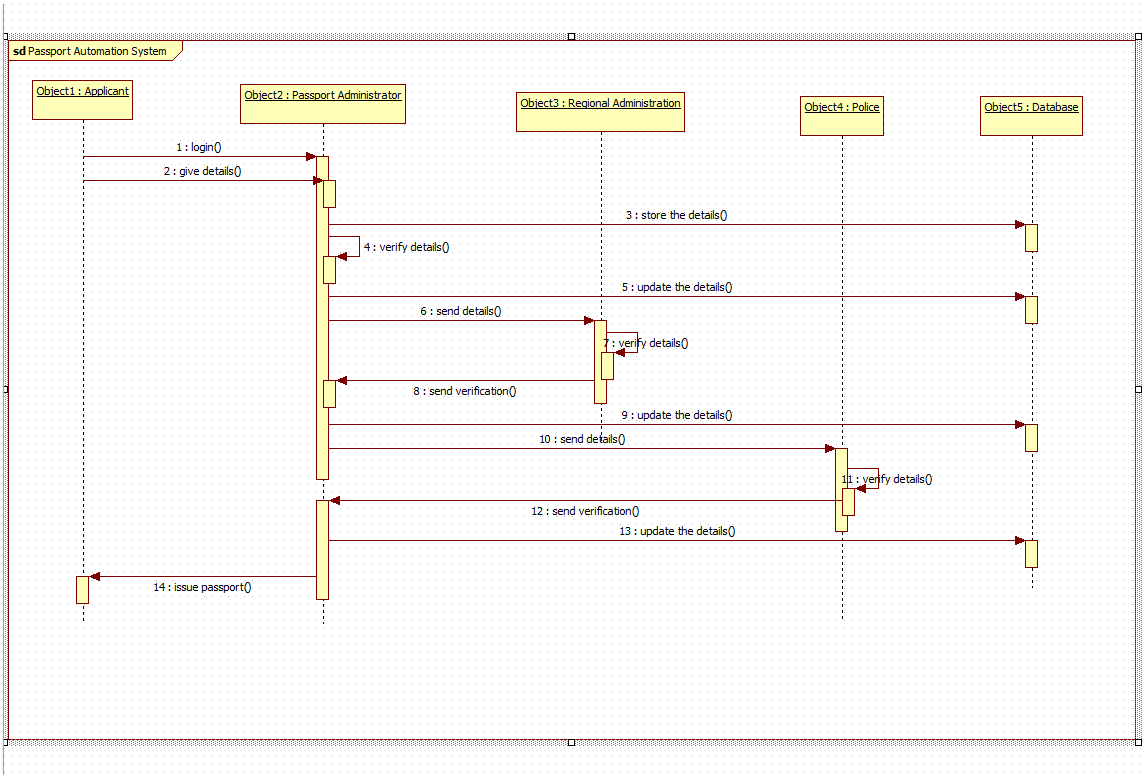
Workflow Features:

- End-to-end application lifecycle management
- Integrated police verification system
- Document validation and status tracking
- Automated reporting and data management
- Multi-level verification processes

Simple Sequence Diagram:



Advanced Sequence Diagram:



Simple Sequence:

This sequence diagram shows the passport automation process. The applicant logs in and provides details. The system stores and verifies data through multiple authorities including regional administration and police. After successful verification, the passport is issued.

Key Technical Features:

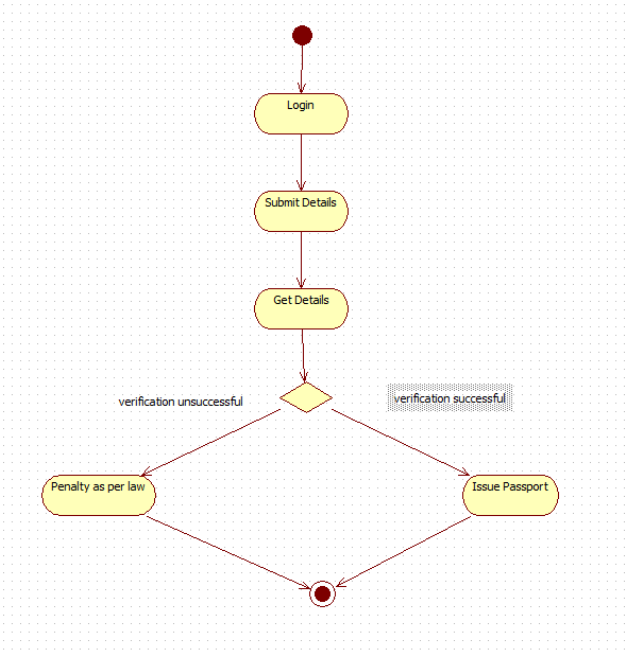
- Sequential dependency between verification stages
- Multiple external authority integrations (Police, Regional Admin)
- Continuous database synchronization
- Secure data flow between government entities
- End-to-end application tracking

Advanced Sequence:

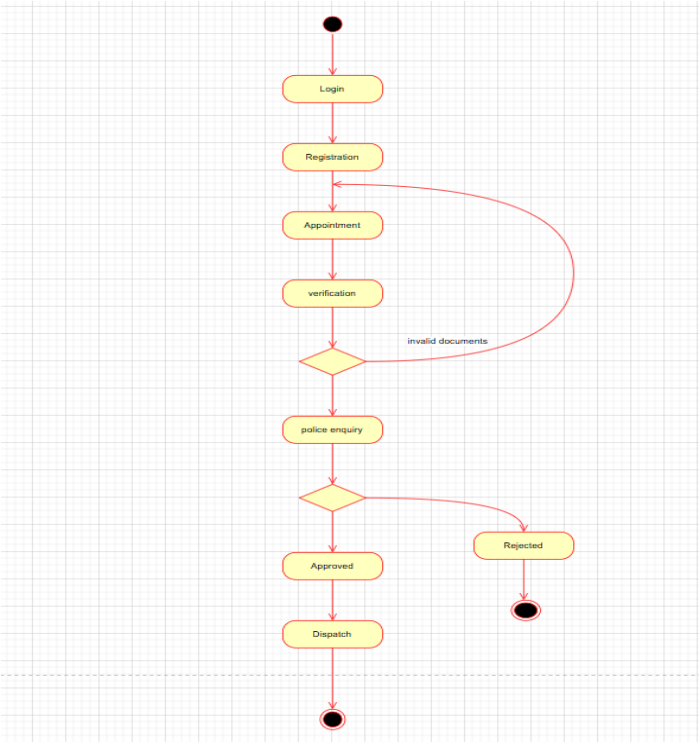
Process Flow & Object Interactions:

- Initialization Phase:
    - Applicant performs login() and give details()
    - Passport Administrator store the details() in Database
- Verification Phase:
    - Multi-level verification chain:
        - Regional Administration verify details() and send verification()
        - Police verify details() and send verification()
    - Real-time database updates after each verification stage
- Completion Phase:
    - Final verification status sent to Passport Administrator
    - Database updated with final approval
    - Passport issued to applicant

Simple Activity Diagram:



Advanced Activity Diagram:



Simple Activity:

Workflow Stages & Control Flow:

- Initial Phase:
    - Login → Registration (user authentication and data collection)
    - Appointment scheduling for document submission
- Verification Phase:
    - Verification of submitted documents
    - Decision node: **[valid documents] → Approval**
    - **[invalid documents] → Police Enquiry → Reported**
- **Final Phase:**
    - **Approved status after successful verification**
    - **Dispatch of passport to applicant**

Advanced Activity:

- Concurrent Verification: Police, background, and financial checks run in parallel
- Swimlanes: Separate lanes for Applicant, Verification, System
- Synchronization: Join point after all verifications complete
- Exception Handling: Timeouts and compensation flows
- Optimization: Priority queues and load balancing
- Persistence: Saga pattern for long-running process

Key Features:

- Linear workflow with conditional branching
- Security integration through police verification
- Document validation checkpoint
- End-to-end application tracking
- Automated status progression