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:

Nishanth Reddy N

1BM23CS214

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 Nishanth Reddy N (1BM23CS214) during the 5th Semester August 2025-December 2025

.

Signature of the Faculty Incharge:

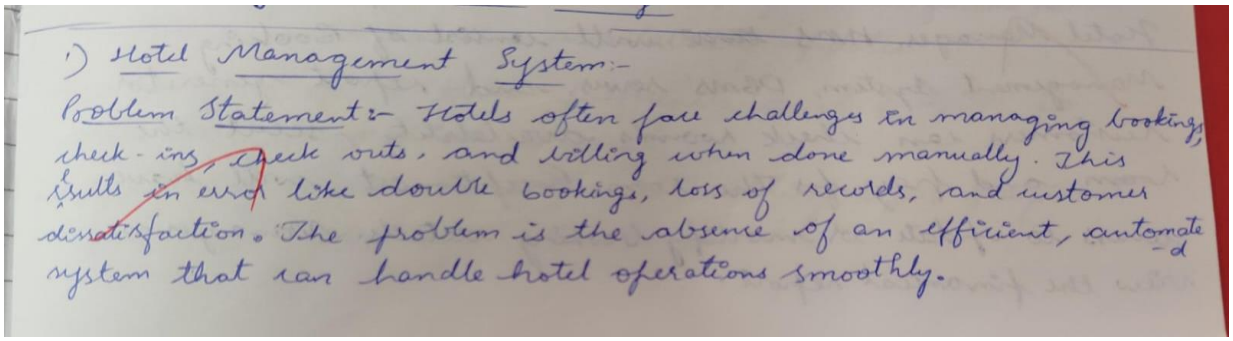
Batch Incharge: Sunayana S
Designation: Assistant Professor

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

Table of Contents:

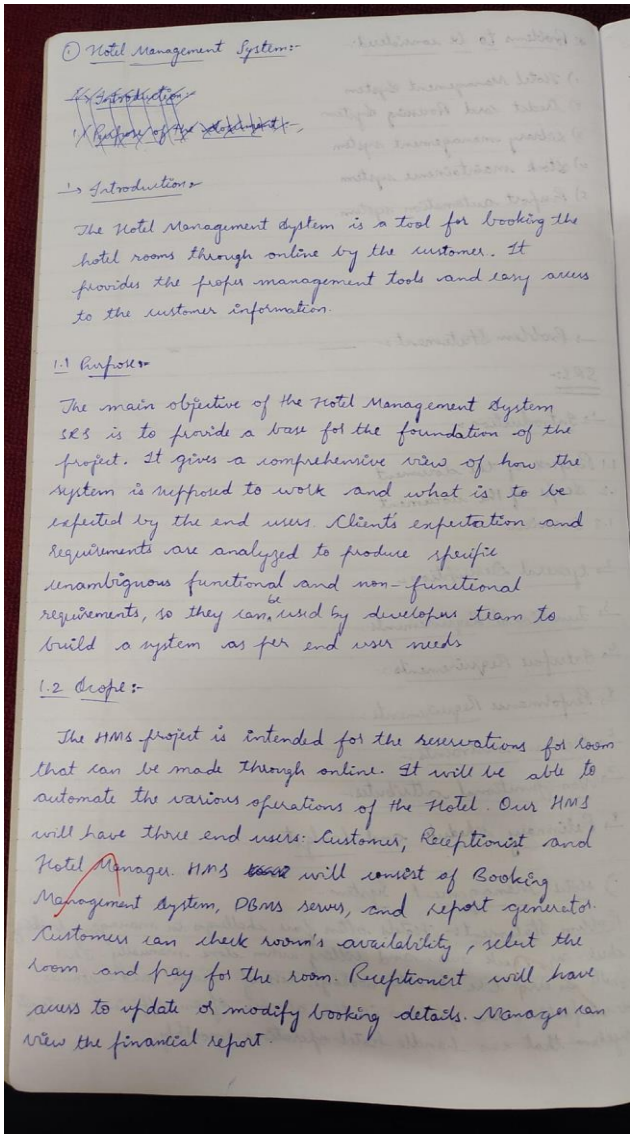
1. Hotel Management System

SRS Document:



1) Hotel Management System:-

Problem Statement:- Hotels often face challenges in managing bookings, check-ins, check outs, and billing when done manually. This results in errors like double bookings, loss of records, and customer dissatisfaction. The problem is the absence of an efficient, automated system that can handle hotel operations smoothly.



1) Hotel Management System:-

~~2) Introduction:-~~

~~1) Purpose of the document:-~~

→ Introduction:-

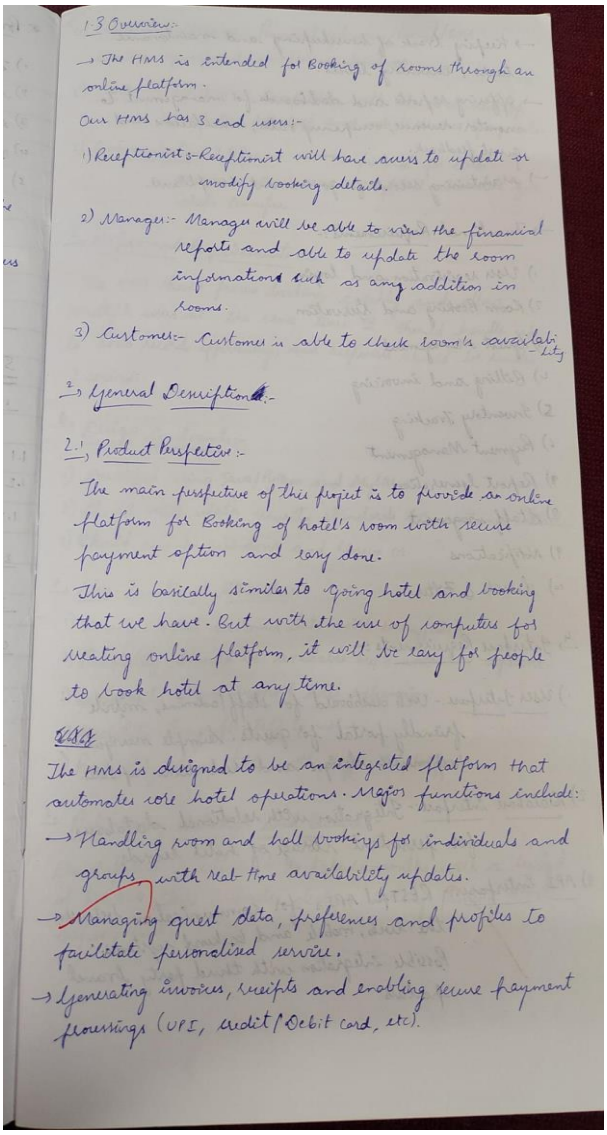
The Hotel Management System is a tool for booking the hotel rooms through online by the customer. It provides the proper management tools and easy access to the customer information.

1.1 Purpose:-

The main objective of the Hotel Management System SRS is to provide a base for the foundation of the project. It gives a comprehensive view of how the system is supposed to work and what is to be expected by the end users. Client's expectation and requirements are analyzed to produce specific unambiguous functional and non-functional requirements, so they can be used by developers team to build a system as per end user needs.

1.2 Scope:-

The HMS project is intended for the reservations for room that can be made through online. It will be able to automate the various operations of the Hotel. Our HMS will have three end users: Customer, Receptionist and Hotel Manager. HMS will consist of Booking Management System, DBMS server, and report generator. Customers can check room's availability, select the room and pay for the room. Receptionist will have access to update or modify booking details. Manager can view the financial report.



1.3 Overview:-

→ The HMS is intended for Booking of rooms through an online platform.

Our HMS has 3 end users:-

1) Receptionist:- Receptionist will have access to update or modify booking details.

2) Manager:- Manager will be able to view the financial reports and able to update the room informations such as any addition in rooms.

3) Customer:- Customer is able to check room's availability

→ General Description:-

2.1) Product Perspective:-

The main perspective of this project is to provide an online platform for Booking of hotel's room with secure payment option and easy done.

This is basically similar to going hotel and booking that we have. But with the use of computers for creating online platform, it will be easy for people to book hotel at any time.

~~2.2~~

The HMS is designed to be an integrated platform that automates core hotel operations. Major functions include:

→ Handling room and hall bookings for individuals and groups with real-time availability updates.

→ Managing guest data, preferences and profiles to facilitate personalised service.

→ Generating invoices, receipts and enabling secure payment processings (UPI, credit/Debit card, etc).

→ Keeping track of housekeeping and maintenance tasks, inventory levels.
→ Offering reports and dashboards for management to monitor revenue, occupancy rates, cancellations and guest feedback.
→ Maintaining records for government compliance.

3. Functional Requirements:-

1) User registration and login
2) Room Booking and Reservation
3) Guest Management
4) Billing and invoicing
5) Inventory Tracking
6) Payment Management
7) Report Generation
8) Staff management
9) Notifications
10) Search & Filter.

4. Interface Requirements:-

1) User Interface:- Web dashboard for staff/admins, mobile friendly portal for guests. Simple navigation, responsive design and real-time updates.
2) Database interface:- Integration with relational databases for persistent storage of hotel records.
3) API interface:- RESTful APIs for communication between the web, mobile and backend systems. Possible integration with third party travel portals.

4) External interfaces:- Payment gateway, email/sms services, government compliance databases.
5) Hardware interfaces:- Compatible with standard desktop computers for staff and mobile devices for guests.
6) Communication interfaces:- HTTPS protocol for web communication, secure encrypted channels for data transfer.

5. Performance Requirements:-

The HMS should process bookings quickly and allow multiple users at the same time. It should handle up to 5000 records efficiently, and responses should be under 2 seconds.

6. Design Constraints:-

1) Developed using Java/Python and MySQL.
2) Should meet data security standards for customer details.
3) Should run on Windows and Linux OS.

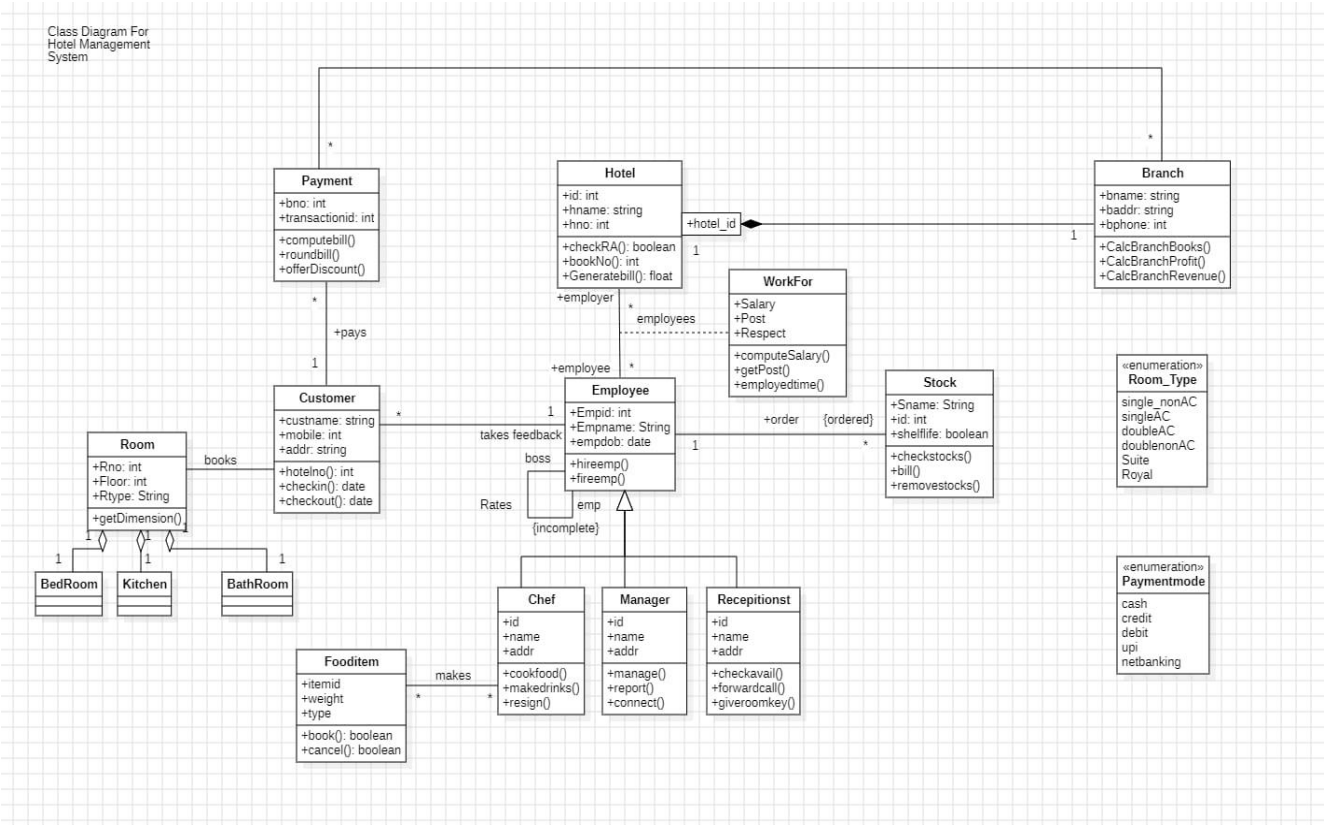
7. Non-Functional Attributes:-

1) Security: Protect customer data with encryption.
2) Reliability: Should be available 24/7 with 99% uptime.
3) Scalability: Must support hotel expansion.
4) Usability: Easy for staff to use.
5) Portability: Compatible with different platforms.

8. Preliminary Schedule and Budget:-

Development is estimated to take 4 months with a budget of $20,000. This includes design, coding, testing and deployment.

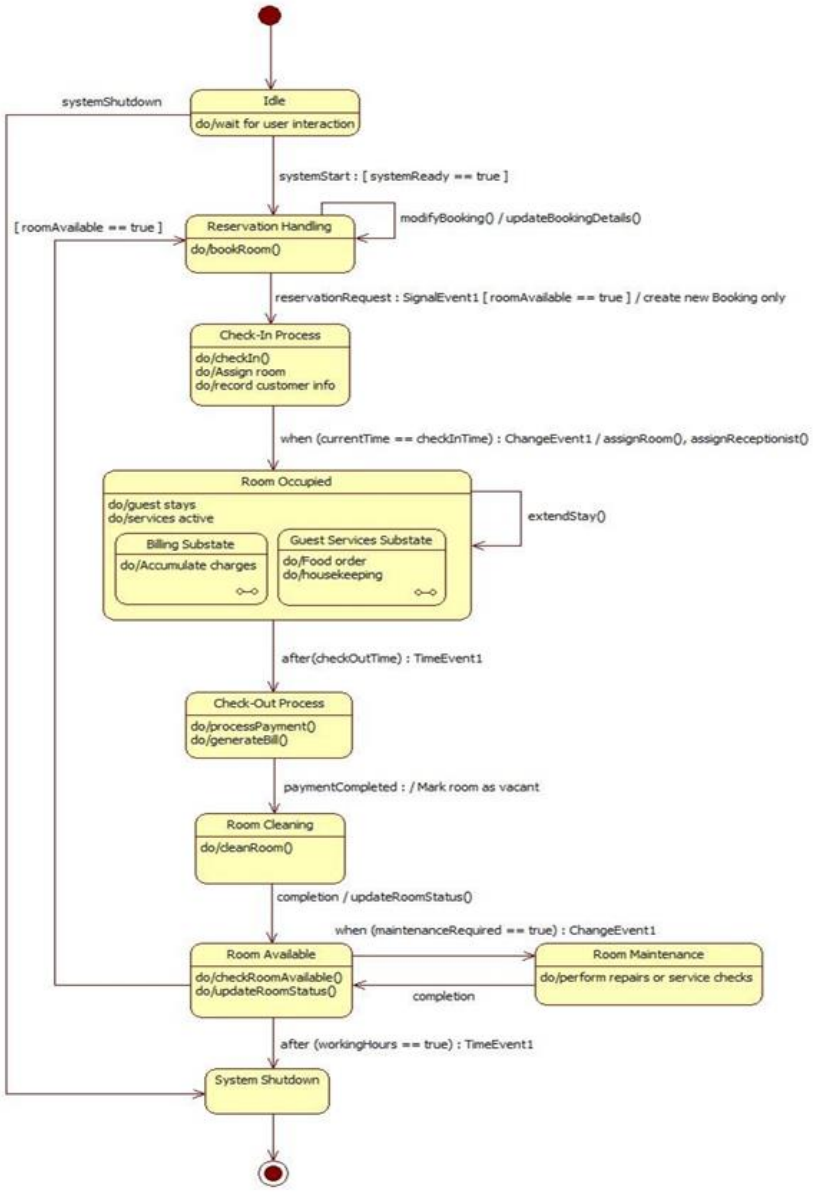
Class Diagram:



Description:

The class diagram for the Hotel Management System outlines the core structure using main classes such as **Hotel, Branch, Customer, Room, Employee, Payment, FoodItem, and Stock**. The Hotel manages its employees and interacts with customers, while branches store location-specific details. Customers book different types of rooms and complete their transactions through the Payment class, which supports various payment modes. Rooms are categorized using room-type enumeration, and components like Bedroom, Kitchen, and Bathroom help define room structure. Employees are organized into roles such as Chef, Manager, and Receptionist through inheritance, supporting operations like cooking, managing, and handling check-ins. FoodItem and Stock classes help track orders and inventory. Overall, the diagram captures key relationships, inheritance, and classifications needed to manage hotel operations smoothly.

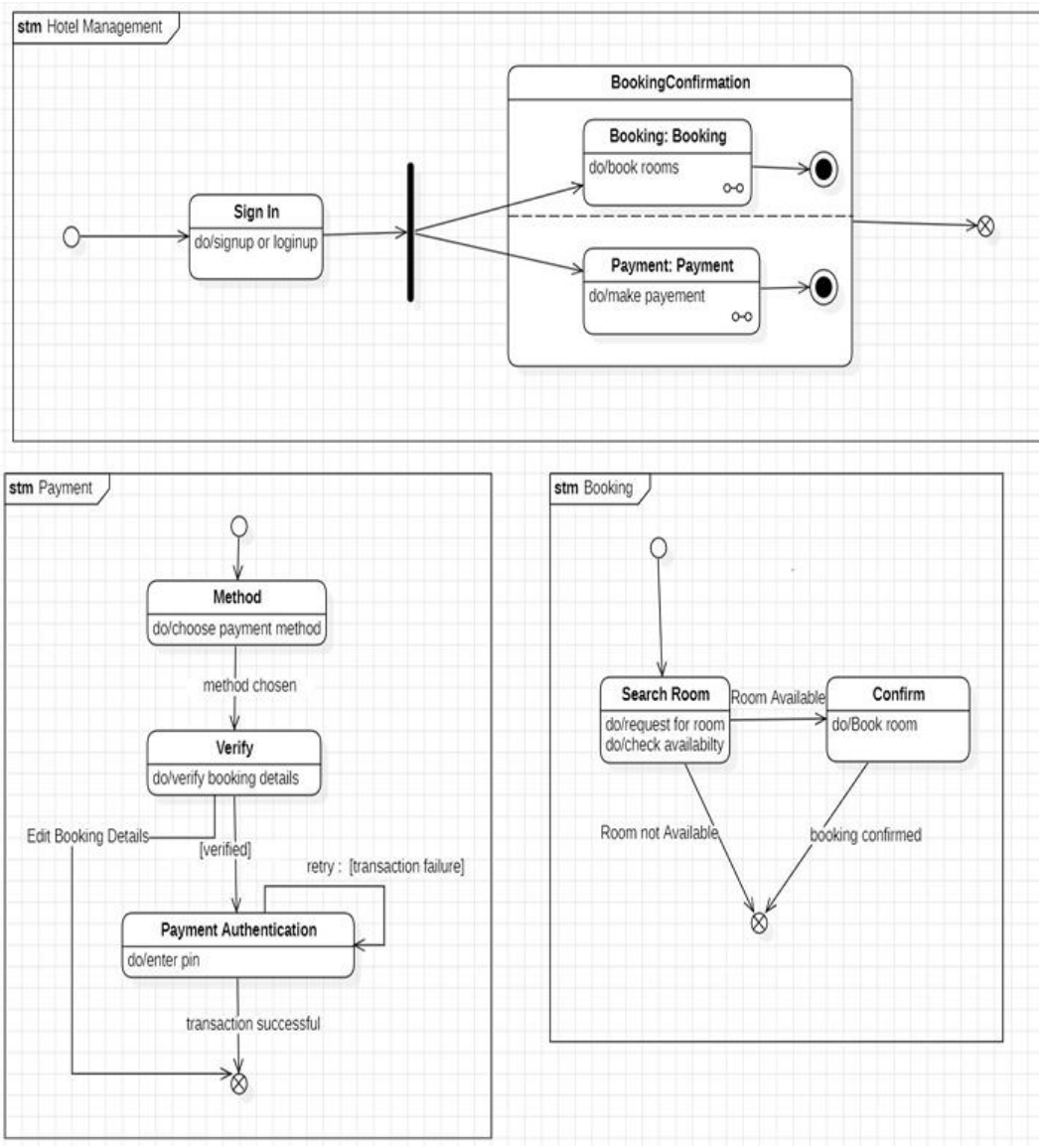
State Diagram (Simple):



Description:

The simple state diagram shows only the basic steps a user follows while booking a room and making a payment. The process starts when the user logs into the system. After logging in, the user searches for available rooms, selects one, and confirms the booking. Once the booking is done, the user chooses a payment method and completes the transaction. This diagram focuses only on the main actions—searching rooms, checking availability, booking, and paying—making it easy to understand the basic workflow.

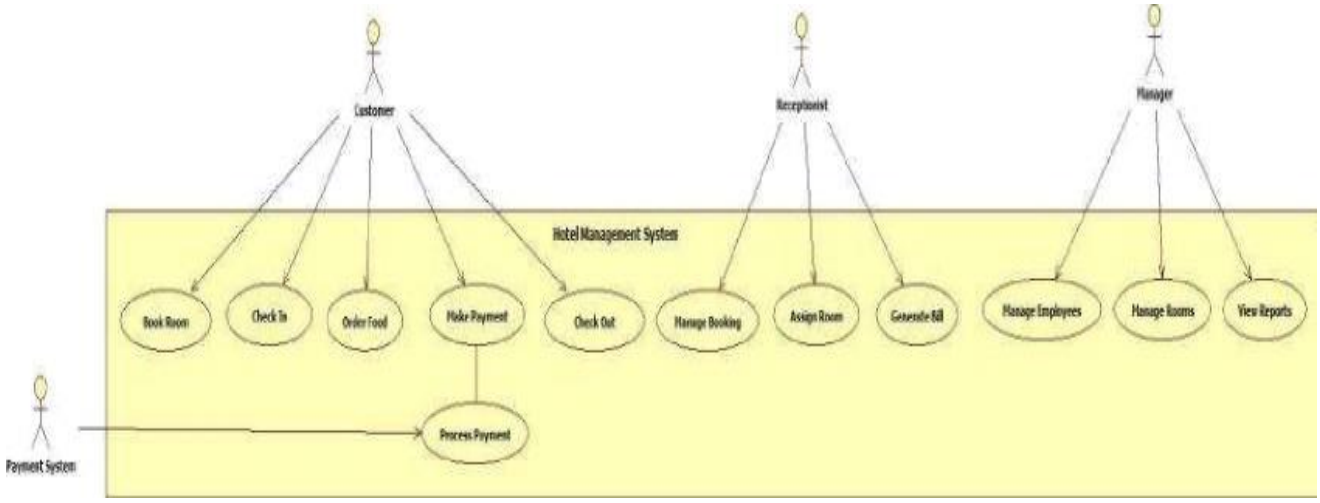
State Diagram (Advanced):



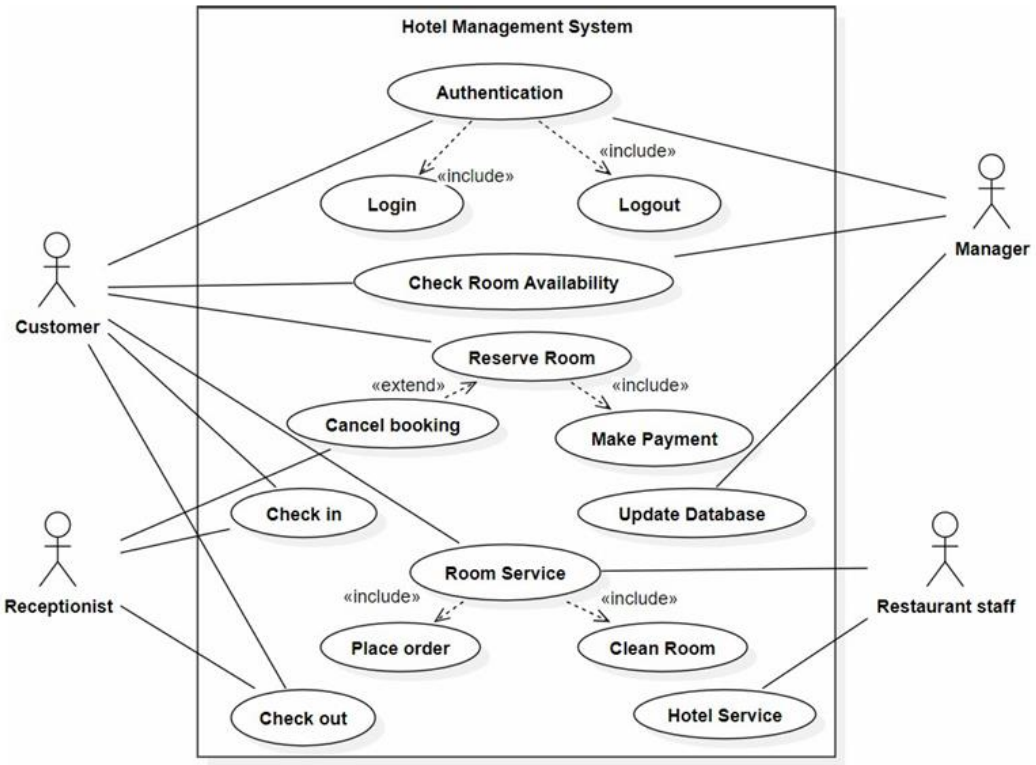
Description:

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

Use Case Diagram(Simple):



Use Case Diagram(Advanced):



Description:

Use Case Diagram(Simple):

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

Use Case Diagram(Advanced):

The use case diagram illustrates how different users interact with the Hotel Management System. The **Customer** can log in, check room availability, reserve rooms, make payments, cancel bookings, check in, request room services, and check out. **Authentication** includes both login and logout processes. When a customer reserves a room, the system may also extend to include cancel booking and must include making payment.

The **Receptionist** handles check-ins, check-outs, and assists with room service tasks. **Restaurant staff** support room service operations such as placing food orders and cleaning rooms. The **Manager** oversees the system by checking room availability and managing updates to the hotel database.

The diagram uses **include** and **extend** relationships to show dependencies between use cases, such as payment being included in reservation and room service including food ordering and cleaning. Overall, the diagram provides a detailed view of how various actors interact with core hotel functions.

Scenarios

1. Book Room

The customer opens the hotel booking page.

The customer selects the desired room type (Single, Double, AC, Non-AC, etc.).

The system filters and displays all available rooms of that type.

The customer chooses one room from the list.

The customer enters personal details (name, contact number, ID proof). The system verifies the entered information.

The customer confirms the booking.

The system generates a booking confirmation and stores reservation details.

2. Process Payment

The customer selects the "Make Payment" option. The system displays the total bill amount.

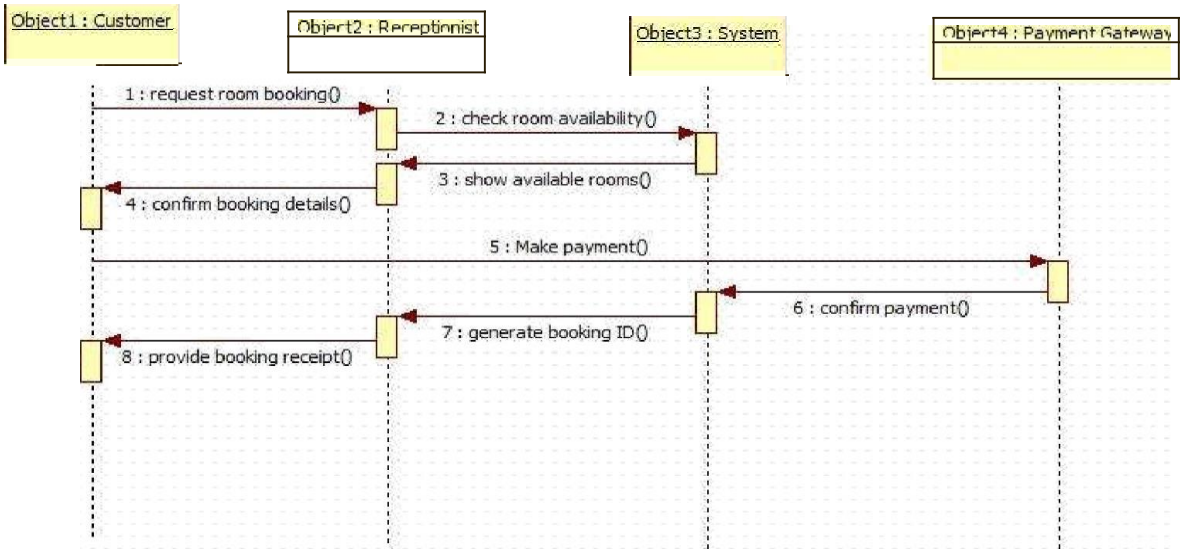
The customer chooses a payment method (UPI, Card, Net Banking, etc.). The system redirects the request to the external payment gateway.

The customer enters required payment details.

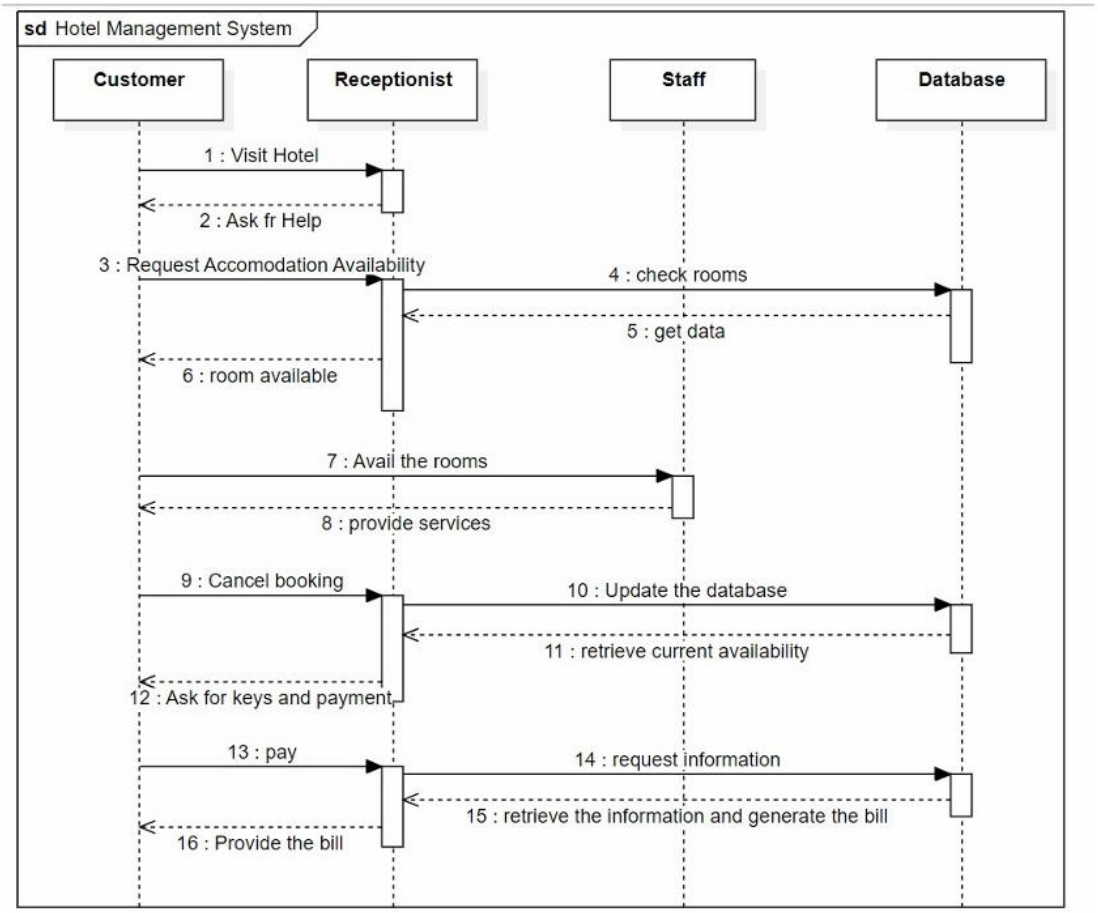
The payment gateway verifies the details and processes the transaction.

On success, the gateway sends a payment success message to the hotel system. The system marks the booking as "Paid" and generates a payment receipt.

Sequence Diagram(Simple):



Sequence Diagram(Advanced):



Description:

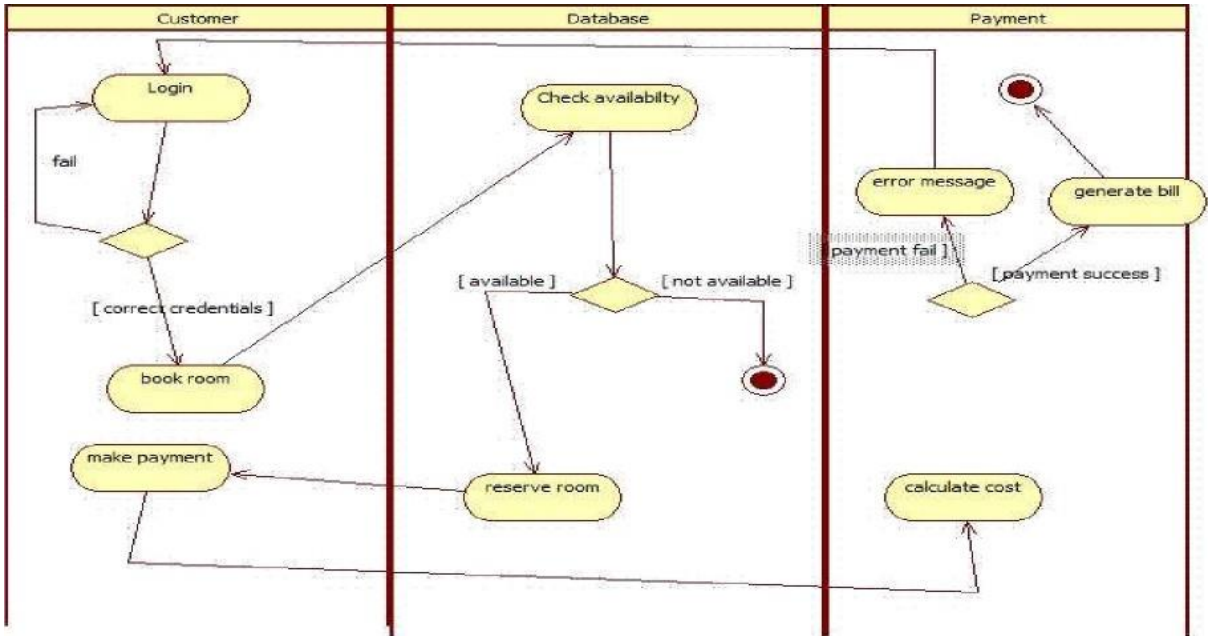
Sequence Diagram(Simple):

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

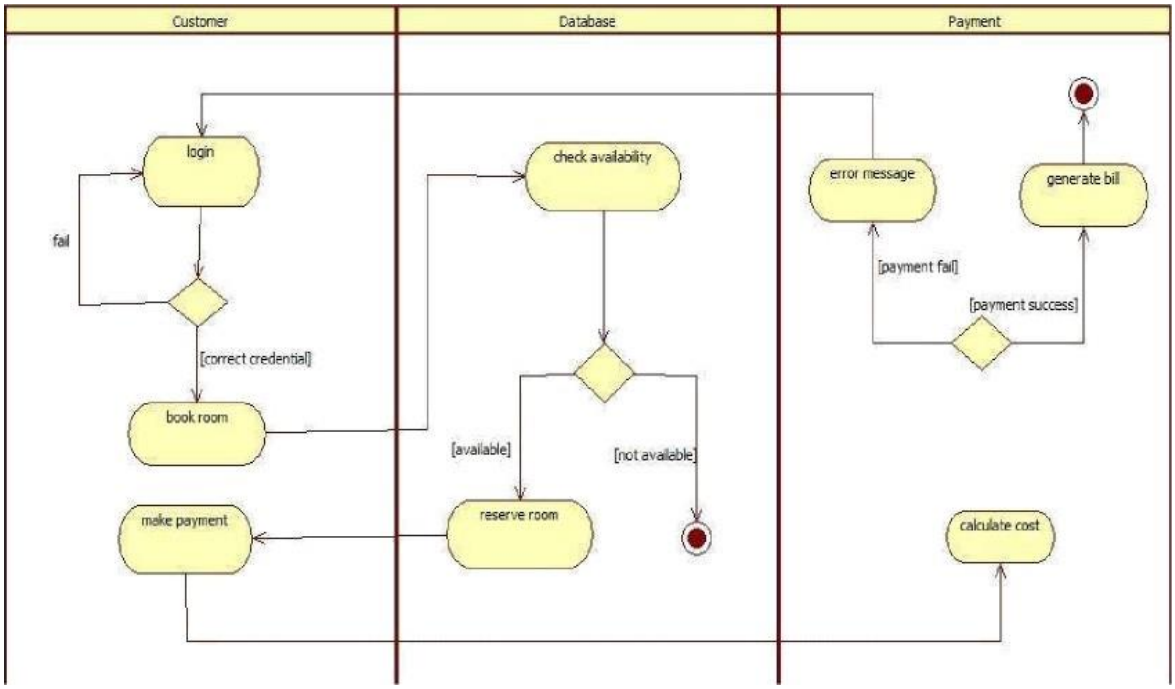
Sequence Diagram(Advanced):

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

Activity Diagram(Simple):



Activity Diagram(Advanced):



Description:

Activity Diagram(Simple):

The simple activity diagram shows the basic steps a customer follows to book a room and make a payment. It includes three main actions:
•t• Login: The customer enters their details. If the login fails, they try again; if it succeeds, they move forward.
•t• Book Room: The system checks room availability. If no rooms are available, the process ends. If a room is available, it is reserved.
•t• Make Payment: The customer pays for the booking. If the payment fails, an error message appears. Ifit succeeds, the system generates the final bill.

This simple diagram focuses only on the essential steps—login, booking, and payment—making it easy for beginners to understand the basic flow.

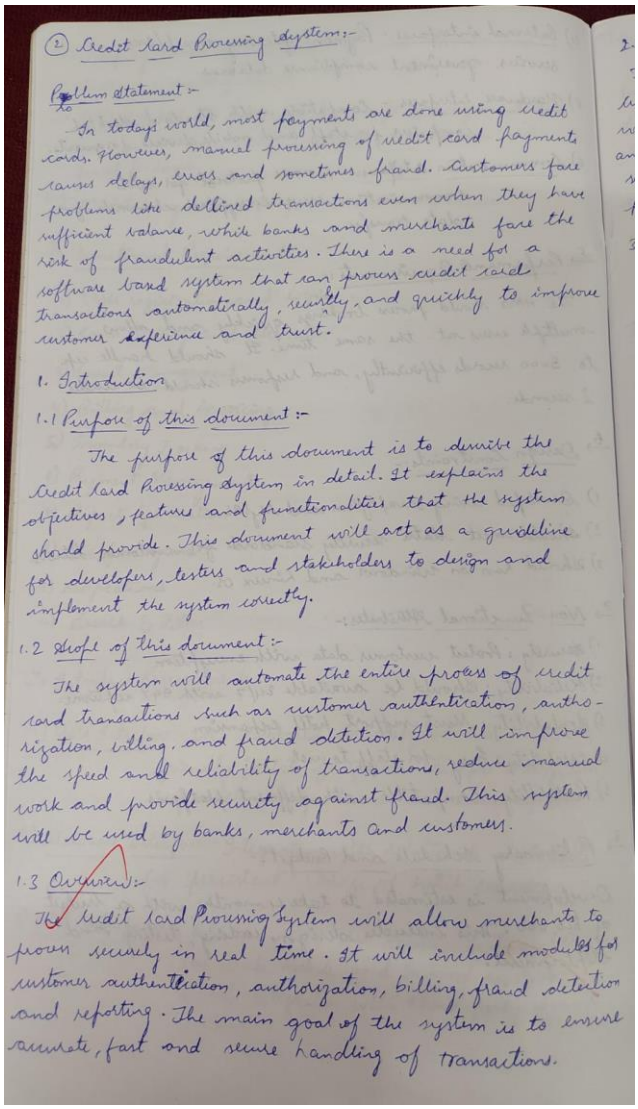
Activity Diagram(Advanced):

The advanced activity diagram shows the same process in more detail and separates the actions into three swimlanes: Customer, Database, and Payment.
•t• In the Customer section, the user logs in, books a room, and makes the payment.
t• In the Database section, the system checks if rooms are available and reserves the room when possible.
•t• In the Payment section, the system calculates the total amount, processes the payment, and generates the bill.

The advanced diagram also includes decision points such as incorrect login, room not available, and payment failure. It presents a clearer and more complete picture of how different parts of the hotel system work together.

2. Credit Card Processing

SRS Document:



② Credit Card Processing System:-

Problem statement:-

In today's world, most payments are done using credit cards. However, manual processing of credit card payments causes delays, errors and sometimes fraud. Customers face problems like declined transactions even when they have sufficient balance, while banks and merchants face the risk of fraudulent activities. There is a need for a software based system that can process credit card transactions automatically, securely, and quickly to improve customer experience and trust.

1. Introduction

1.1 Purpose of this document:-

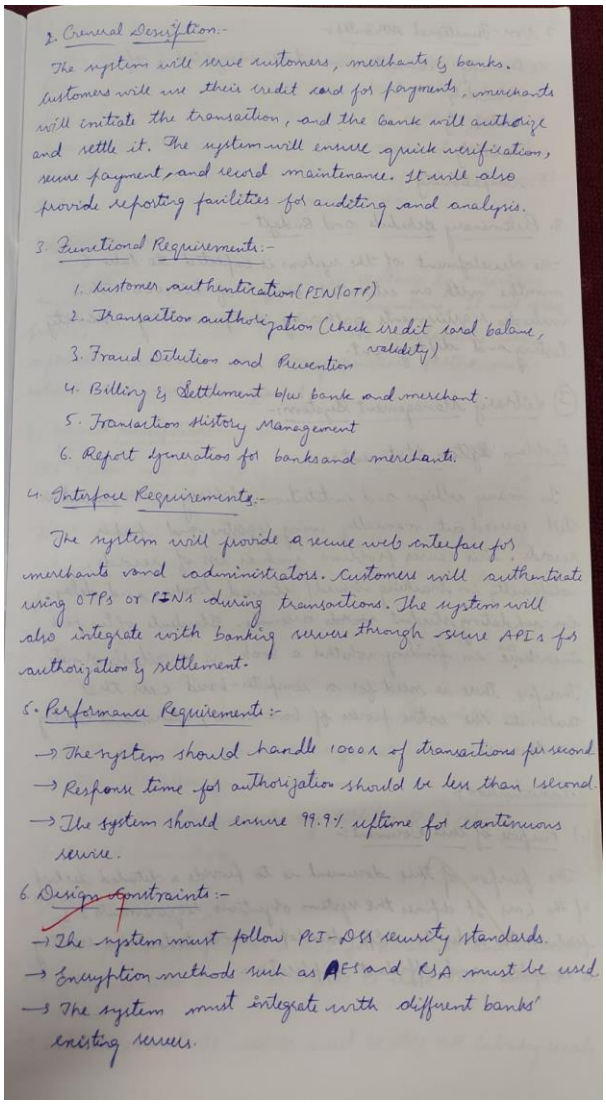
The purpose of this document is to describe the Credit Card Processing System in detail. It explains the objectives, features and functionalities that the system should provide. This document will act as a guideline for developers, testers and stakeholders to design and implement the system correctly.

1.2 Scope of this document:-

The system will automate the entire process of credit card transactions such as customer authentication, authorization, billing, and fraud detection. It will improve the speed and reliability of transactions, reduce manual work and provide security against fraud. This system will be used by banks, merchants and customers.

1.3 Overview:-

The Credit Card Processing System will allow merchants to process securely in real time. It will include modules for customer authentication, authorization, billing, fraud detection and reporting. The main goal of the system is to ensure accurate, fast and secure handling of transactions.



2. General Description:-

The system will serve customers, merchants & banks. Customers will use their credit card for payments, merchants will initiate the transaction, and the bank will authorize and settle it. The system will ensure quick verification, secure payment, and record maintenance. It will also provide reporting facilities for auditing and analysis.

3. Functional Requirements:-

1. Customer authentication (PIN/OTP)
2. Transaction authorization (check credit card balance, validity)
3. Fraud Detection and Prevention
4. Billing & Settlement b/w bank and merchant
5. Transaction History Management
6. Report generation for banks and merchants.

4. Interface Requirements:-

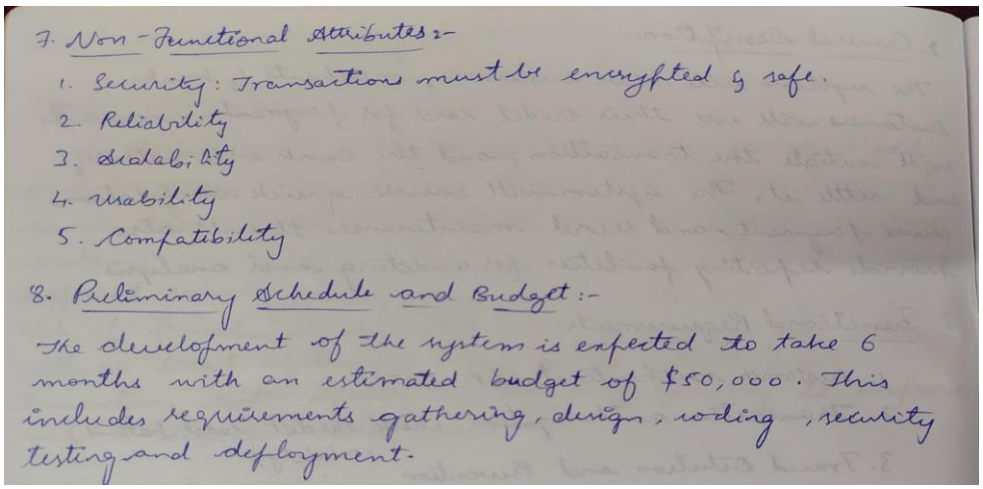
The system will provide a secure web interface for merchants and administrators. Customers will authenticate using OTPs or PINs during transactions. The system will also integrate with banking servers through secure APIs for authorization & settlement.

5. Performance Requirements:-

→ The system should handle 1000s of transactions per second
→ Response time for authorization should be less than 1 second
→ The system should ensure 99.9% uptime for continuous service.

6. Design Constraints:-

→ The system must follow PCI-DSS security standards.
→ Encryption methods such as AES and RSA must be used.
→ The system must integrate with different banks' existing servers.



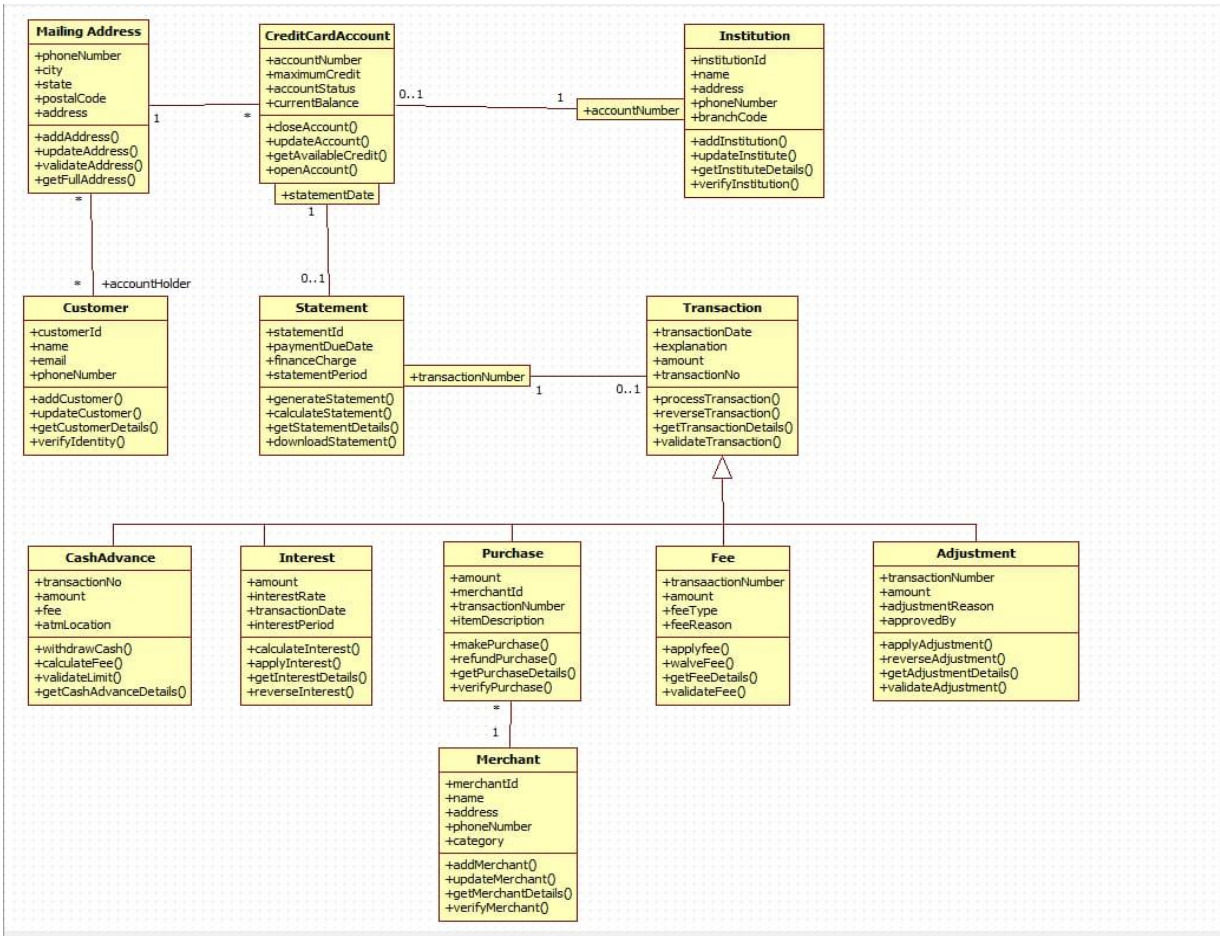
7. Non-Functional Attributes:-

1. Security: Transactions must be encrypted & safe.
2. Reliability
3. Scalability
4. Usability
5. Compatibility

8. Preliminary Schedule and Budget:-

The development of the system is expected to take 6 months with an estimated budget of $50,000. This includes requirements gathering, design, coding, security testing and deployment.

Class Diagram:



Description:

This class diagram represents a **Credit Card Management System**, showing how customer accounts, transactions, statements, and institutions are organized. The **Customer** class stores personal information and is linked to a **Mailing Address** and a **CreditCardAccount**, which manages account details such as credit limits, status, and balance. Each account is associated with an **Institution**, like a bank or financial organization.

The **Statement** class generates monthly statements for the credit card account, calculating charges, finance fees, and providing downloadable records. All financial activities are recorded as **Transactions**, which include date, amount, explanation, and processing functions.

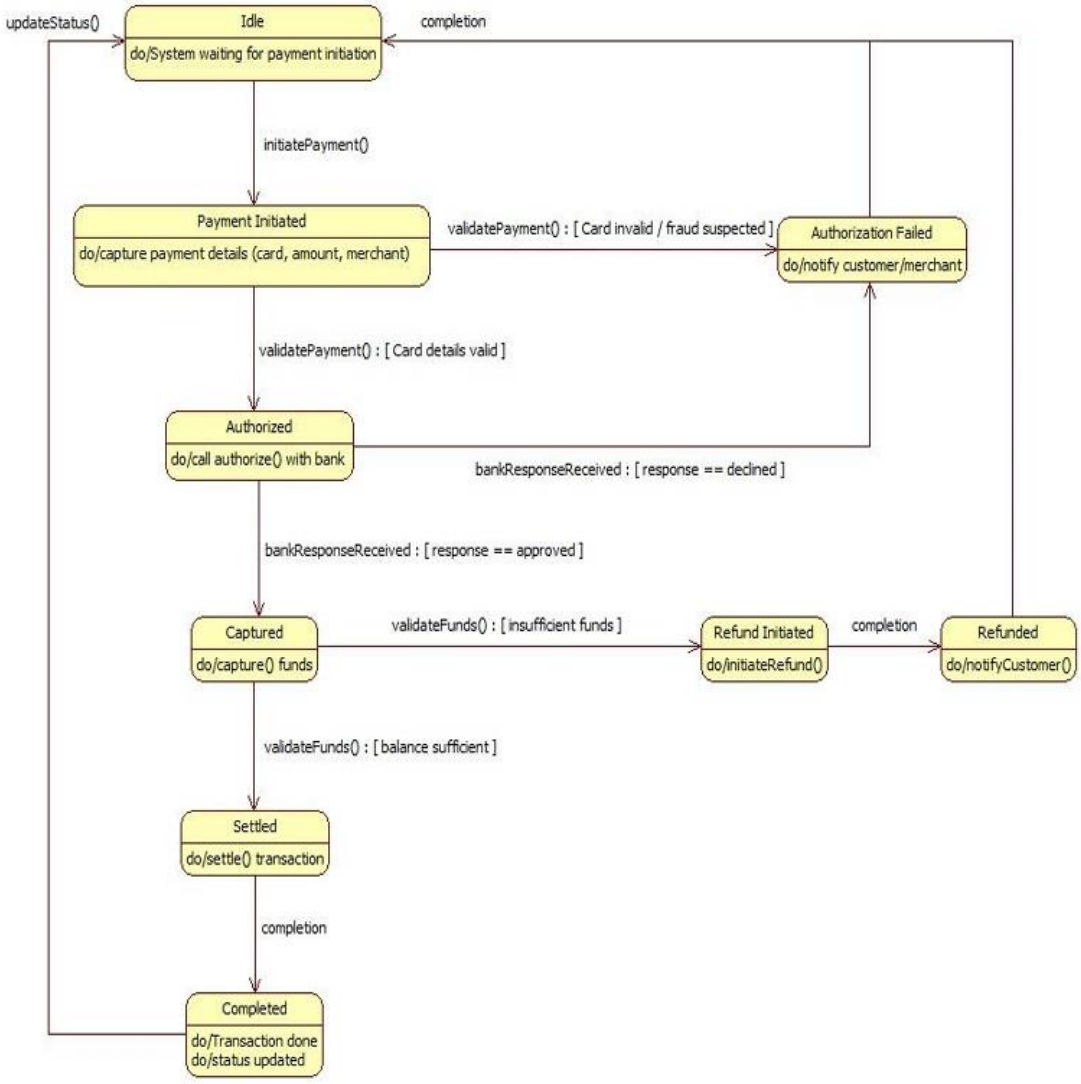
Different types of transactions are specialized through inheritance:

- **CashAdvance** handles cash withdrawals.
- **Interest** stores interest charges.
- **Purchase** represents card purchases and links to **Merchant** details.

- **Fee** manages additional service fees.
- **Adjustment** handles corrections or reversals made by the institution.

Overall, the diagram models how customer information, account data, and multiple transaction types interact to maintain a complete and verifiable credit card system.

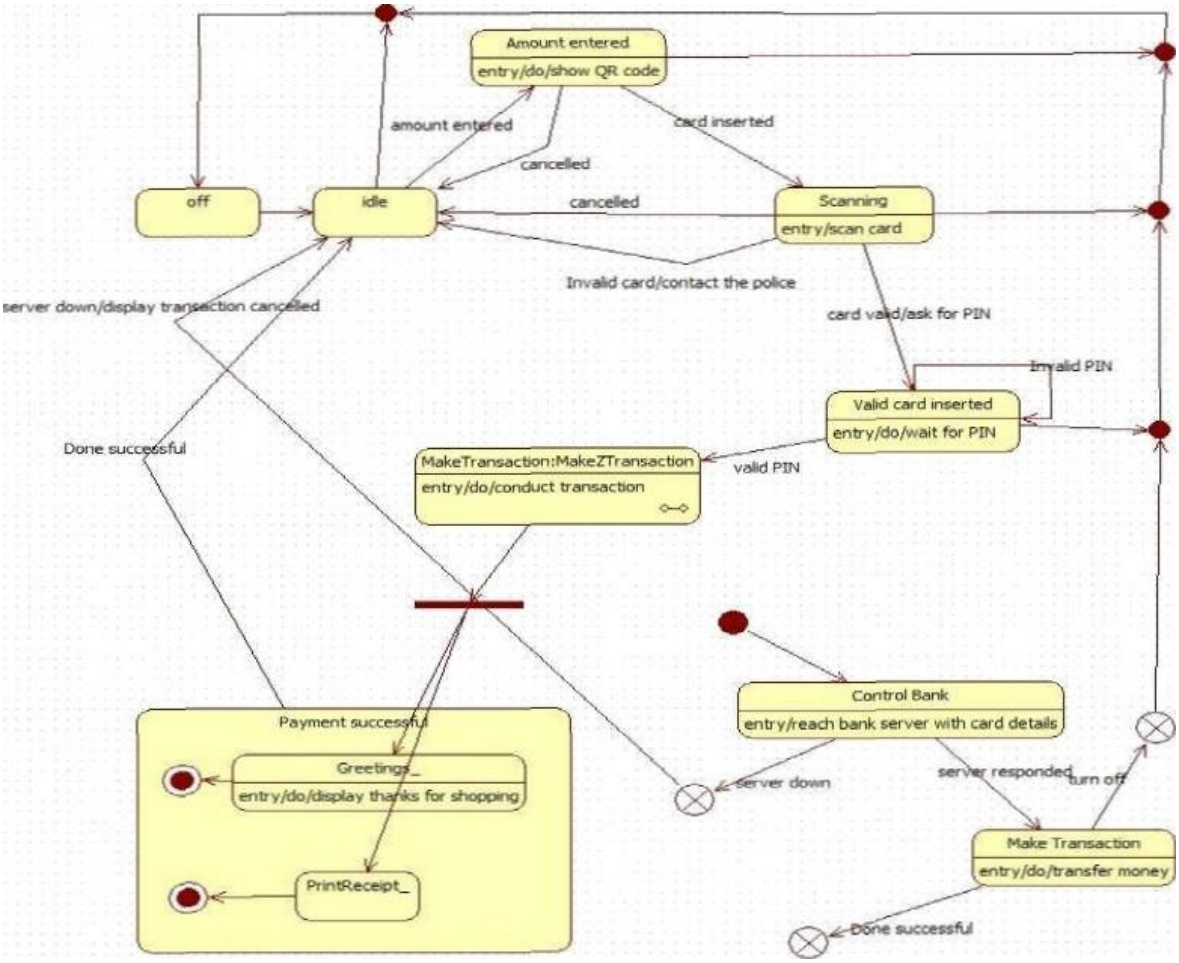
State Diagram (Simple):



Description:

The simple state diagram illustrates a typical payment processing workflow, starting from an idle system awaiting payment initiation, followed by capturing payment details and validating the card information. If the card is valid, the system seeks authorization from the bank, proceeding to capture funds upon approval or failing if the response is declined. After funds are captured, the system checks account sufficiency, settling and completing the transaction if there are enough funds, or initiating a refund process and notifying the customer in the case of insufficient balance, invalid card, or suspected fraud. The flow includes clear branches for failure handling and notification at each critical step, ensuring a robust and transparent payment process.

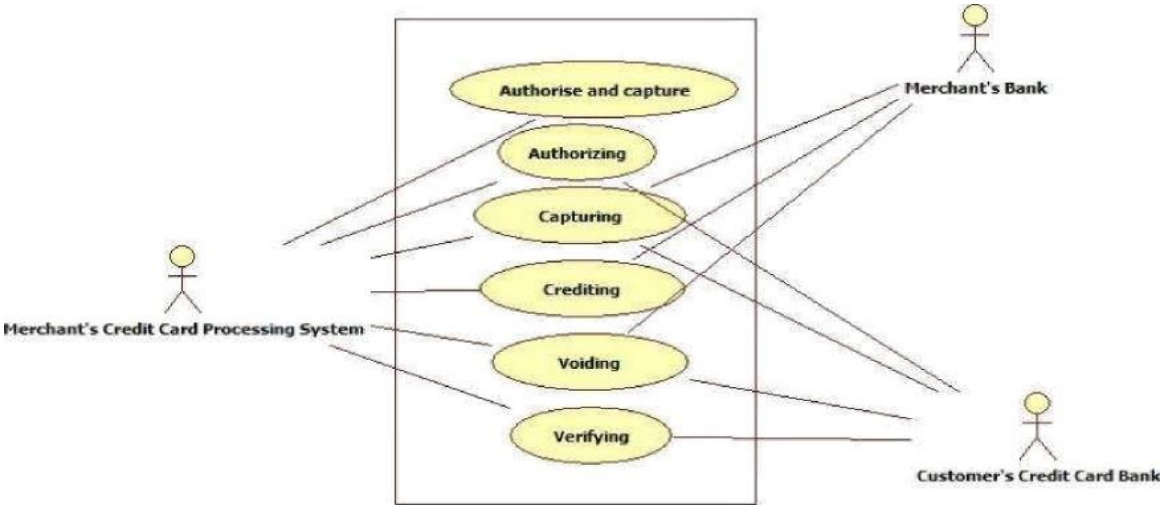
State Diagram (Advanced):



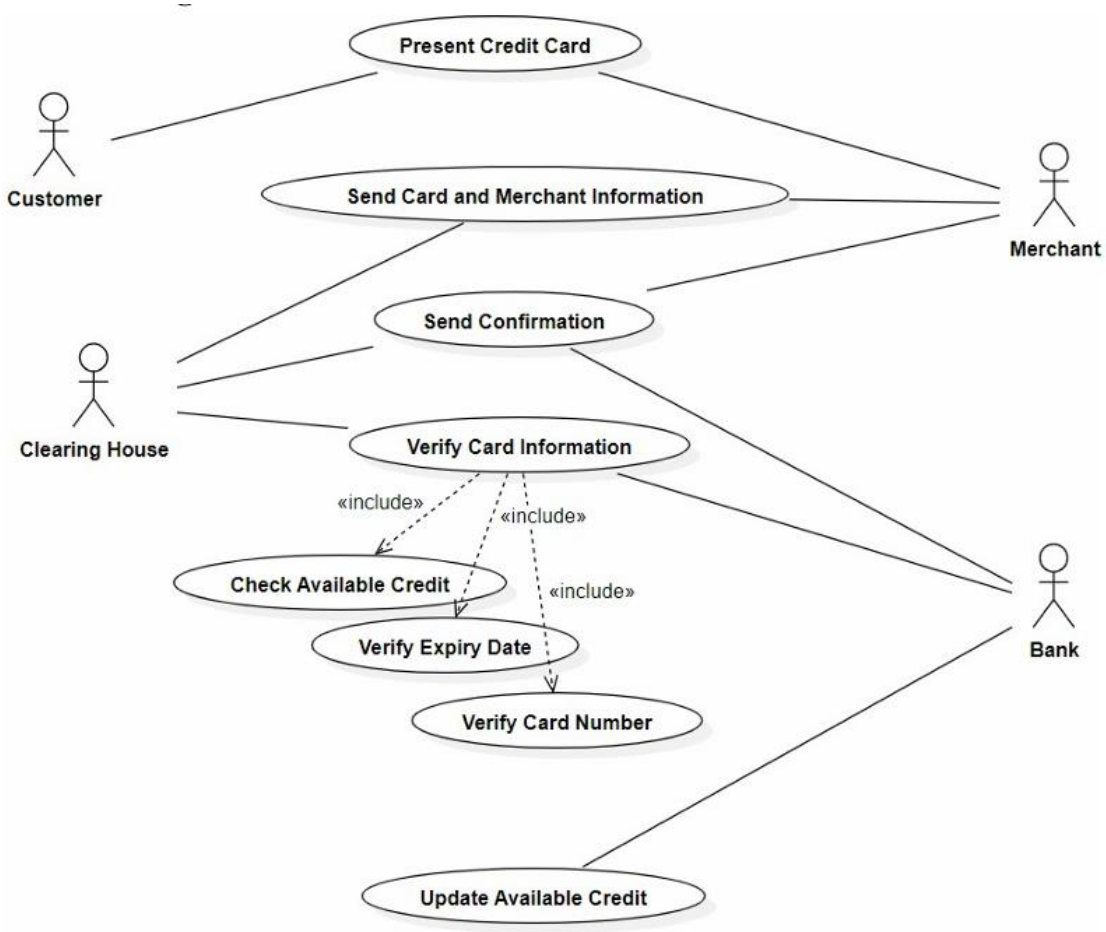
Description:

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

Use Case Diagram(Simple):



Use Case Diagram(Advanced):



Description:

Use Case Diagram(Simple):

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

Use Case Diagram(Advanced):

The advanced use case diagram presents a use case model for credit card transaction processing, showing the interactions between the customer, merchant, clearing house, and bank. The process starts when the customer presents the credit card to the merchant, who then sends the card and merchant information to the clearing house. The clearing house sends a confirmation and initiates card information verification, which includes checking available credit, verifying the expiry date, and confirming the card number. The bank is involved in these verification steps and finally updates the available credit based on the transaction outcome, ensuring accurate credit management and transaction validation among all parties involved.

Scenarios:

1: Make Payment

The customer selects the “Make Payment” option. The system displays the total bill amount.

The customer chooses a payment method (UPI, card, banking, etc.). The system sends the payment request to the payment gateway.

The customer enters payment details.

The payment gateway validates the details and processes the transaction. On success, the system updates the payment status.

A receipt is generated and shown to the customer.

2:Create Customer Account

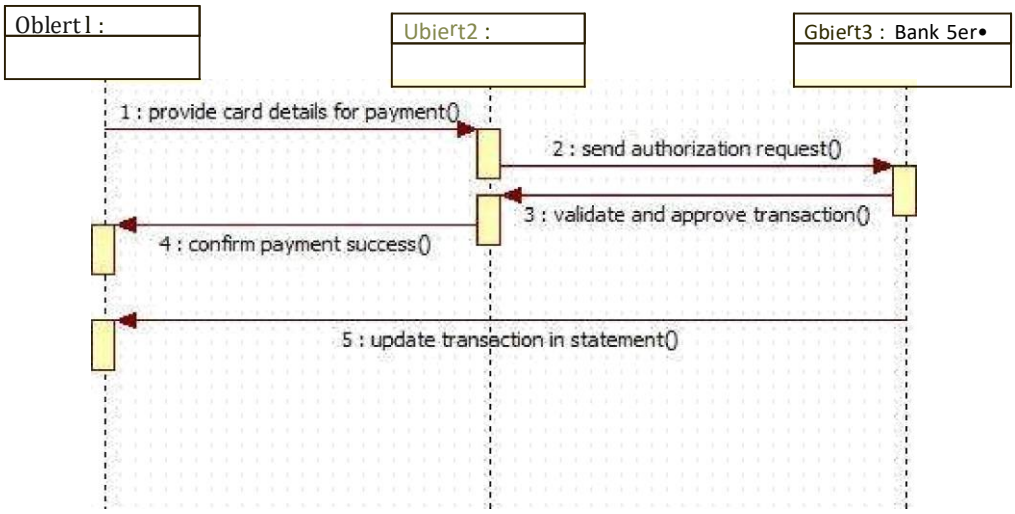
The admin selects the “Create Customer” option. The system displays a form to enter customer details.

The admin enters name, ID proof, contact details, and address. The system validates the information.

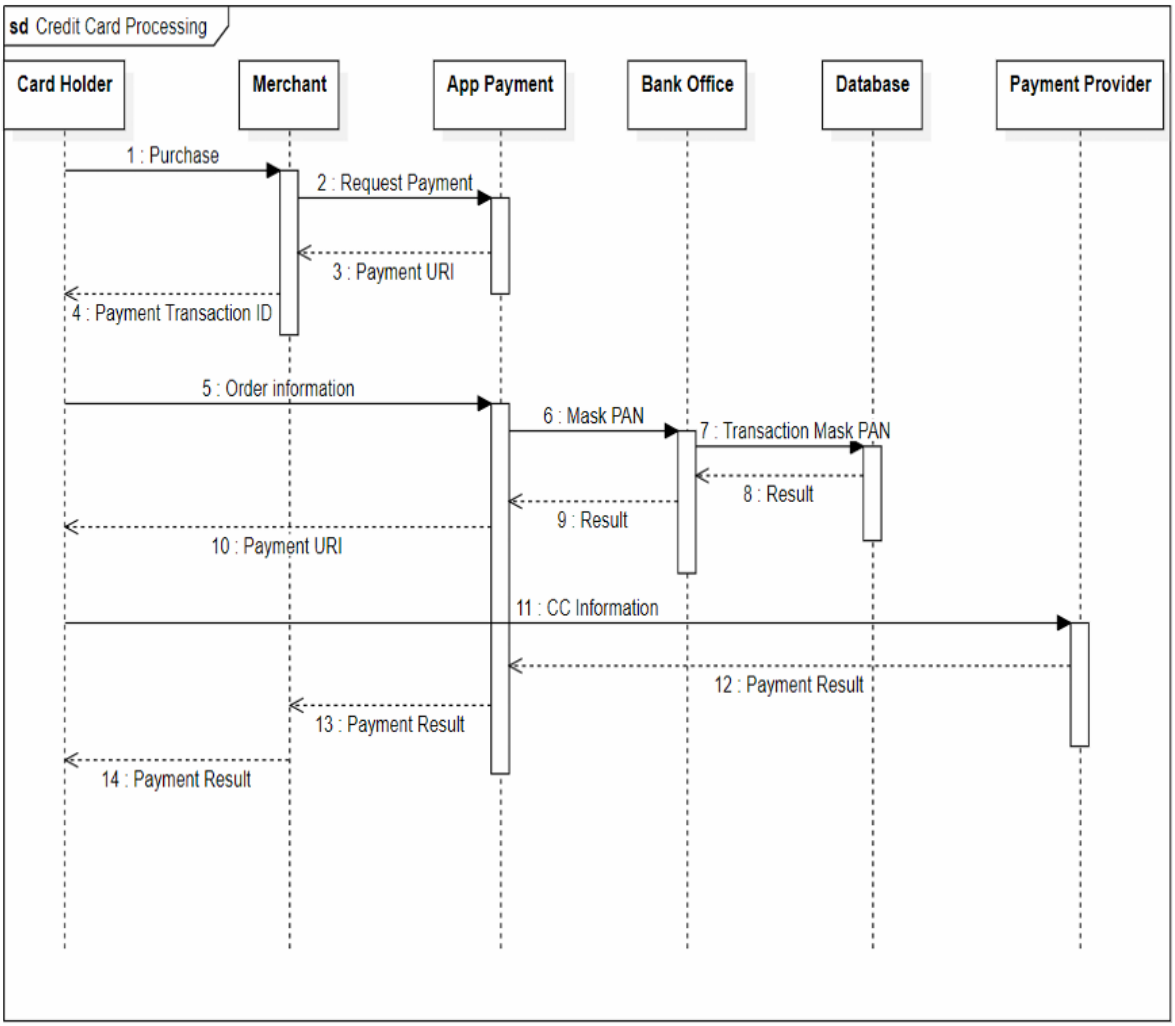
A unique customer ID is assigned automatically.

The customer account is created and stored in the database. The system confirms successful registration.

Sequence Diagram(Simple):



Sequence Diagram(Advanced):



Description:

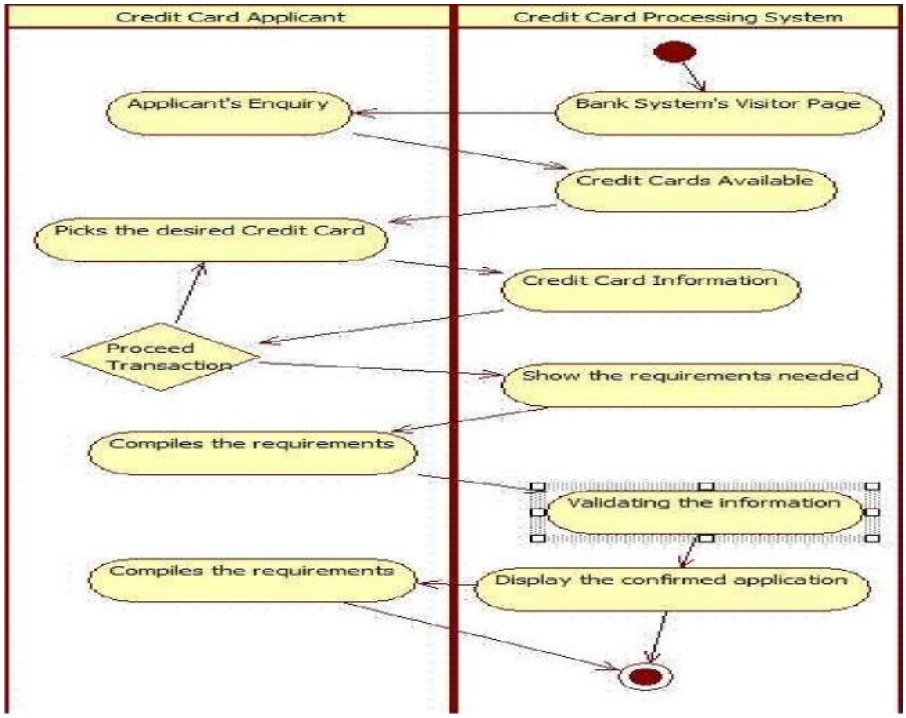
Sequence Diagram(Simple):

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

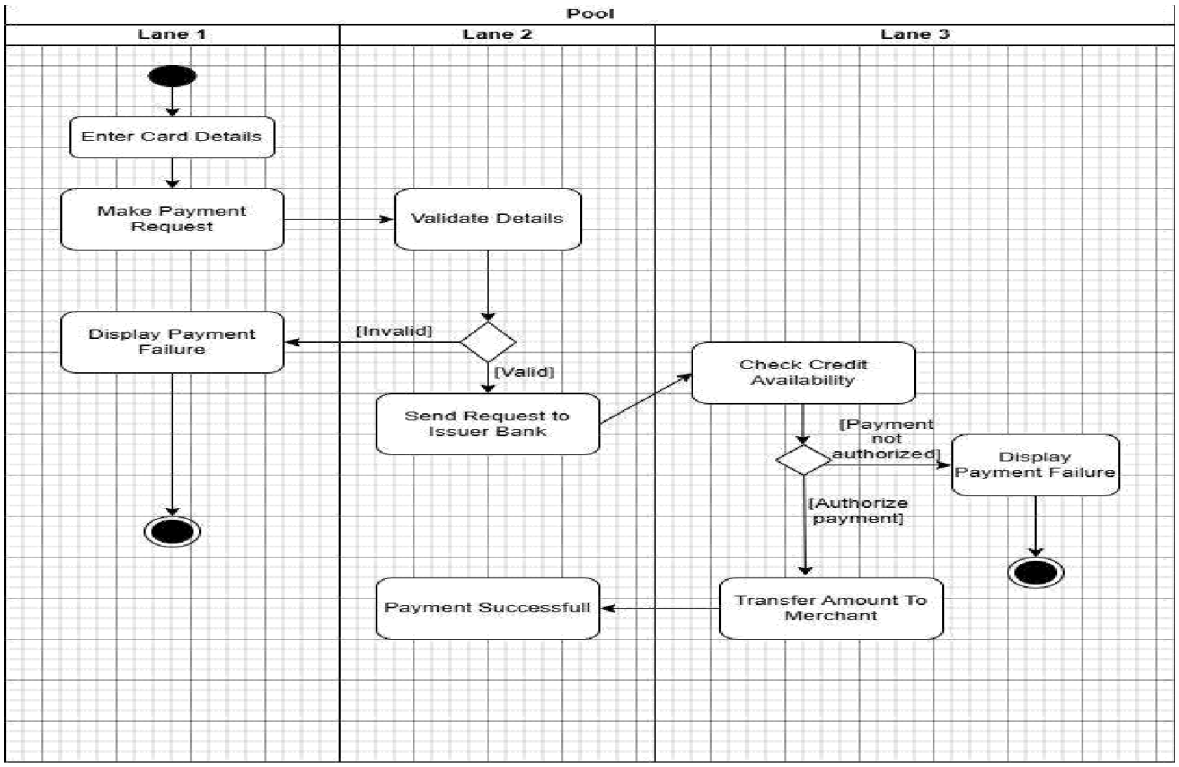
Sequence Diagram(Advanced):

The diagram depicts an advanced sequence for credit card processing involving six participants: Card Holder, Merchant, App Payment, Bank Office, Database, and Payment Provider. The process begins when the card holder initiates a purchase, prompting the merchant to request payment through the App Payment service. The App Payment entity coordinates with the Bank Office to mask the Primary Account Number (PAN), which then interacts with the database to process and mask the transaction details before returning results sequentially. Once verification and masking are complete, payment results and transaction information are relayed back through the chain, ultimately updating the card holder with the payment outcome. This detailed sequence captures real-world payment flows, security handling, and intricate system interplay essential for secure credit card transactions.

Activity Diagram(Simple):



Activity Diagram(Advanced):



Description:

Activity Diagram(Simple):

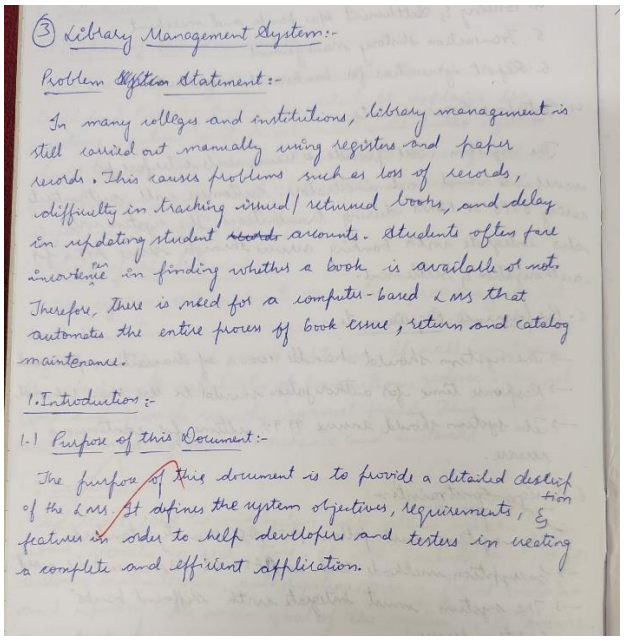
The simple activity diagram explains the basic steps in completing a credit card payment. The process starts when the customer enters card details and submits a payment request. The system then validates the details. If the card information is invalid, the system immediately displays a payment failure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

Activity Diagram(Advanced):

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

3. Library Management System

SRS Document:



③ Library Management System:-

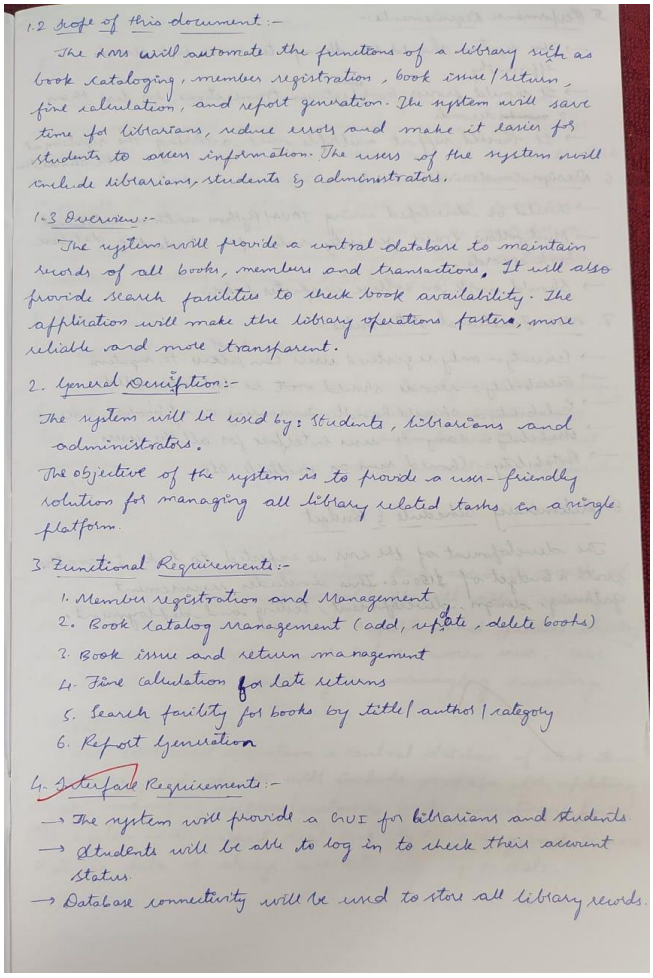
Problem Statement:-

In many colleges and institutions, library management is still carried out manually using registers and paper records. This causes problems such as loss of records, difficulty in tracking issued/returned books, and delay in updating student accounts. Students often face inconvenience in finding whether a book is available or not. Therefore, there is need for a computer-based LMS that automates the entire process of book issue, return and catalog maintenance.

1. Introduction:-

1.1 Purpose of this Document:-

The purpose of this document is to provide a detailed description of the LMS. It defines the system objectives, requirements, & features in order to help developers and testers in creating a complete and efficient application.



1.2 Scope of this document:-

The LMS will automate the functions of a library such as book cataloging, member registration, book issue/return, fine calculation, and report generation. The system will save time for librarians, reduce errors and make it easier for students to access information. The users of the system will include librarians, students & administrators.

1.3 Overview:-

The system will provide a central database to maintain records of all books, members and transactions. It will also provide search facilities to check book availability. The application will make the library operations faster, more reliable and more transparent.

2. General Description:-

The system will be used by: Students, librarians and administrators.

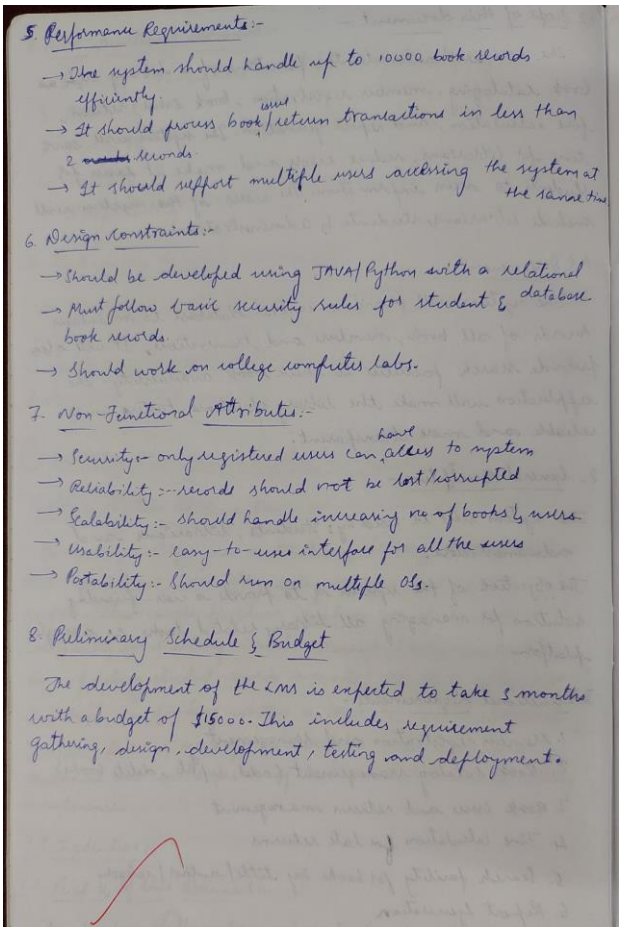
The objective of the system is to provide a user-friendly solution for managing all library related tasks on a single platform.

3. Functional Requirements:-

1. Member registration and management
2. Book catalog management (add, update, delete books)
3. Book issue and return management
4. Fine calculation for late returns
5. Search facility for books by title/author/category
6. Report generation

4. Interface Requirements:-

→ The system will provide a GUI for librarians and students.
→ Students will be able to log in to check their account status.
→ Database connectivity will be used to store all library records.



5. Performance Requirements:-

→ The system should handle up to 10000 book records efficiently.
→ It should process book issue/return transactions in less than 2 seconds.
→ It should support multiple users accessing the system at the same time.

6. Design Constraints:-

→ Should be developed using JAVA/Python with a relational database.
→ Must follow basic security rules for student & book records.
→ Should work on college computer labs.

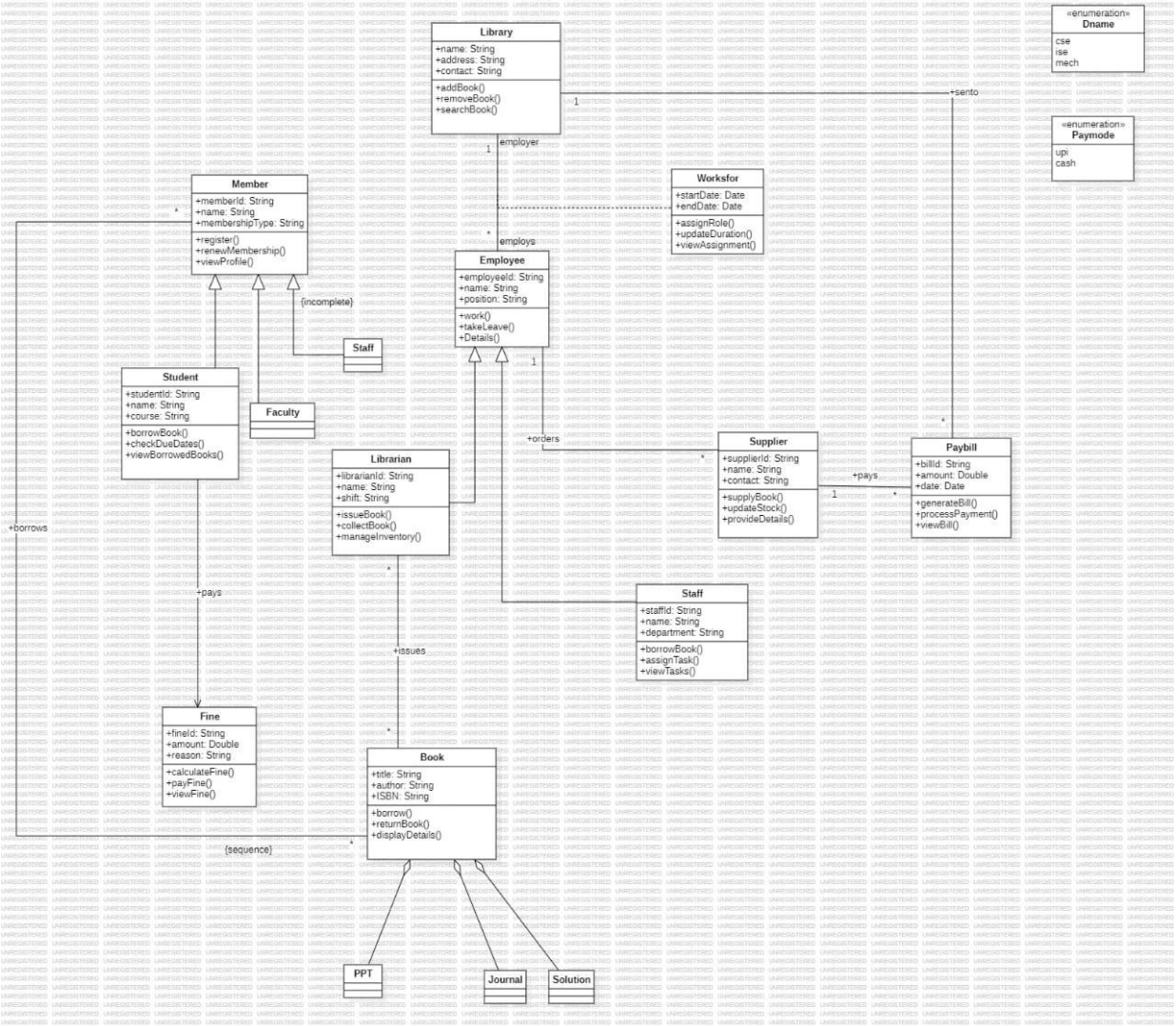
7. Non-Functional Attributes:-

→ Security:- only registered users can have access to system
→ Reliability:- records should not be lost/corrupted
→ Scalability:- should handle increasing no of books & users
→ Usability:- easy-to-use interface for all the users
→ Portability:- Should run on multiple OSs.

8. Preliminary Schedule & Budget

The development of the LMS is expected to take 3 months with a budget of $15000. This includes requirement gathering, design, development, testing and deployment.

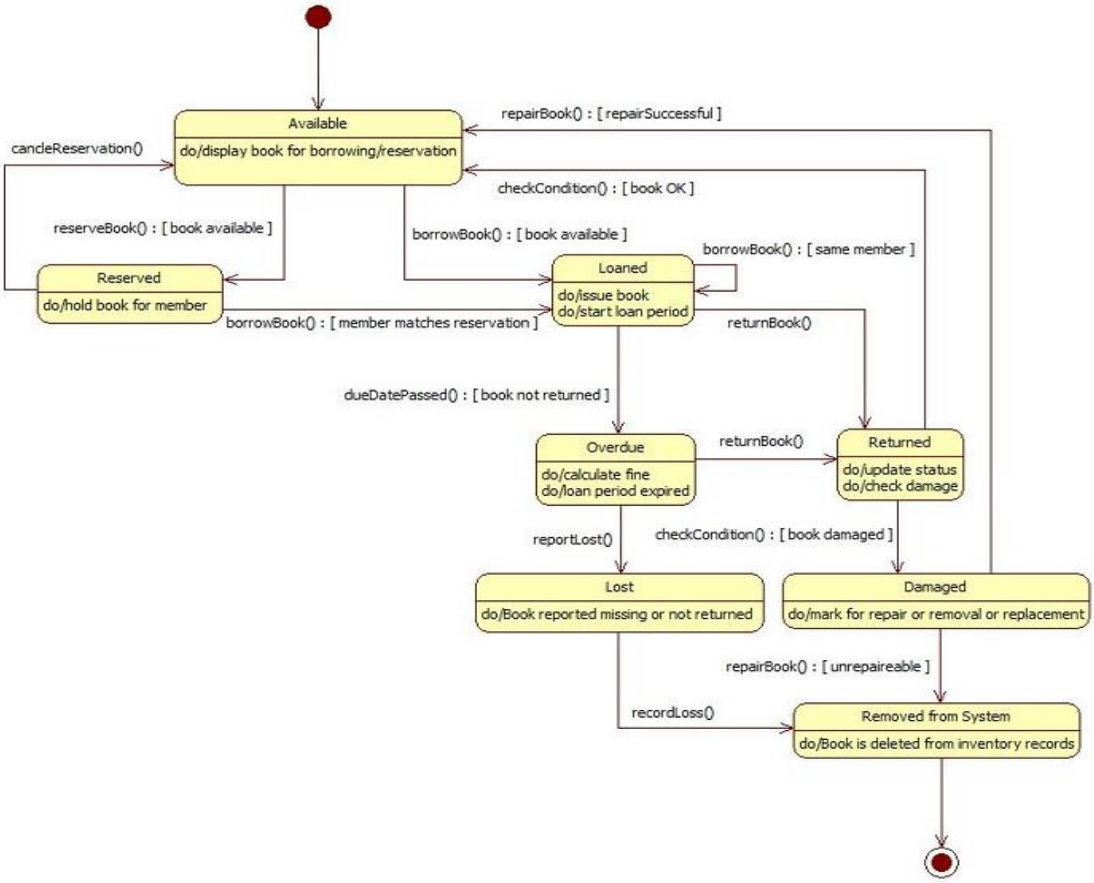
Class Diagram:



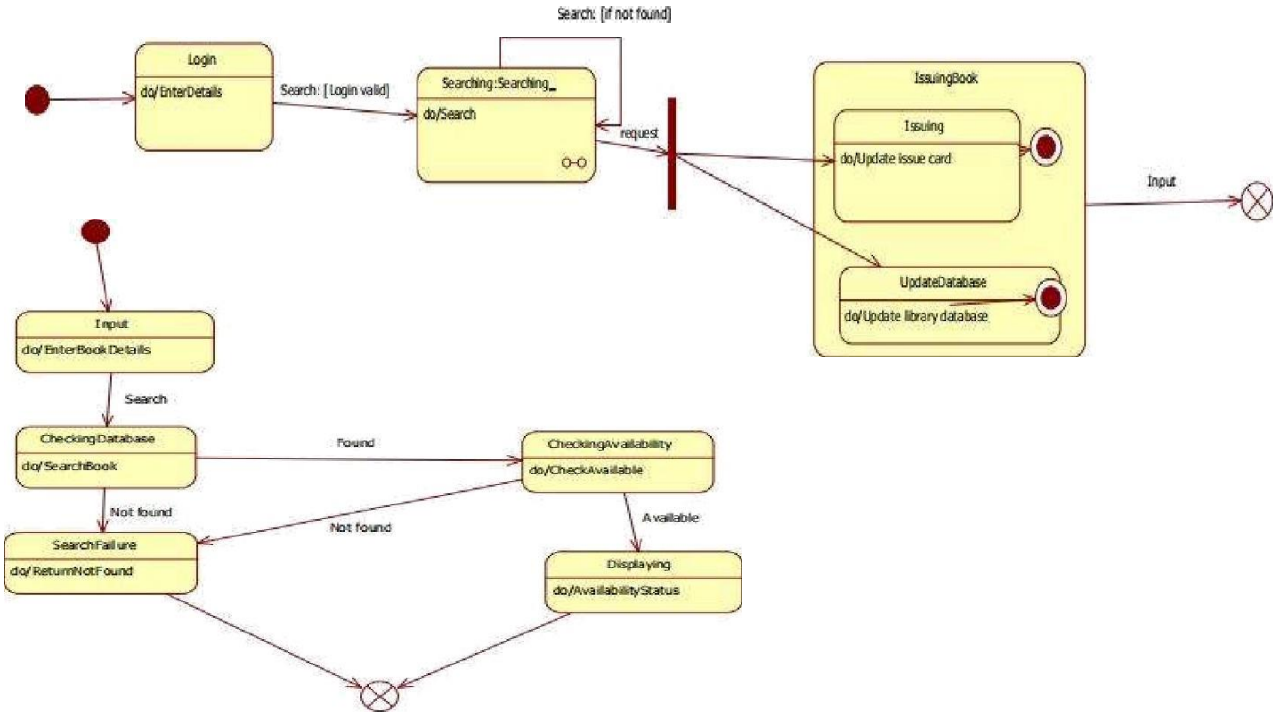
Description:

The diagram is a comprehensive UML class diagram for a library management system, showcasing entities such as Library, Member (with subclasses Student and Faculty), Employee (with subclasses Librarian and Staff), Book (associated with categories like PPT, Journal, Solution), Supplier, Fine, and Paybill. It outlines relationships and interactions: Members can borrow books, pay fines, and manage their profiles; Employees, including Librarians, handle book issuing, inventory, and library tasks; Suppliers provide books and details, while Paybill manages payment processing with various payment modes (cash, UPI, etc). The diagram establishes associations, inheritances, and connections between entities, accurately modeling the complex processes and roles involved in library operations.

State Diagram(Simple):



State Diagram(Advanced):



Description:

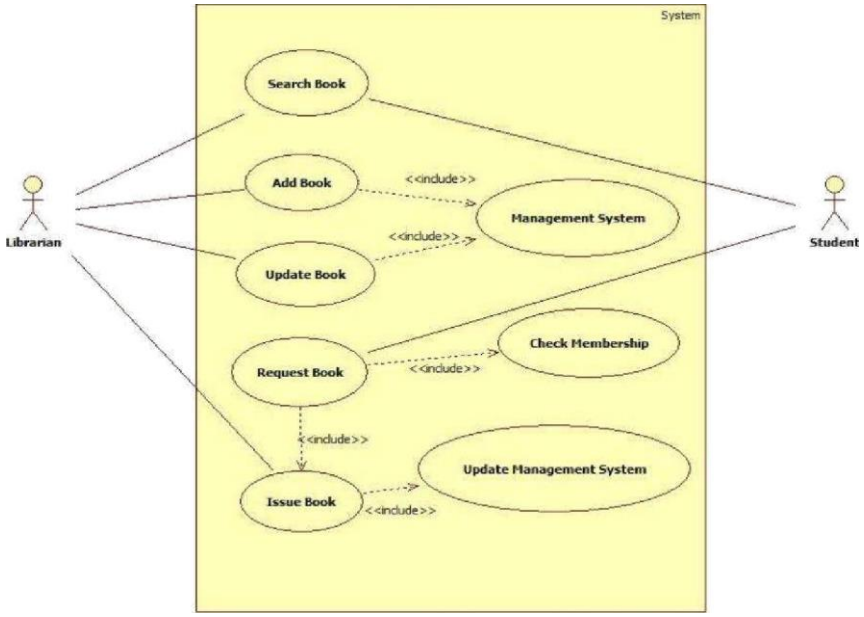
State Diagram(Simple):

The diagram is a state transition diagram that models the lifecycle of a book in a library system. It begins with the book in the "Available" state, where it can be borrowed or reserved. When reserved, it moves to the "Reserved" state, and if borrowed, transitions to "Loaned." If the loan is not returned by the due date, the book enters the "Overdue" state, where fines are calculated. A book that is reported missing or not returned becomes "Lost," while a returned book is evaluated and, if damaged, marked as "Damaged" for repair or removal. If irreparable or officially recorded as lost, the book moves to "Removed from System," indicating it is deleted from inventory. Throughout, the diagram defines events and actions for each transition, such as issuing books, checking conditions, calculating fines, repairing, and updating statuses.

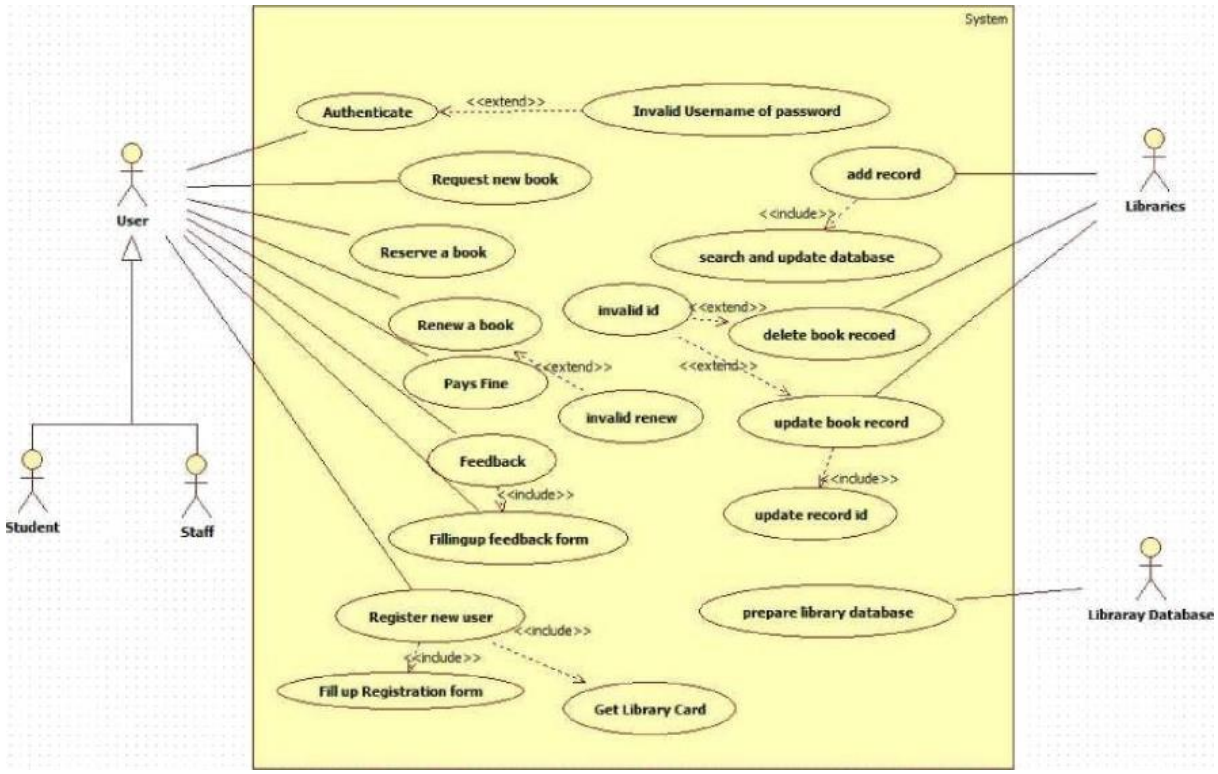
State Diagram(Advanced):

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

Use Case Diagram(Simple):



Use Case Diagram(Advanced)



Description:

Use Case Diagram(Simple):

The simple use case diagram focuses on the core interactions in the Library Management System between the Librarian and the Student. The librarian performs essential tasks such as Searching Books, Adding Books, Updating Book Details, Requesting Books, and Issuing Books. Many of these actions include interactions with the Management System, such as checking membership and updating records. The student mainly interacts through the system to request or search for books. The diagram highlights only the fundamental operations that support the basic flow of library activities—finding, adding, updating, and issuing books—making it easy to understand the primary responsibilities of the librarian and how students request books.

Use Case Diagram(Advanced):

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

Scenarios

1: Issue Book

The librarian selects the “Issue Book” option.

The librarian enters the member ID and book ID.

The system checks the member's status and borrowing limit.

The system verifies the availability of the book.

If valid, the system updates the issue record and marks the book as borrowed.

The librarian hands the book to the member.

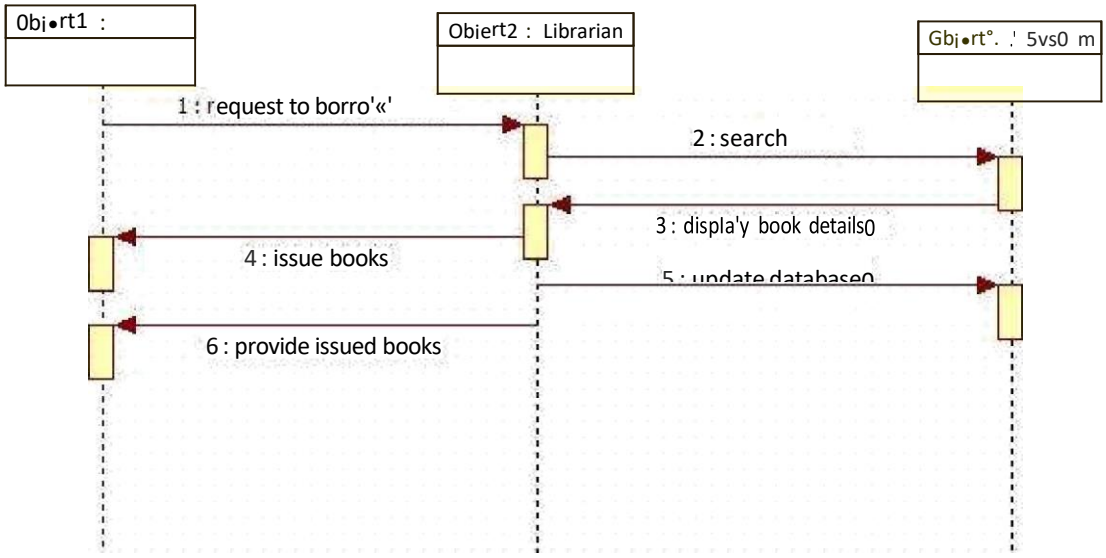
2: Search Book

The student selects the “Search Book” option. The system prompts for the book title/author/ID. The student enters the details.

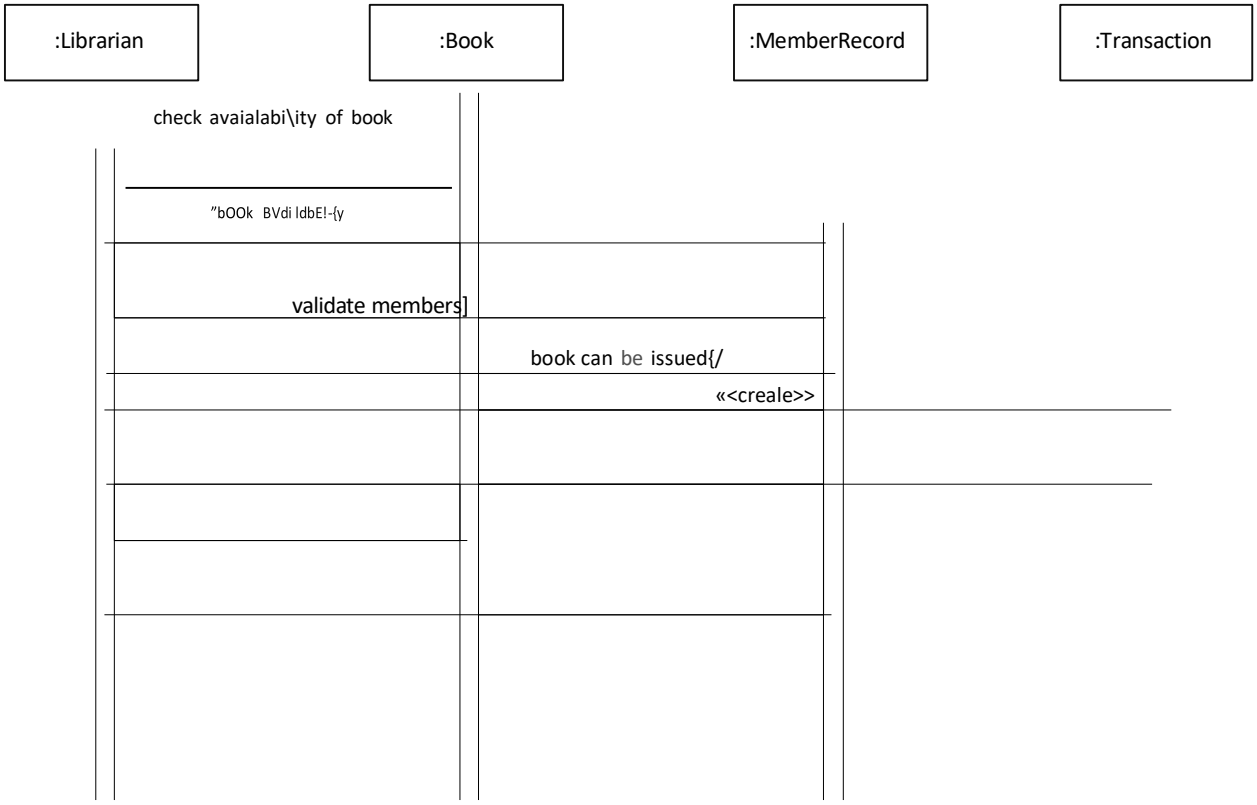
The system scans the database for matching books.

The system displays the availability status (Available / Issued / Not Found).

Sequence Diagram(Simple):



Sequence Diagram(Advanced):



Description:

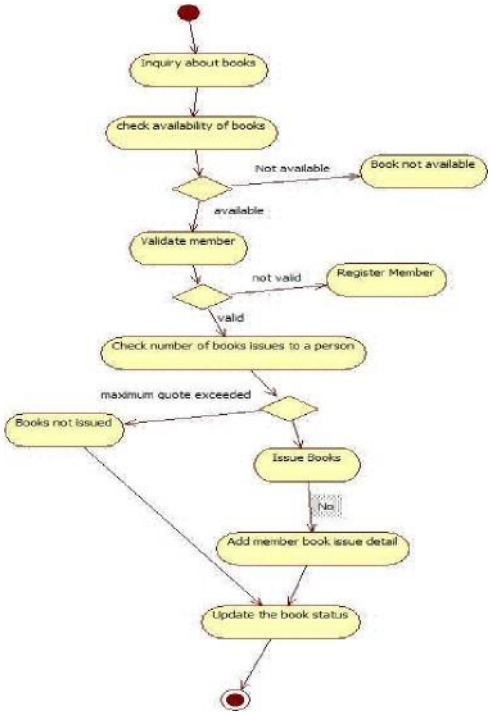
Sequence Diagram (Simple): (Member — Librarian — System)

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

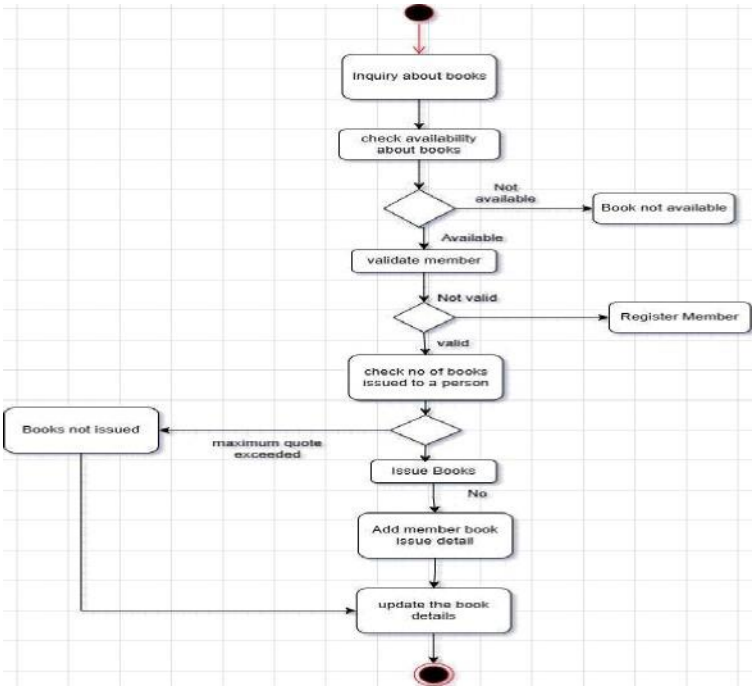
Sequence Diagram (Advanced): (Librarian — Book — MemberRecord — Transaction)

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

Activity Diagram(Simple)



Activity Diagram(Advanced)



Description:

Activity Diagram(Simple):

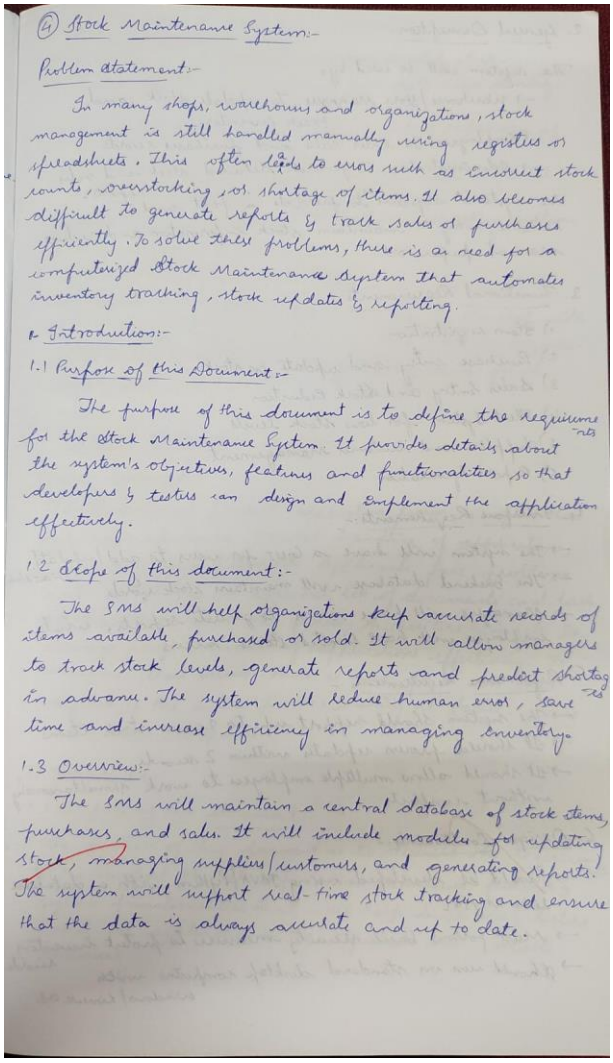
The simple activity diagram shows the basic steps followed when a user wants to issue a book. The process starts with an inquiry about the book, followed by checking book availability. If the book is not available, the process ends. If it is available, the system then validates the member. If the member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

Activity Diagram(Advanced):

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

4. Stock Maintenance System

SRS Document:



④ Stock Maintenance System:-

Problem statement:-

In many shops, warehouses and organizations, stock management is still handled manually using registers or spreadsheets. This often leads to errors such as incorrect stock counts, overstocking, or shortage of items. It also becomes difficult to generate reports & track sales or purchases efficiently. To solve these problems, there is an need for a computerized Stock Maintenance System that automates inventory tracking, stock updates & reporting.

1. Introduction:-

1.1 Purpose of this Document:-

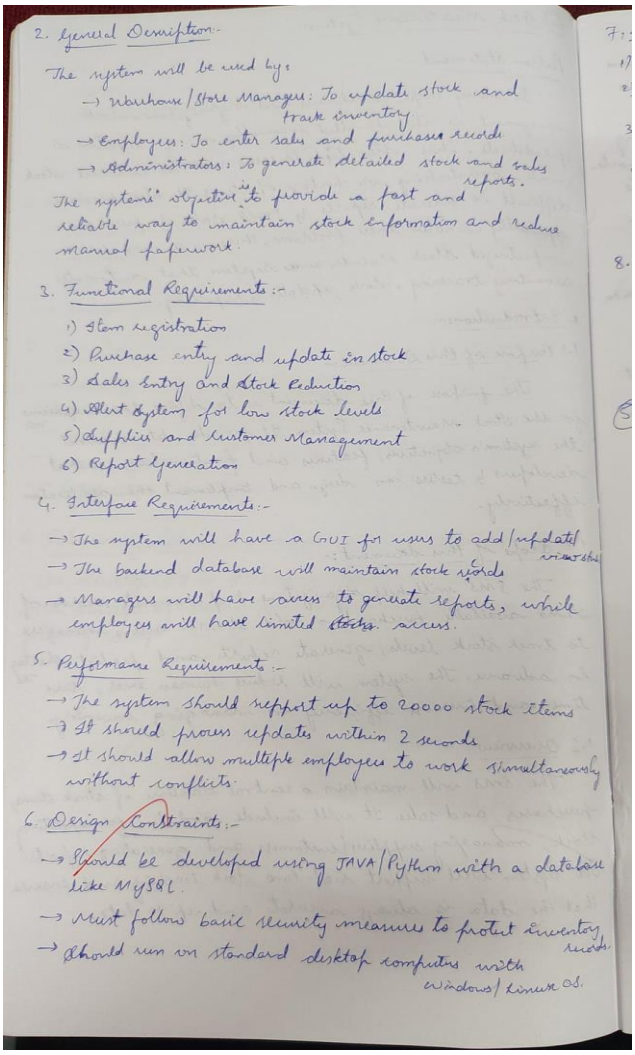
The purpose of this document is to define the requirements for the Stock Maintenance System. It provides details about the system's objectives, features and functionalities so that developers & testers can design and implement the application effectively.

1.2 Scope of this document:-

The SMS will help organizations keep accurate records of items available, purchased or sold. It will allow managers to track stock levels, generate reports and predict shortages in advance. The system will reduce human errors, save time and increase efficiency in managing inventory.

1.3 Overview:-

The SMS will maintain a central database of stock items, purchases, and sales. It will include modules for updating stock, managing suppliers/customers, and generating reports. The system will support real-time stock tracking and ensure that the data is always accurate and up to date.



2. General Description:-

The system will be used by:
→ Warehouse/Store Managers: To update stock and track inventory
→ Employees: To enter sales and purchase records
→ Administrators: To generate detailed stock and sales reports.

The system's objective is to provide a fast and reliable way to maintain stock information and reduce manual paperwork.

3. Functional Requirements:-

1) Item registration
2) Purchase entry and update in stock
3) Sales Entry and Stock Reduction
4) Alert system for low stock levels
5) Suppliers and Customer Management
6) Report Generation

4. Interface Requirements:-

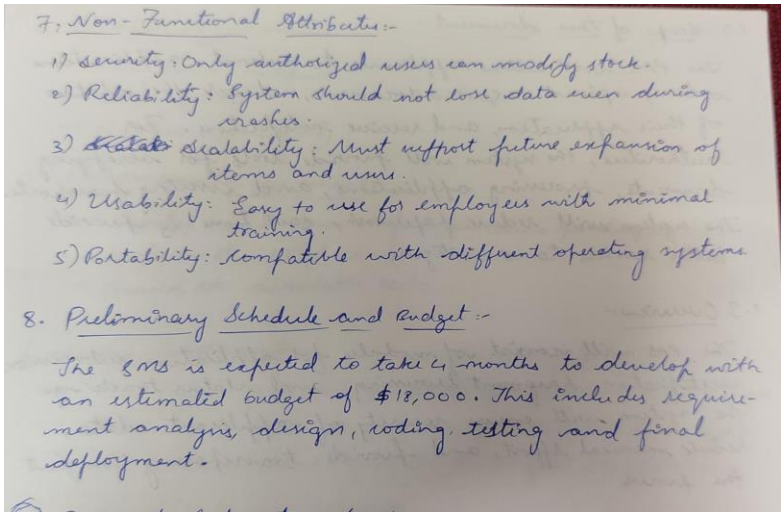
→ The system will have a GUI for users to add/update/view items
→ The backend database will maintain stock records
→ Managers will have access to generate reports, while employees will have limited access.

5. Performance Requirements:-

→ The system should support up to 20000 stock items
→ It should process updates within 2 seconds
→ It should allow multiple employees to work simultaneously without conflicts.

6. Design Constraints:-

→ Should be developed using JAVA/Python with a database like MySQL
→ Must follow basic security measures to protect inventory records.
→ Should run on standard desktop computers with Windows/Linux OS.



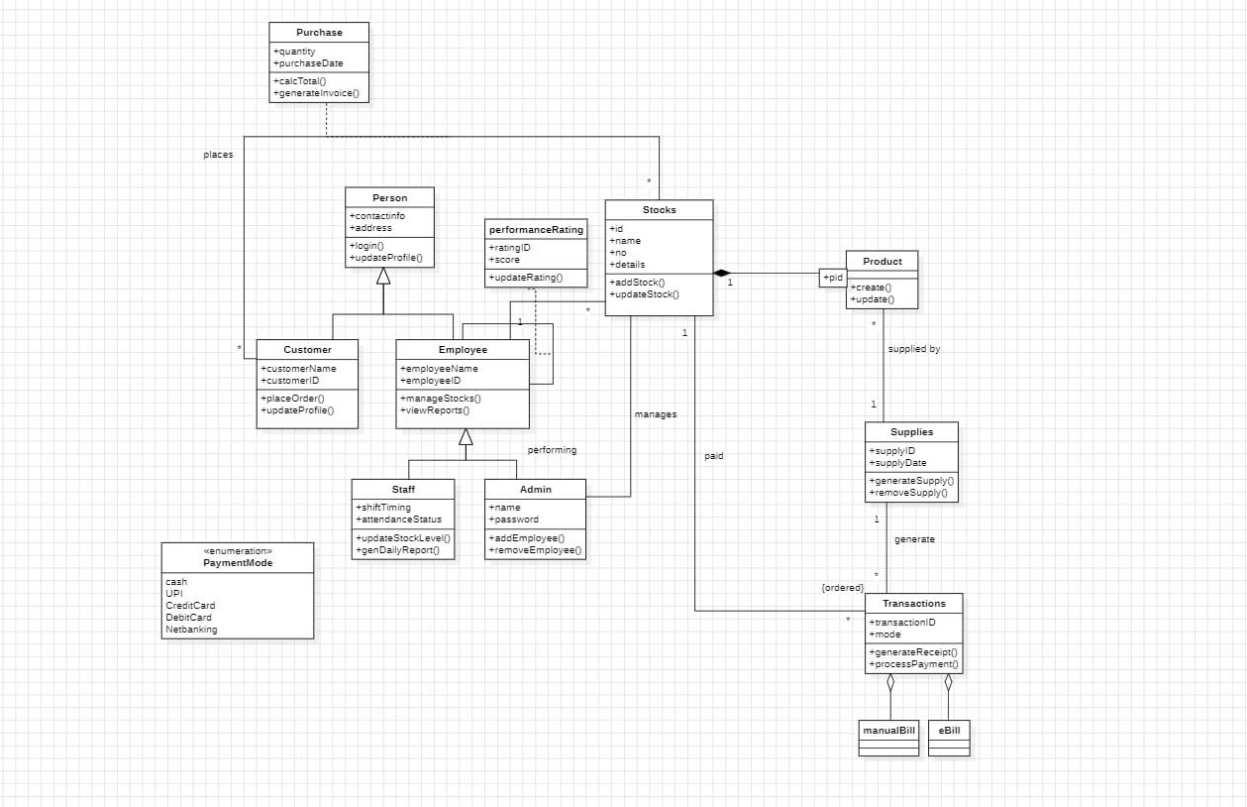
7. Non-Functional Attributes:-

1) Security: Only authorized users can modify stock.
2) Reliability: System should not lose data even during crashes.
3) ~~Scalability~~ Scalability: Must support future expansion of items and users.
4) Usability: Easy to use for employees with minimal training.
5) Portability: Compatible with different operating systems

8. Preliminary Schedule and Budget:-

The SMS is expected to take 4 months to develop with an estimated budget of $18,000. This includes requirement analysis, design, coding, testing and final deployment.

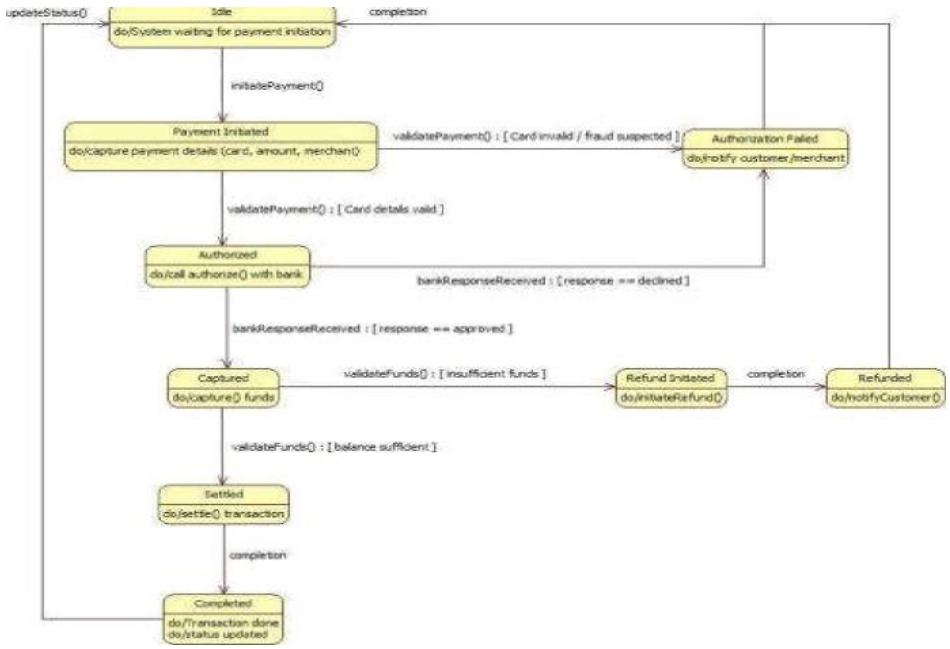
Class Diagram:



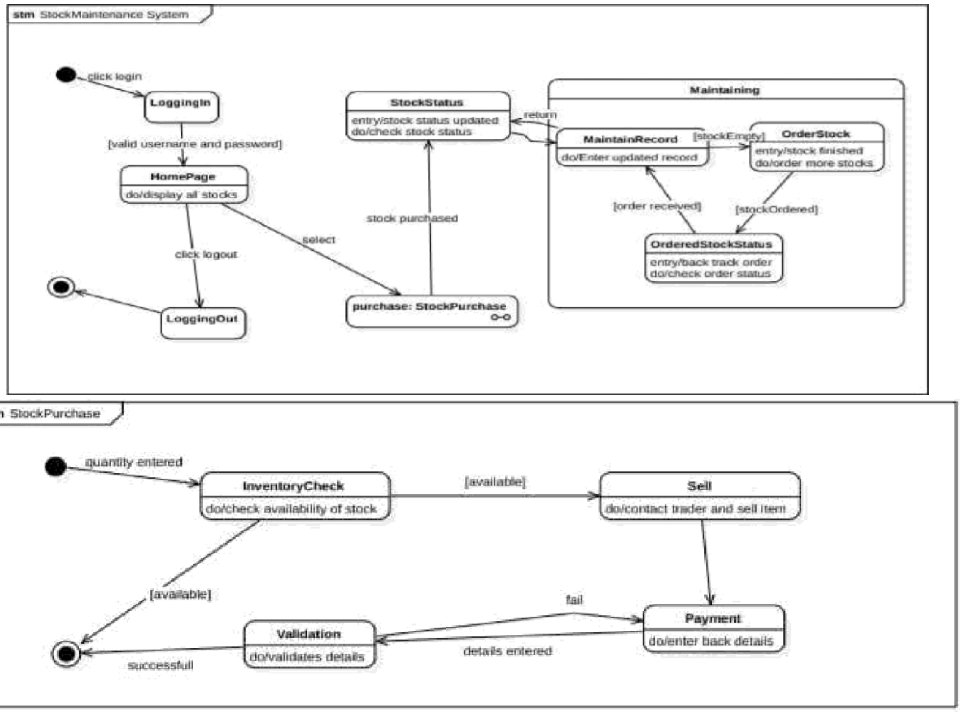
Description:

The diagram represents a UML class diagram for an inventory and transaction management system. It models entities including Person (superclass for Customer and Employee), performanceRating, Stocks, Product, Supplies, Transactions (supporting manual and electronic bills), and Purchase. Customers can place orders that affect stock levels, while Employees (including Staff and Admin) manage stocks, generate reports, and handle employee management. Products are managed in stocks and supplied by Supplies, linking each product with supply records. The Transactions class supports multiple payment modes (cash, UPI, credit/debit card, netbanking) and is responsible for generating receipts and processing payments. The system incorporates relationships for placing orders, managing and updating stocks, performing and tracking supplies, and processing purchases and transactions in a detailed way.

State Diagram(Simple):



State Diagram(Advanced):



Description:

State Diagram(Simple):

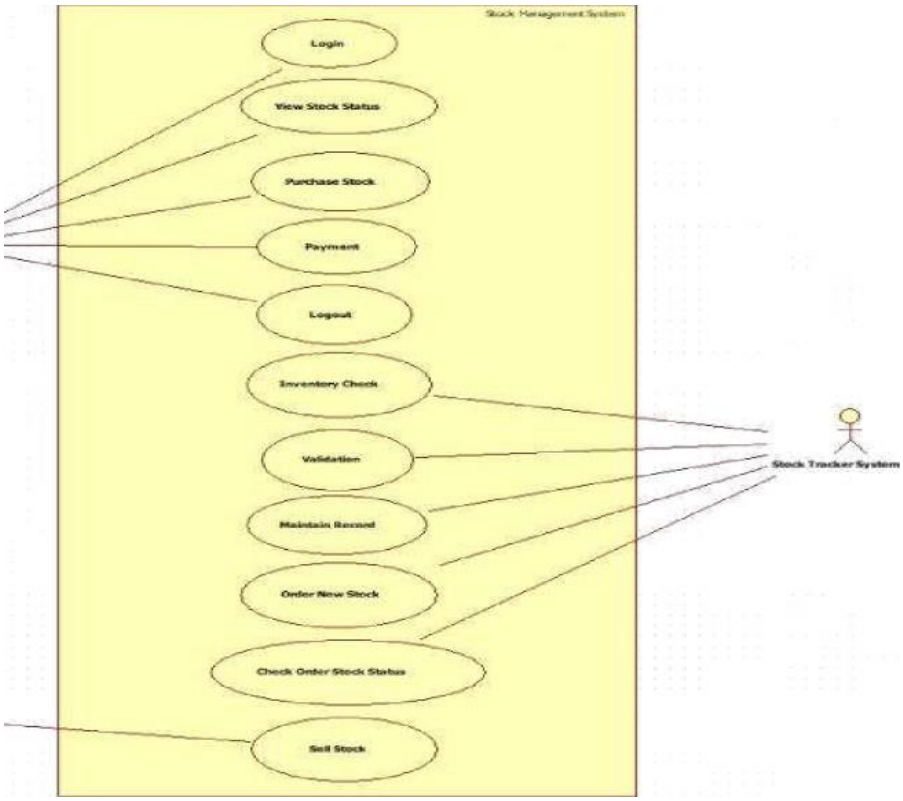
This state diagram illustrates the various stages an item goes through in an inventory or order management system, from being available to the possibility of being out of stock. The process begins when an item is requested, transitioning the item from "Available" to "Reserved" if the quantity is sufficient, or returning to "Available" if a reservation is canceled. After payment, the item is "Dispatched" and moves to "Delivered" once confirmed by the customer. If returned within the allowed period, it enters the "Returned" state for inspection and potential restocking. At any point, if stock runs out, the item moves to the "Out of Stock" state, where reordering and restocking actions are managed.

State Diagram(Advanced):

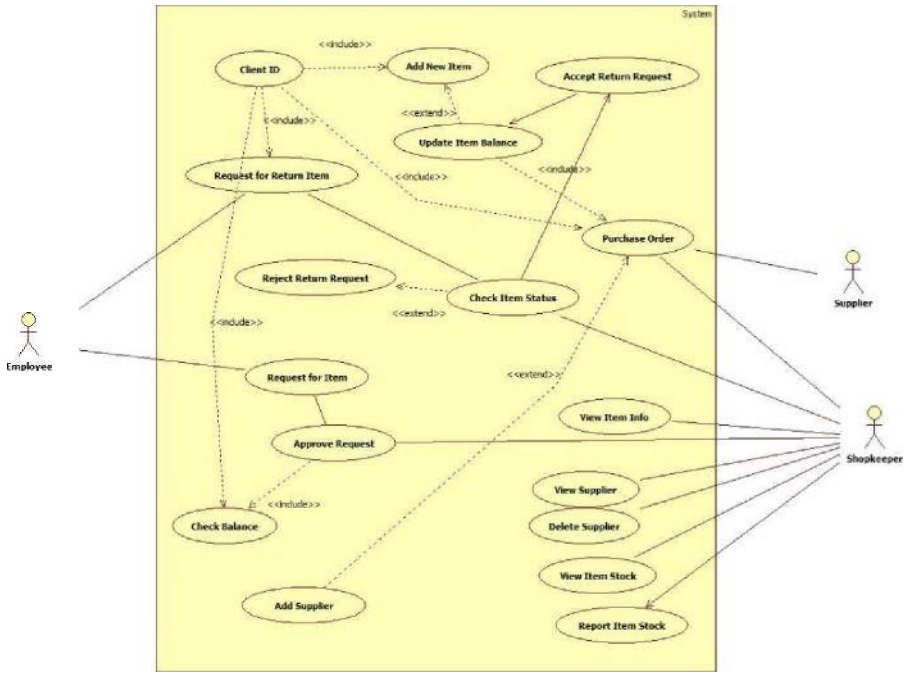
This advanced state diagram models the Stock Maintenance System, focusing on two core processes: managing overall stock status and handling stock purchase transactions. The top section illustrates user actions like logging in, navigating the homepage to view stocks, initiating stock purchases, and logging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

The lower section details the Stock Purchase process, starting when a quantity is entered. The flow continues with an inventory check; if stock is available, the item is sold and the system validates and processes the payment. If any details are invalid, the process prompts re-entry of information. Successful transactions complete the state flow, while unavailability or validation failure leads to process termination. Overall, the diagram effectively captures the operational logic and interdependencies involved in maintaining stock and processing purchases within an inventory system.

Use Case Diagram(Simple):



Use Case(Advanced):



Description:

Use Case Diagram(Simple):

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

Use Case(Advanced):

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

•t• The diagram maps three actors: Manager, Store Staff, and Supplier, each with distinct responsibilities. The Manager handles buying stock, making payments, and supervising staff.

t• Store Staff are responsible for inventory reporting, product quality checks, and selling stock.

•t• The Supplier delivers orders, receives payments, and manages quality issue reports. t• <> and <> relationships show functional dependencies between use cases.

•t• For example, ”Buy Stock" includes ”Give Payment" and may extend to ”Report quality issue to supplier."

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 stuff 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

2: Stock Rejection and Return (Exception Scenario)

Supplier delivers 300 units of Item-Y, but 80 units found damaged

Warehouse stuff performs quality check — 8m units United

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

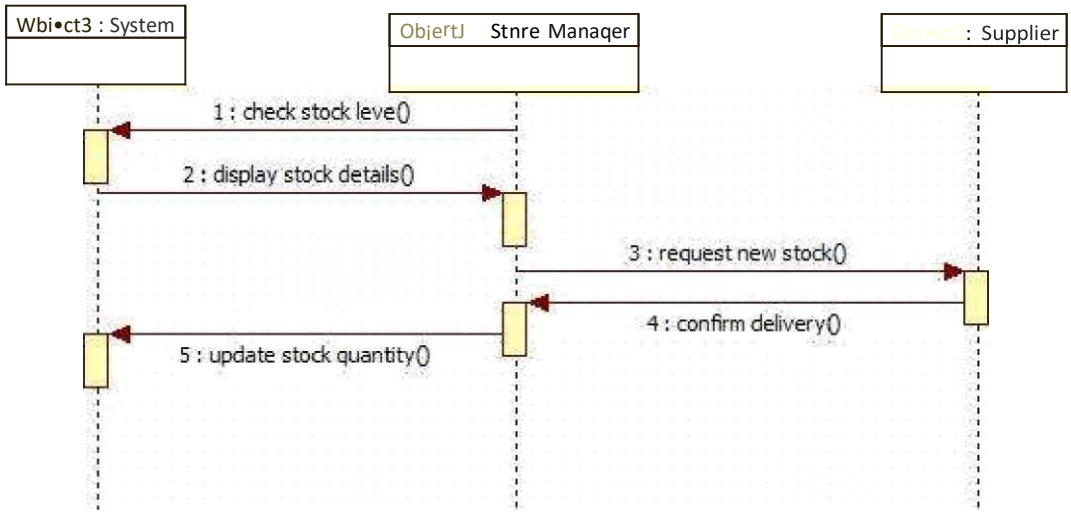
Stock level of Item-Y updated correctly (only +220)

Supplier acknowledges return, issues Credit Note

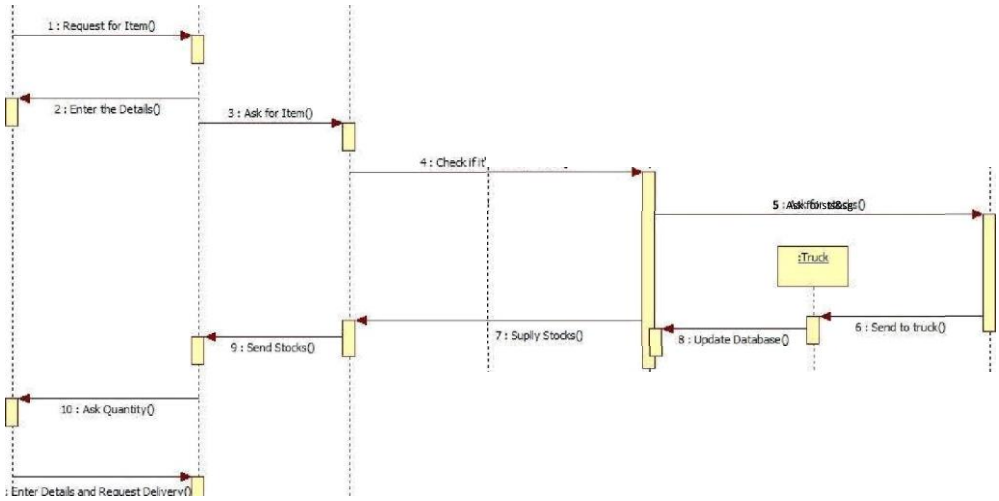
Inventory report reflects accurate stock and pending returns

Low-stock alert remains active since actual receipt is below requirement

Sequence Diagram(Simple):



Sequence Diagram(Advanced):



Description:

Sequence Diagram(Simple):

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

Sequence Diagram(Advanced):

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

•t• Customer details are validated before login is granted.

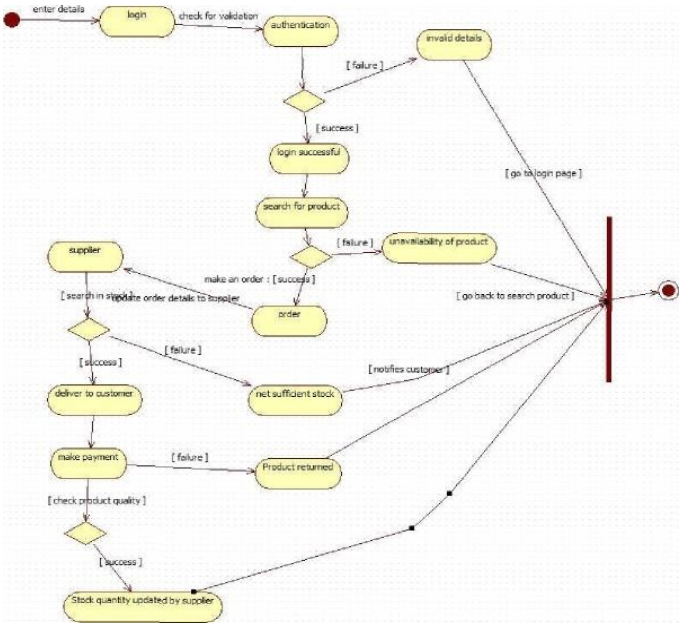
•t• Product search triggers quantity check in stock.

•t• If stock is unavailable, customer is prompted to choose a new product.

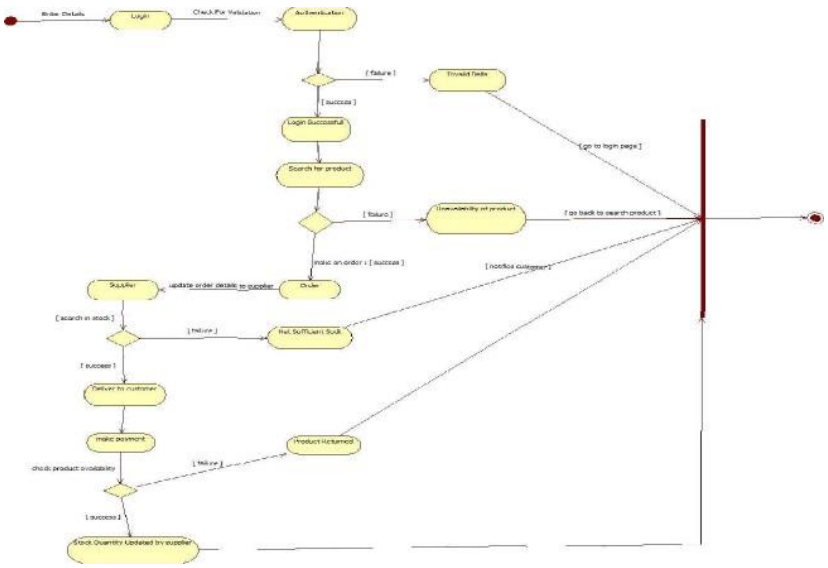
•t• Supplier receives order details and delivers to the customer.

•t• Payment, quantity update, and refund (if needed) complete the transaction flow

Activity Diagram(Simple):



Activity Diagram(Advanced):



Description:

Activity Diagram(Simple):

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

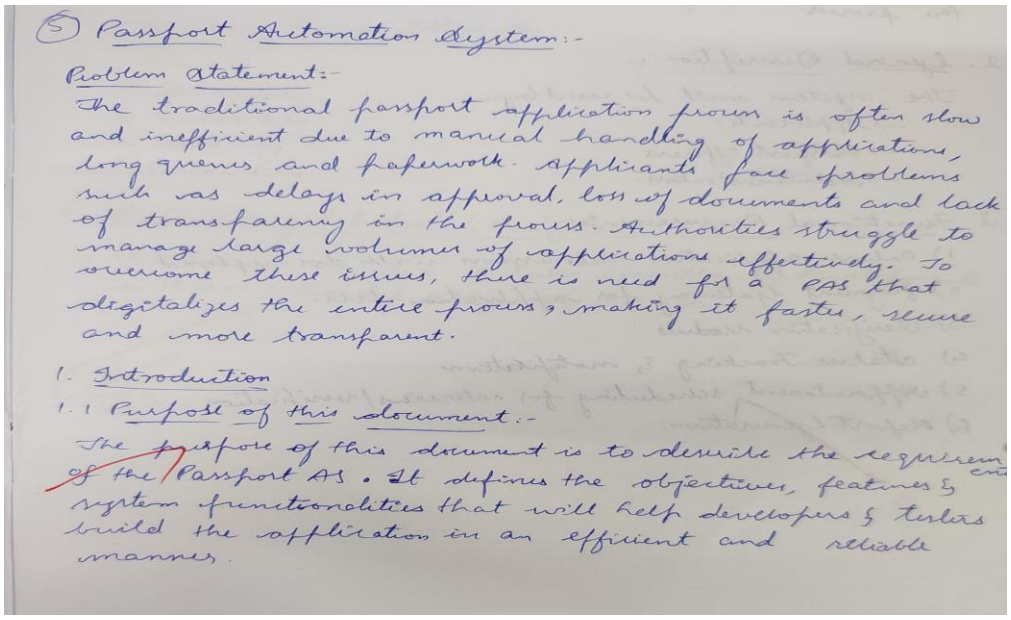
Activity Diagram(Advanced):

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

The final steps involve quality checks and stock updates. Once the product is delivered, the customer can report any quality issues. If a defect is found, the system initiates a refund process. Meanwhile, the supplier updates stock quantities to reflect the latest inventory status. This ensures real-time tracking and accountability. The flowchart's design emphasizes decision points and feedback loops, allowing for efficient handling of exceptions. By integrating supplier coordination, customer notifications, and inventory updates, the system maintains a seamless shopping experience while ensuring operational integrity and customer satisfaction.

5. Passport Automation System

SRS Document



5) Passport Automation System:-

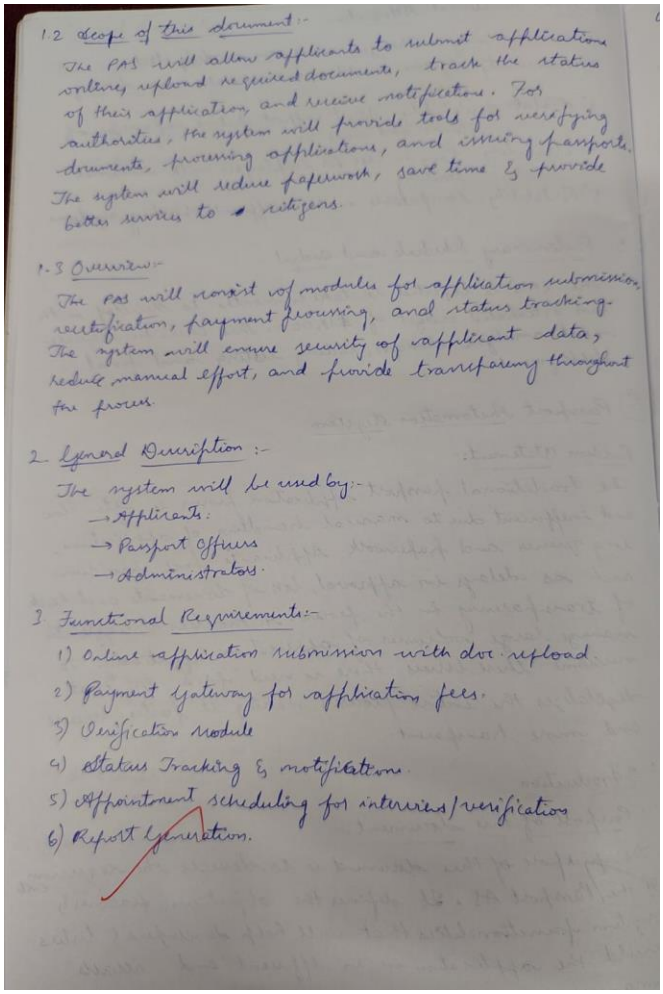
Problem Statement:-

The traditional passport application process is often slow and inefficient due to manual handling of applications, long queues and paperwork. Applicants face problems such as delays in approval, loss of documents and lack of transparency in the process. Authorities struggle to manage large volumes of applications effectively. To overcome these issues, there is need for a PAS that digitalizes the entire process, making it faster, secure and more transparent.

1. Introduction

1.1 Purpose of this document:-

The purpose of this document is to describe the requirements of the Passport AS. It defines the objectives, features & system functionalities that will help developers & testers build the application in an efficient and reliable manner.



1.2 Scope of this document:-

The PAS will allow applicants to submit applications online, upload required documents, track the status of their application and receive notifications. For authorities, the system will provide tools for verifying documents, processing applications, and issuing passports. The system will reduce paperwork, save time & provide better services to citizens.

1.3 Overview:-

The PAS will consist of modules for application submission, verification, payment processing, and status tracking. The system will ensure security of applicant data, reduce manual effort, and provide transparency throughout the process.

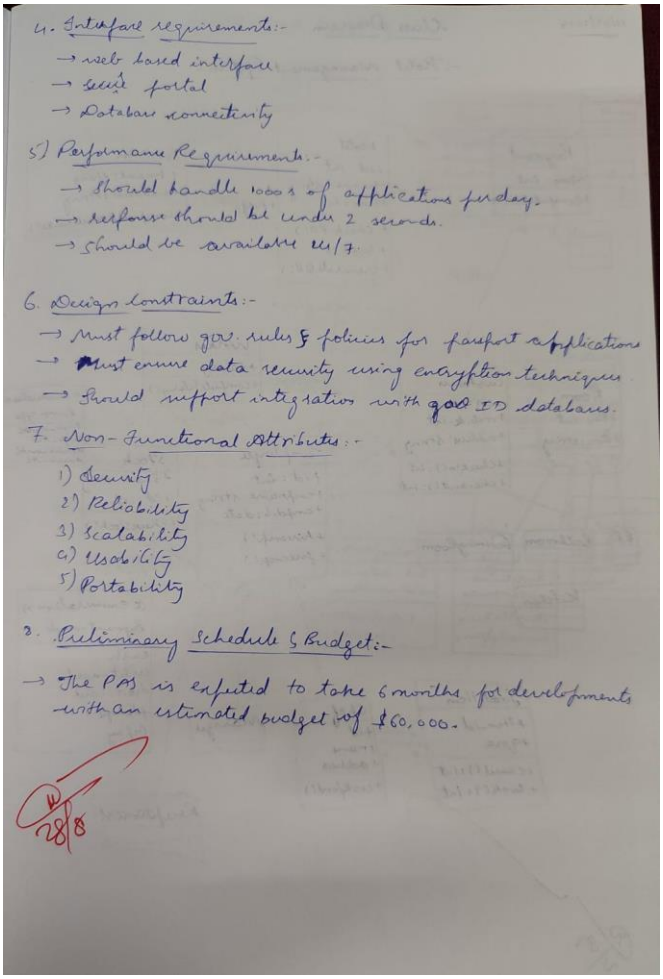
2. General Description:-

The system will be used by:-

→ Applicants.
→ Passport Officers
→ Administrators.

3. Functional Requirements:-

1) Online application submission with doc upload
2) Payment gateway for application fees.
3) Verification module
4) Status Tracking & notifications
5) Appointment scheduling for interview/verification
6) Report Generation.



4. Interface requirements:-

→ Web based interface
→ Secure portal
→ Database connectivity

5) Performance Requirements:-

→ Should handle 1000's of applications per day.
→ Response should be under 2 seconds.
→ Should be available 24/7.

6. Design Constraints:-

→ Must follow govt. rules & policies for passport applications
→ Must ensure data security using encryption techniques
→ Should support integration with govt ID databases.

7. Non-Functional Attributes:-

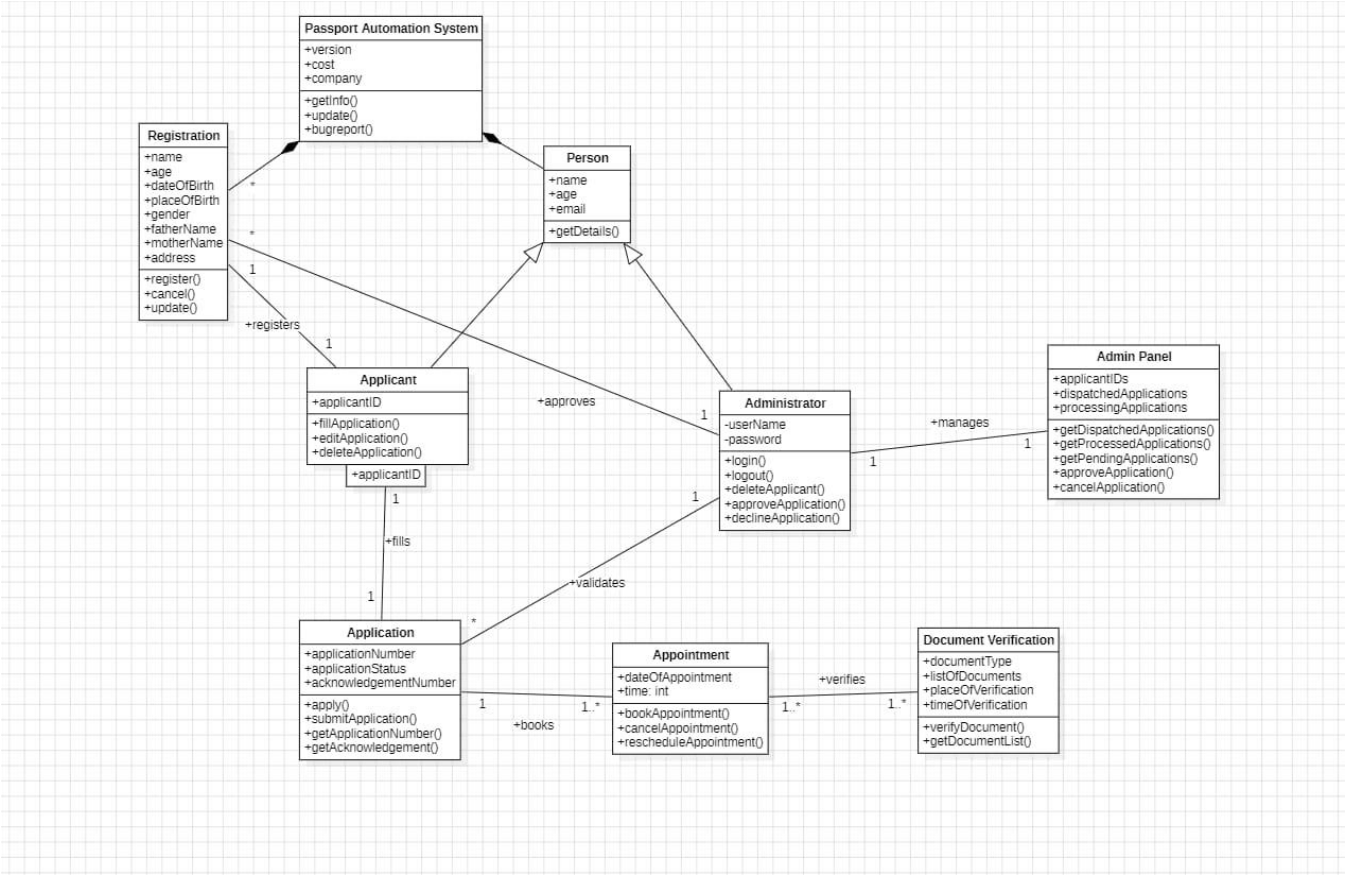
1) Security
2) Reliability
3) Scalability
4) Usability
5) Portability

8. Preliminary Schedule & Budget:-

→ The PAS is expected to take 6 months for developments with an estimated budget of $60,000.

28/8

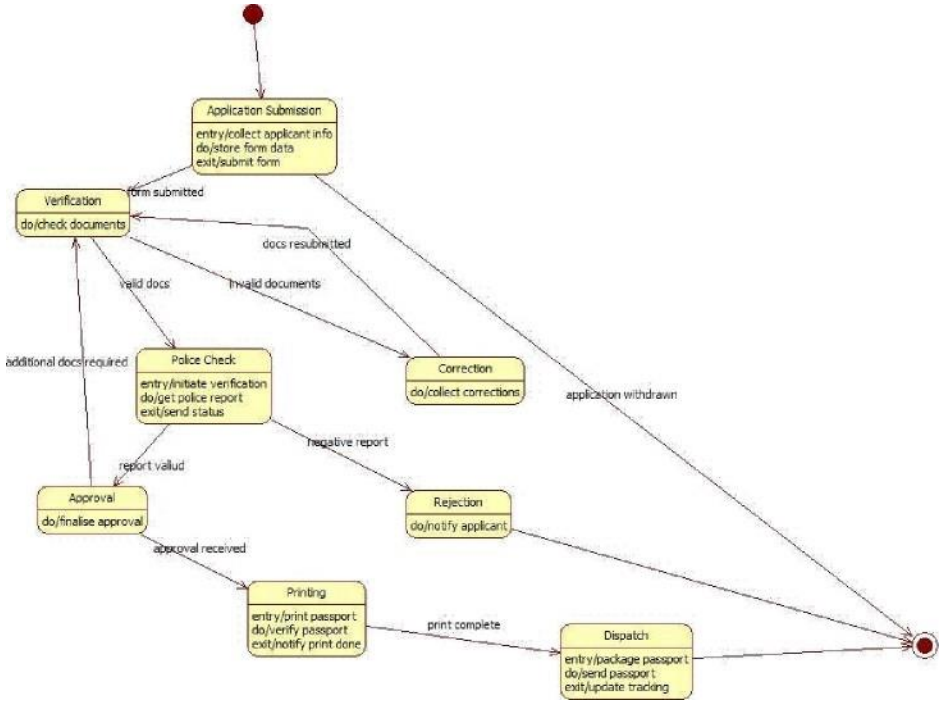
Class Diagram:



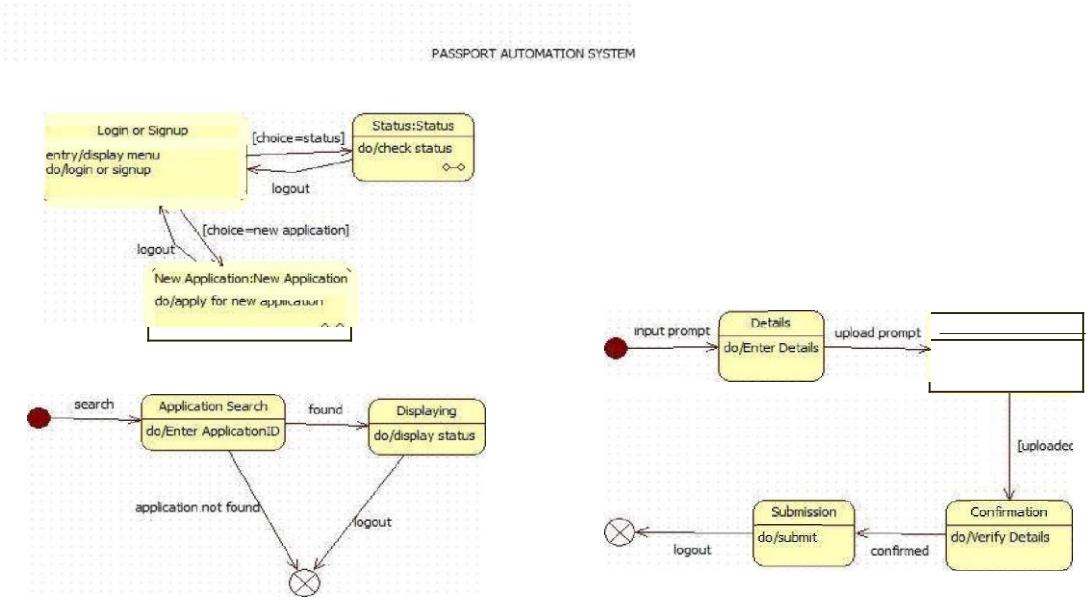
Description:

The diagram is a UML class diagram for a passport automation system, illustrating entities and their relationships involved in managing passport applications. Central to the system is the Passport Automation System class, which ties together Registration (for collecting user data), Applicant (for handling individual user activities like filling, editing, and deleting applications), and Administrator (for proctoring and approving/declining applications). Applications can be booked for appointments and must pass through document verification. The Admin Panel manages application states and provides admin functionalities for processing, dispatching, and approving or canceling applications. Key attributes and operations are defined for each class, showing how the process covers registration, application filling, approvals, scheduling appointments, and verifying documents in a systematic workflow.

State Diagram(Simple):



Advanced State Diagram:



Description:

State Diagram(Simple):

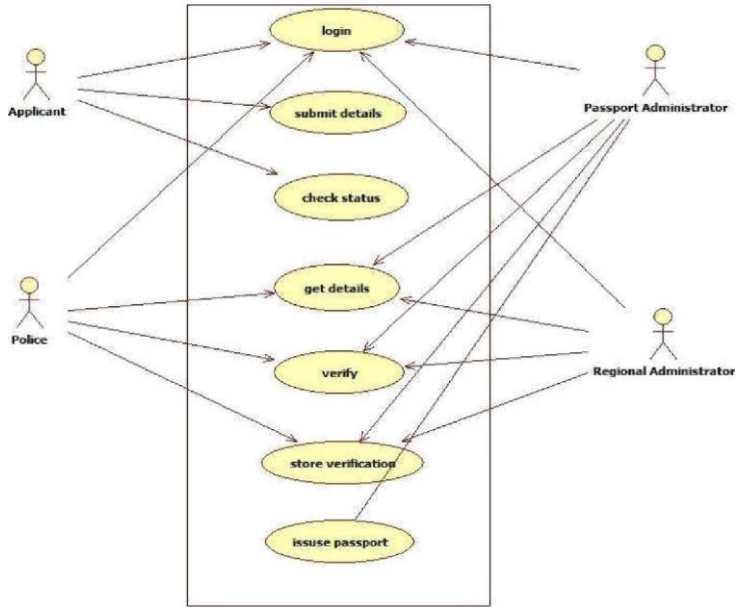
This activity diagram depicts the complete life-cycle of a passport application. It begins with the applicant submitting the online/offline form and documents. The application then undergoes Verification; if documents are invalid, corrections are sought or the application is withdrawn. Valid documents trigger Police Check; a negative report leads to Rejection, while a clear report moves to final Approval. Once approved, the passport proceeds to Printing and then Dispatch, completing the process. The diagram clearly shows the main success path (submission —+ verification —› police check —› approval —+ printing —• dispatch) along with alternate paths for invalid documents, corrections, negative police reports, and rejection.

State Diagram(Advanced):

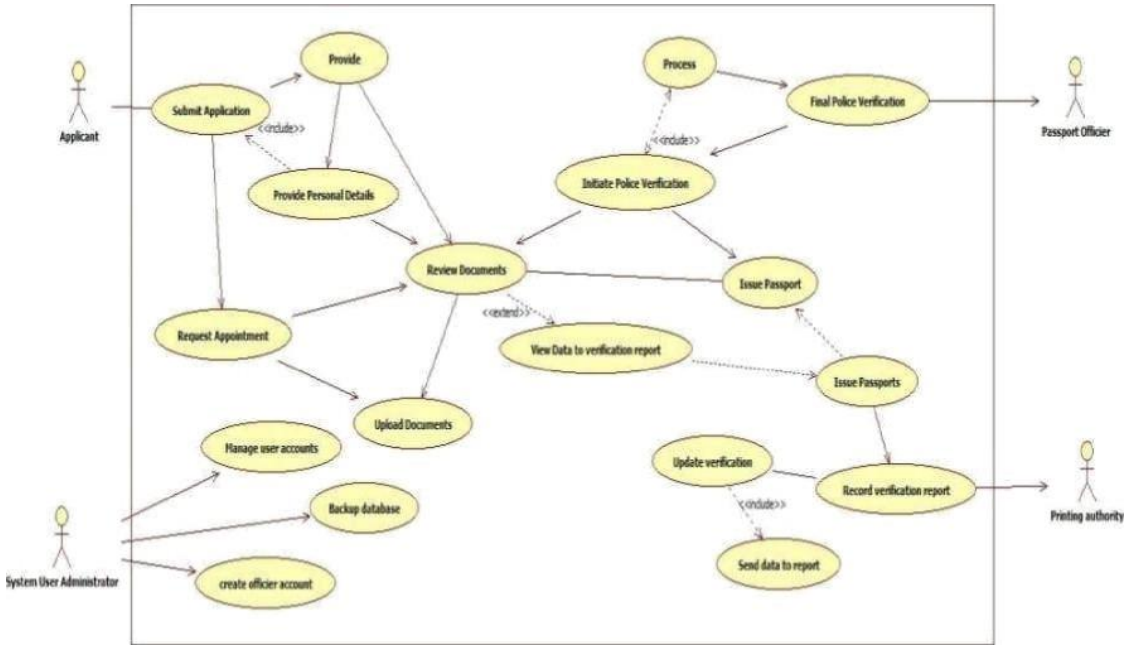
The activity diagram presents the main menu of a Passport Automation System, where users begin by logging in or signing up. After successful authentication, they are prompted to choose between two primary options: checking the status of an existing application or starting a new passport application.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

Selecting “New Application” initiates a dedicated subprocess where users first input personal details, then proceed to upload required documents. Successful upload triggers the “Submission” activity, followed by system-side confirmation and verification of the entered data. Only after verification is complete does the applicant receive final confirmation. Logout options are strategically placed throughout to ensure secure sessions. The clear sequential flow — details —+ documents —› submission —+ confirmation — minimizes errors, ensures all mandatory information is captured, and provides immediate feedback, making the online passport application process efficient, user-friendly, and compliant with official requirements.

Use Case(Simple):



Use Case(Advanced):



Description:

Use Case Diagram(Simple):

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

Use Case Diagram(Advanced):

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

The Admin manages the entire backend by handling user accounts, officer profiles, and generating reports. Key «include» relationships highlight that scheduling an appointment always triggers notification delivery and requires document verification, preventing incomplete applications from proceeding. The diagram clearly separates responsibilities: applicants handle submission and payments, officers manage verification and decision-making, police provide clearance, and admin oversees system governance. This structured, role-based design with included use cases ensures security, accountability, and efficient processing from registration to final decision, making the passport issuance process faster, paperless, and user-friendly.

Scenario

1: Main Success Scenario (Happy Path)

Applicant registers and logs into the passport portal

Fills application form correctly and uploads clear documents

Pays fees online successfully

Books and attends appointment at PSK on scheduled date

Officer verifies original documents and captures biometrics smoothly

Police verification completed with clear/no-adverse report

Regional authority approves the application

Passport printed and dispatched via Speed Post

Applicant tracks status online and receives passport within 25-30 days

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

Applicant completes registration, form filling and fee payment

Attends appointment; document & biometric verification successful

Police station visits applicant's address for enquiry

System detects pending criminal case / FIR

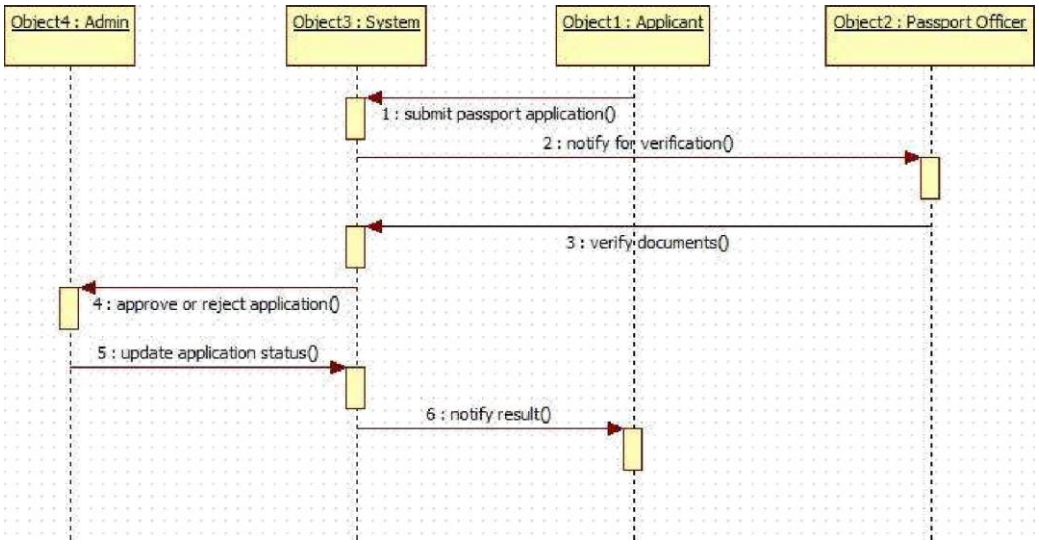
Police submit adverse verification report

Regional authority reviews and rejects the application

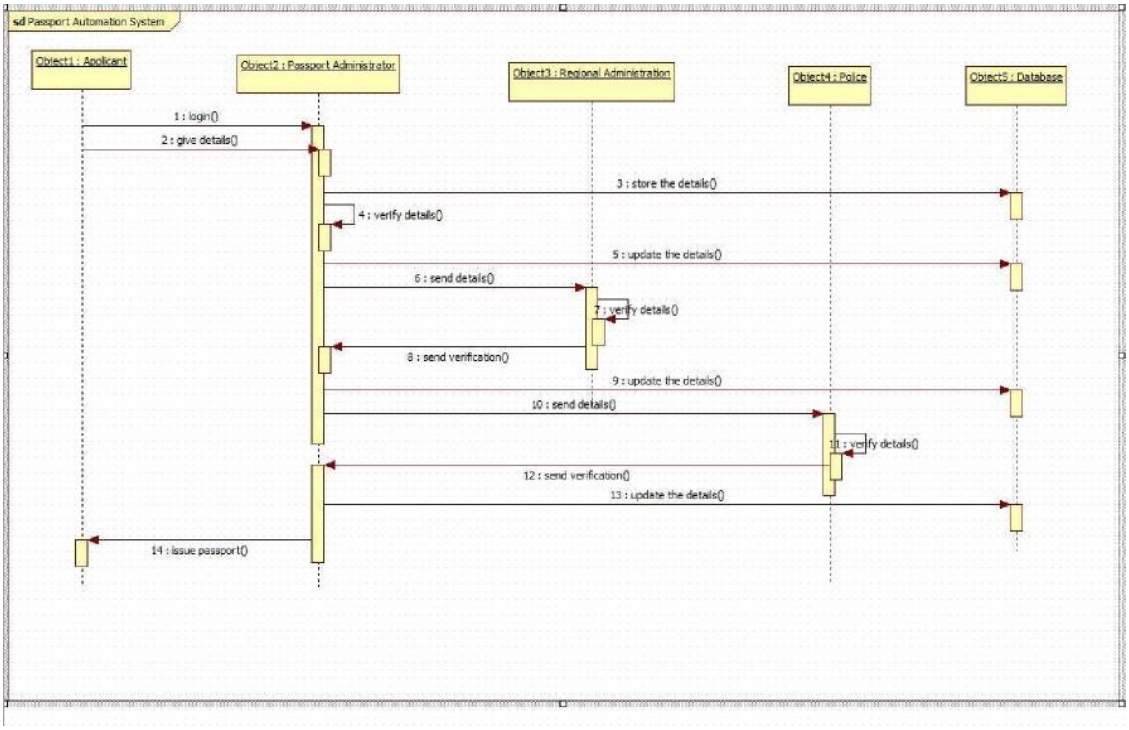
Applicant receives SMS + email notification sent with rejection reason

Status updated as “Rejected” on porta

Sequence Diagram(Simple):



Sequence Diagram(Advanced):



Sequence Diagram(Simple):

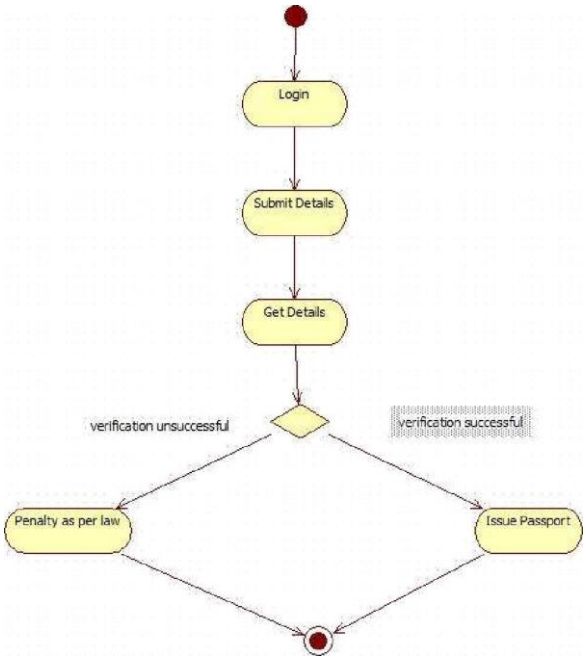
This sequence diagram illustrates the interaction flow in a Passport Automation System. The Applicant (Objectl) initiates by submitting the passport application to the System (Object3). The System immediately notifies the Passport Officer (Object2) for verification. The Officer verifies the documents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

Sequence Diagram(Advanced):

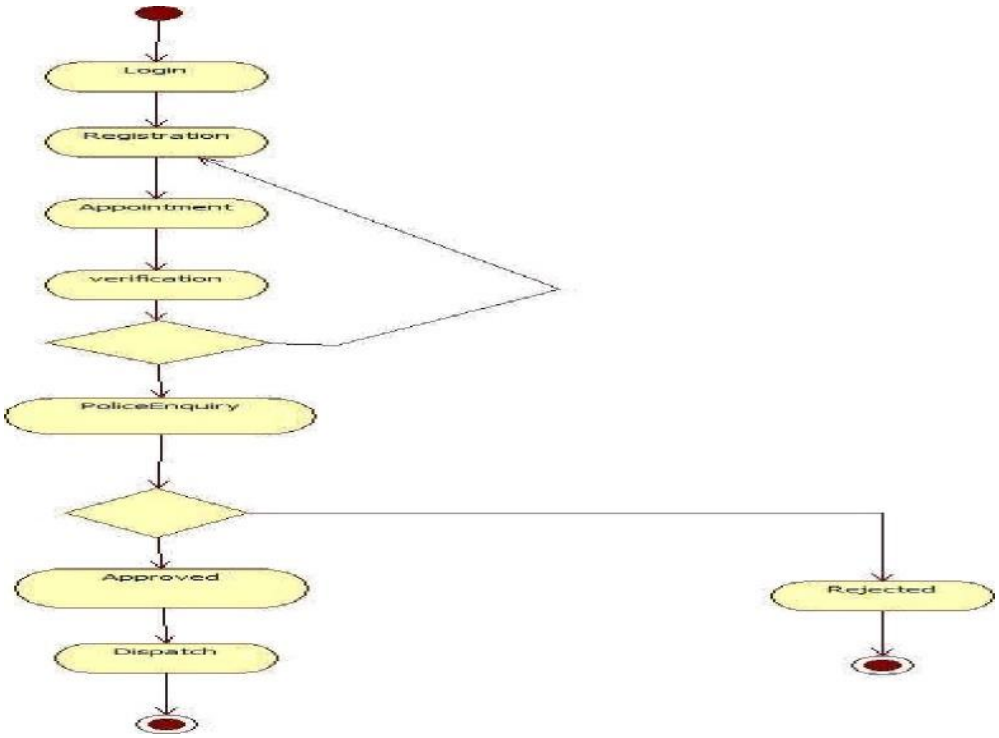
The sequence diagram depicts the complete passport application verification process. The Applicant first logs in and submits personal details to the Passport Administrator. The Administrator verifies the details, stores them in the Database, and forwards them to the Regional Administration. The Regional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

Once police verification is complete, the Passport Administrator receives confirmation, updates the Database, and the Regional Administration issues the final approval. The Database is updated with the final status, and the Passport Administrator issues the passport to the Applicant. Throughout the flow, the Database serves as the central repository, updated at every stage (initial submission, regional processing, police verification, and final issuance). This structured, step-by- step interaction among Applicant, Passport Administrator, Regional Administration, Police, and Database ensures transparency, accountability, and secure passport issuance with proper background checks and official validations at each level.

Activity Diagram(Simple):



Activity Diagram(Advanced):



Activity Diagram(Simple):

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

Activity Diagram(Advanced):

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

After police enquiry, another decision node determines the outcome: if clearance is received, the application is Approved and proceeds to Dispatch of the passport; if any issue is found, the application is Rejected and the process ends. The diagram clearly shows two termination points — one for successful dispatch and one for rejection. By incorporating appointment scheduling, multi-level verification (document + police), and explicit decision points, the flow ensures security, transparency, and compliance with legal procedures, preventing issuance to ineligible persons while providing a smooth path for genuine applicants from registration to final delivery.