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

Lab Record

Object Oriented Analysis and Design

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:

Navanidhi D J

1BM23CS204

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

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

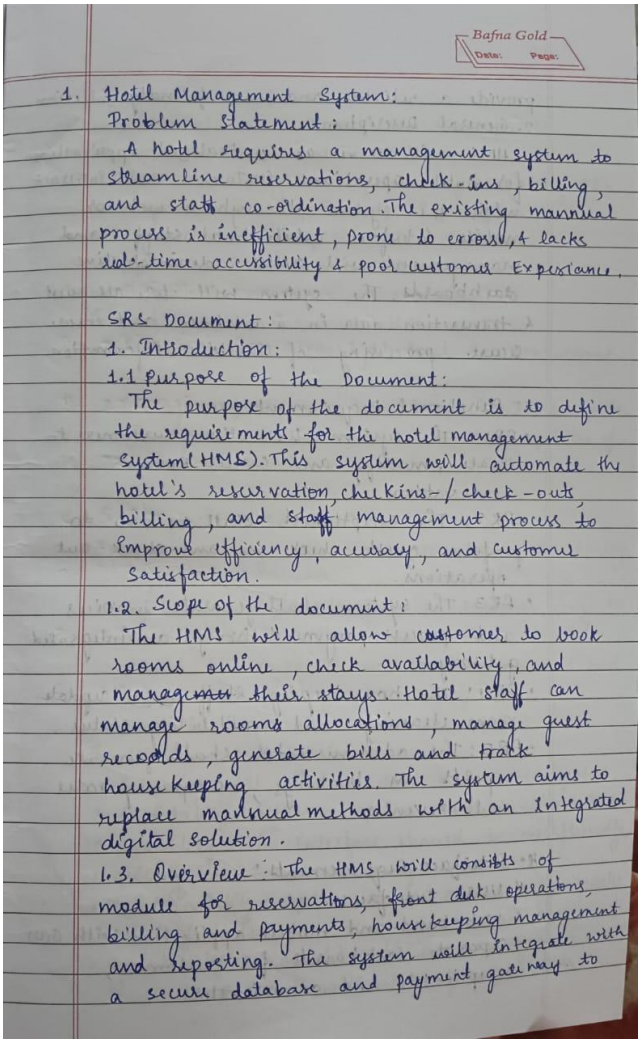
This is to certify that the Object-Oriented Analysis and Design(23CS5PCOOM) laboratory has been carried out by Navanidhi D J (1BM23CS204) during the 5th Semester August-December 2025

Signature of the Faculty Incharge:

NAME OF THE FACULTY:
Sunayana S
Assistant Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

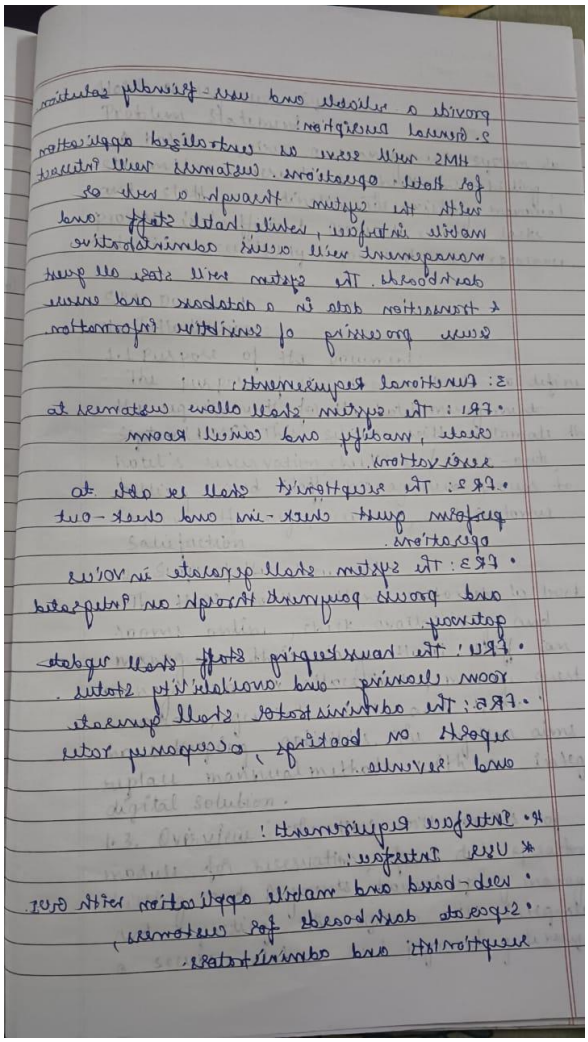
Table of Contents

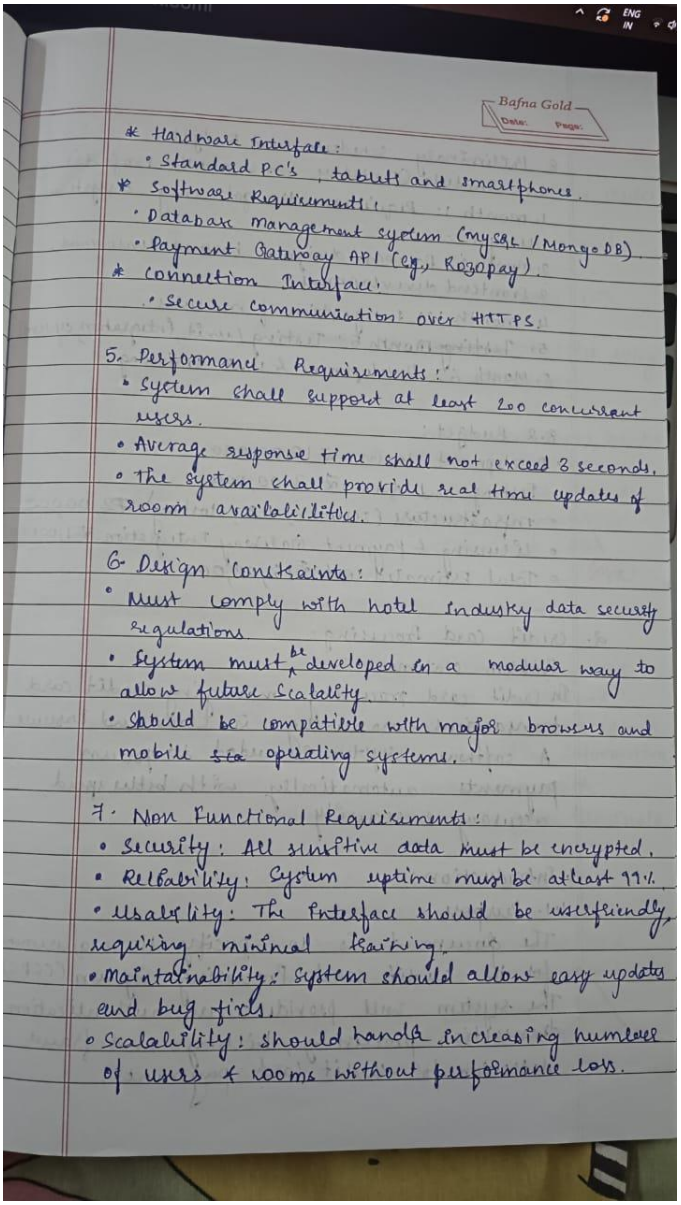
1. Hotel Management System



1. Hotel Management System:
Problem Statement:
A hotel requires a management system to streamline reservations, check-ins, billing, and staff co-ordination. The existing manual process is inefficient, prone to errors, & lacks real-time accessibility & poor customer Experience.

SRS Document:
1. Introduction:
1.1 Purpose of the Document:
The purpose of the document is to define the requirements for the hotel management system (HMS). This system will automate the hotel's reservation, checkins-/check-outs, billing, and staff management process to improve efficiency, accuracy, and customer satisfaction.
1.2. Scope of the document:
The HMS will allow customer to book rooms online, check availability, and manage their stays. Hotel staff can manage rooms allocations, manage guest records, generate bills and track housekeeping activities. The system aims to replace manual methods with an integrated digital solution.
1.3. Overview: The HMS will consists of module for reservations, front desk operations, billing and payments, housekeeping management and reporting. The system will integrate with a secure database and payment gateway to





Bafna Gold
Date: Page:

* Hardware Interface:
  - Standard PC's, tablets and smartphones.
* Software Requirements:
  - Database management system (mySQL / MongoDB).
  - Payment Gateway API (eg, Razopay).
* Connection Interface:
  - Secure communication over HTTPS.

5. Performance Requirements:
  - System shall support at least 200 concurrent users.
  - Average response time shall not exceed 3 seconds.
  - The system shall provide real time updates of room availabilities.

6. Design Constraints:
  - Must comply with hotel industry data security regulations.
  - System must be developed in a modular way to allow future scalability.
  - Should be compatible with major browsers and mobile ~~os~~ operating systems.

7. Non Functional Requirements:
  - Security: All sensitive data must be encrypted.
  - Reliability: System uptime must be at least 99%.
  - Usability: The interface should be user-friendly, requiring minimal training.
  - Maintainability: System should allow easy updates and bug fixes.
  - Scalability: should handle increasing numbers of users & rooms without performance loss.

1.3 Class Diagram

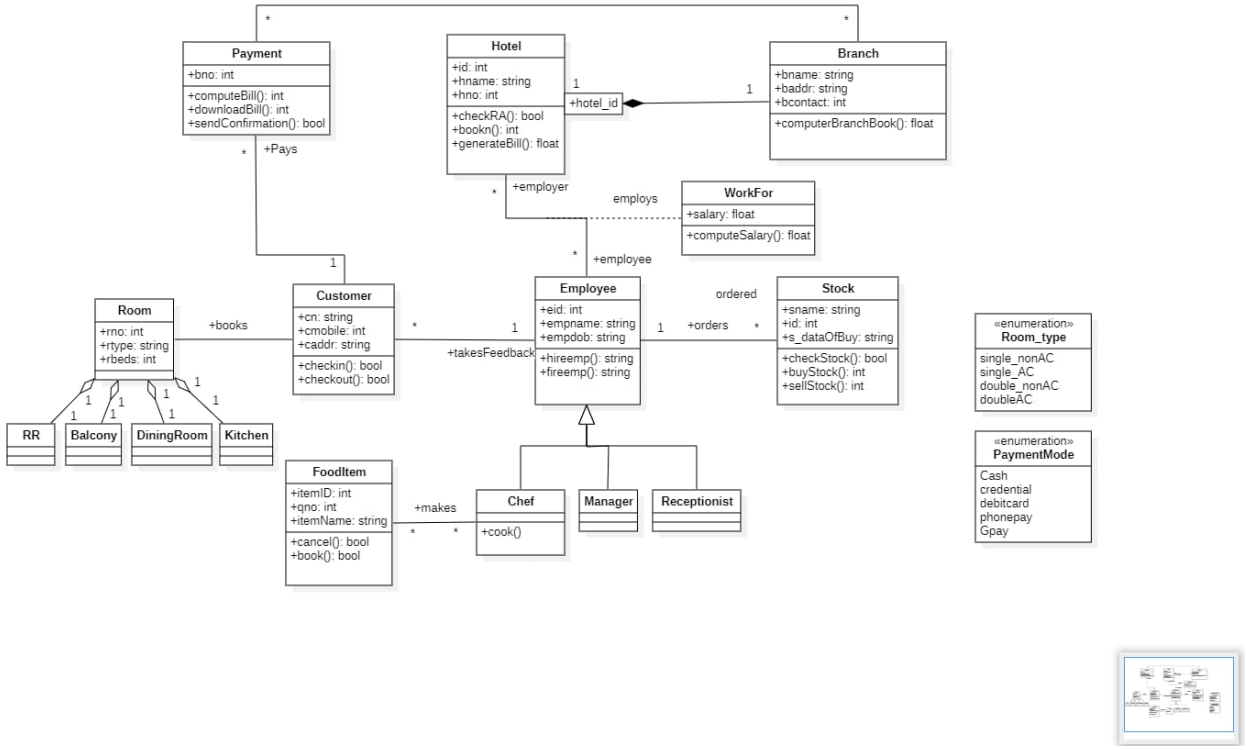


Fig 1.1 Class Diagram for Hotel Management System

The class diagram illustrates the structural design of the hotel management system by showing all major classes, their attributes, methods, and relationships. Key classes such as Hotel, Branch, Customer, Employee, Room, Payment, Stock, and FoodItems form the foundation of the system. It also depicts inheritance, where Chef, Manager, and Receptionist extend the Employee class, reflecting role-specific responsibilities. Relationships like a Customer booking a Room, a Payment settling booking charges, and a Hotel employing Employees are clearly shown using associations and multiplicities. Overall, the diagram provides a detailed blueprint of the system's components and how they interact to support various hotel operations.

1.4 State Diagram

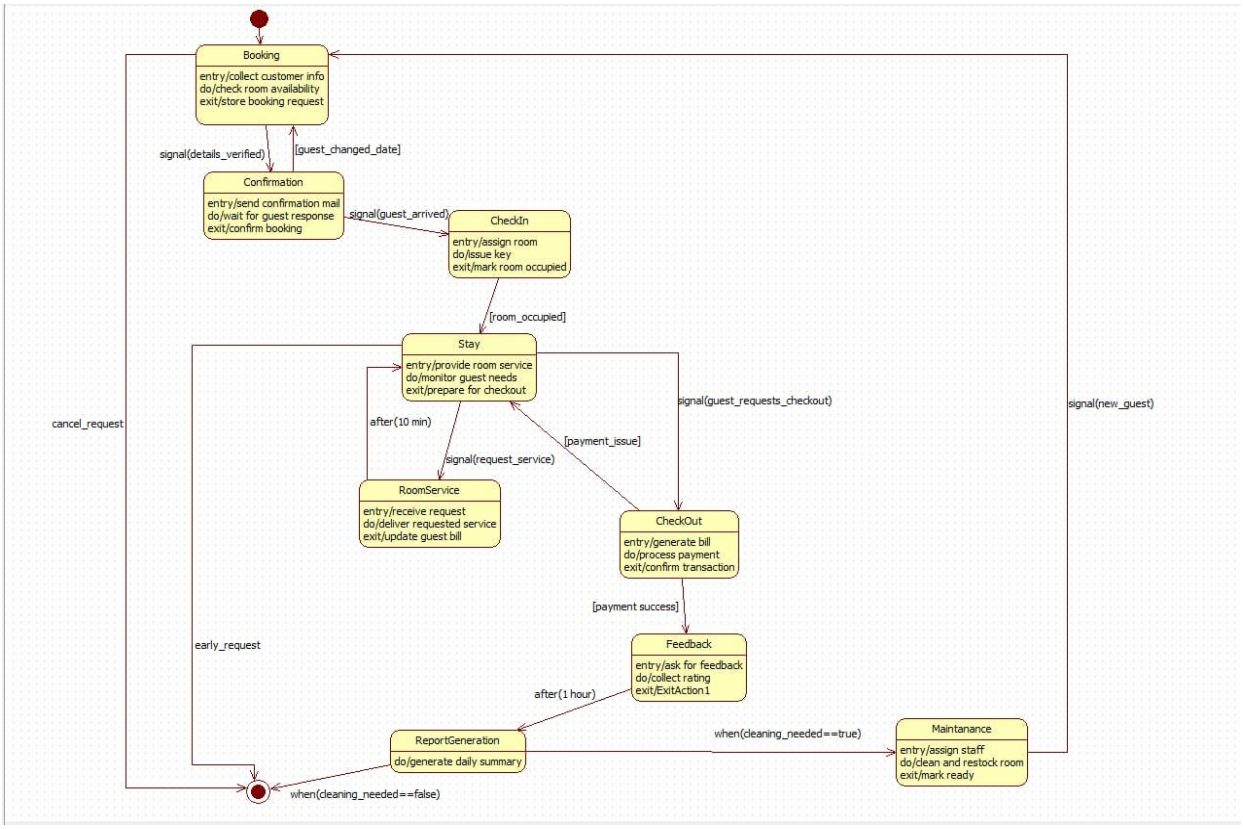


Fig 1.2 State Diagram for Hotel Management System (1)

This state machine diagram represents the dynamic flow of the hotel booking lifecycle from the moment a customer initiates a booking until checkout and post-stay feedback. The process begins at the Booking state where customer details are collected, followed by Confirmation, Check-in, Stay, and Room Service states. Transitions occur based on events such as guest arrival, payment success, or maintenance requirements. The diagram also includes system-driven states like Report Generation and Maintenance to ensure operational efficiency. Overall, it highlights how the booking system responds to different events and moves through various stages of a guest's hotel experience.

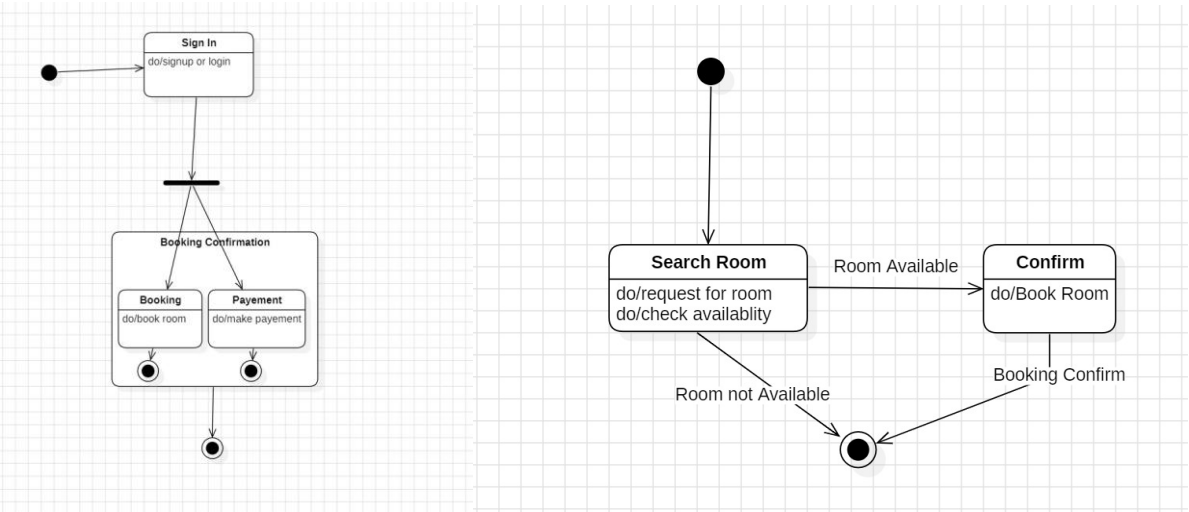


Fig 1.3 State Diagram for Hotel Management System(2)

The room search diagram shows a simple decision-based flow for checking room availability. The process starts with the Search Room state, where the system processes the customer's request and checks availability. If a room is available, the flow transitions to the Confirm state where booking can be completed. If no rooms are available, the process terminates at the final state. This diagram provides a clear depiction of the basic decision-making logic involved in room searching and reservation.

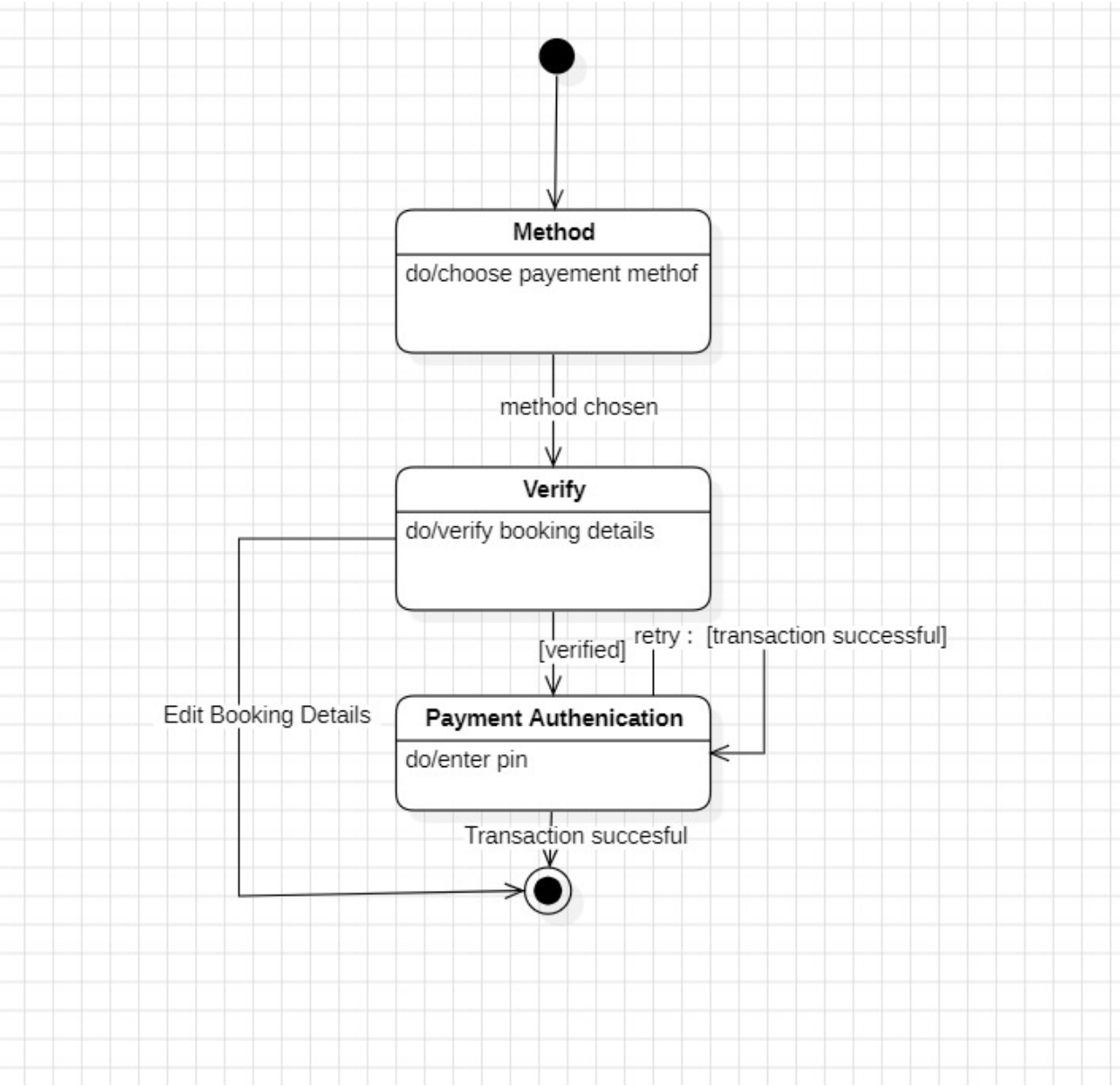


Fig 1.4 State Diagram for Hotel Management System(3)

The payment state machine diagram outlines the sequential steps involved in completing a hotel payment transaction. It begins with the Method state, where the customer chooses a payment method. It then moves to the Verify state, which checks booking and payment details. If everything is valid, the flow proceeds to Payment Authentication, where the user provides security credentials such as a PIN or OTP. Upon successful authentication, the process ends at the final state. If verification fails, the user may edit booking details or retry the transaction. This diagram effectively demonstrates the dynamic process and decision paths in payment handling.

1.5 Use Case Diagram

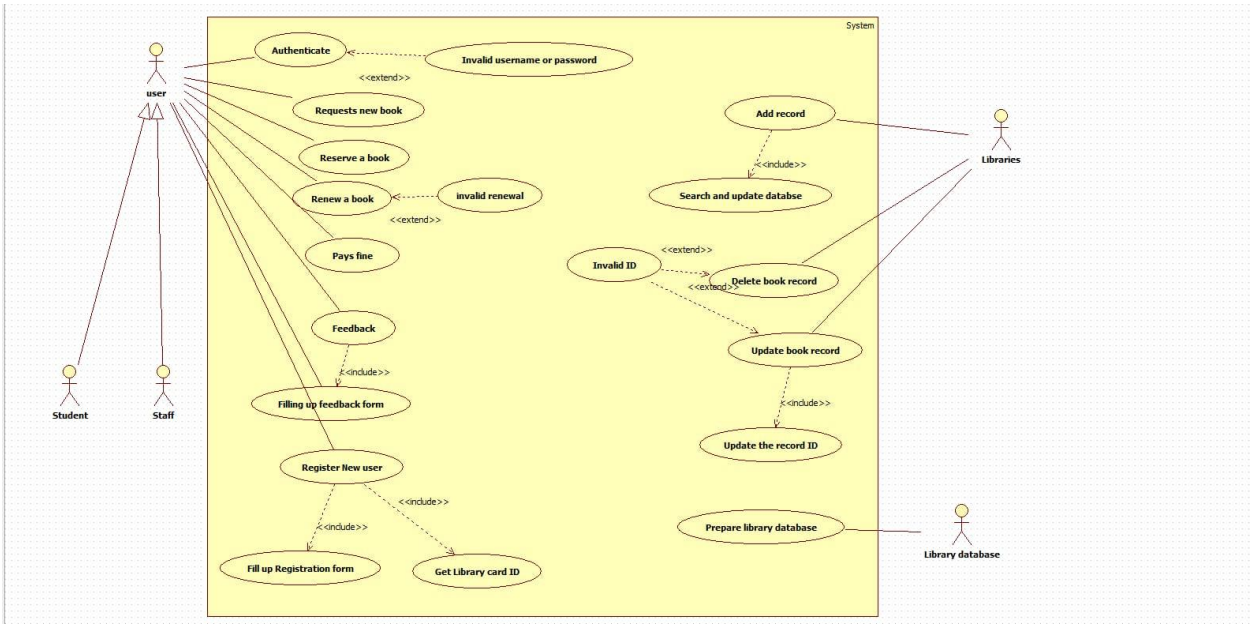


Fig 1.5 Use Case Diagram for Hotel Management System

The use case diagram represents the functional interactions between different users and the library management system. The main actors—Students, Staff, Librarians, and the Library Database—engage with the system through activities such as authentication, searching for books, reserving books, paying fines, giving feedback, and managing book records. Librarians are responsible for adding, updating, and deleting records, while the system includes exception flows like invalid credentials or invalid IDs. Overall, the diagram captures the complete set of services provided by the library system and the roles played by different users.

1.6 Sequence Diagram

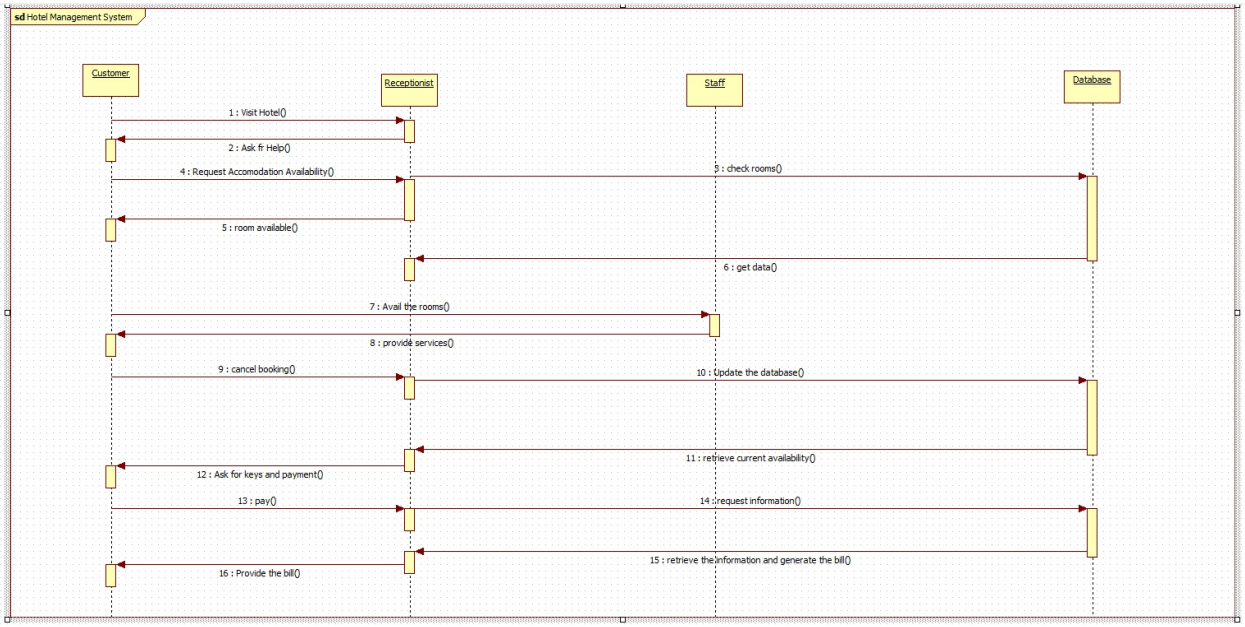


Fig 1.6 Sequence Diagram for Hotel Management System

The participants in this diagram are the Customer, Receptionist, Staff, and the Database. The sequence starts when a customer visits the hotel and requests assistance from the receptionist to check for room availability. The receptionist then communicates with the staff to check the rooms, which in turn involves querying the database for real-time information. The diagram then illustrates several possible scenarios. For a successful booking, the customer avails the room, and the staff provides services. It also shows the process for a customer cancelling a booking, which involves the receptionist instructing the staff to update the database accordingly. The final part of the sequence details the payment process, where the customer asks for the bill, the receptionist retrieves the necessary information from the database, the customer makes the payment, and the receptionist provides the final generated bill.

1.7 Activity Diagram

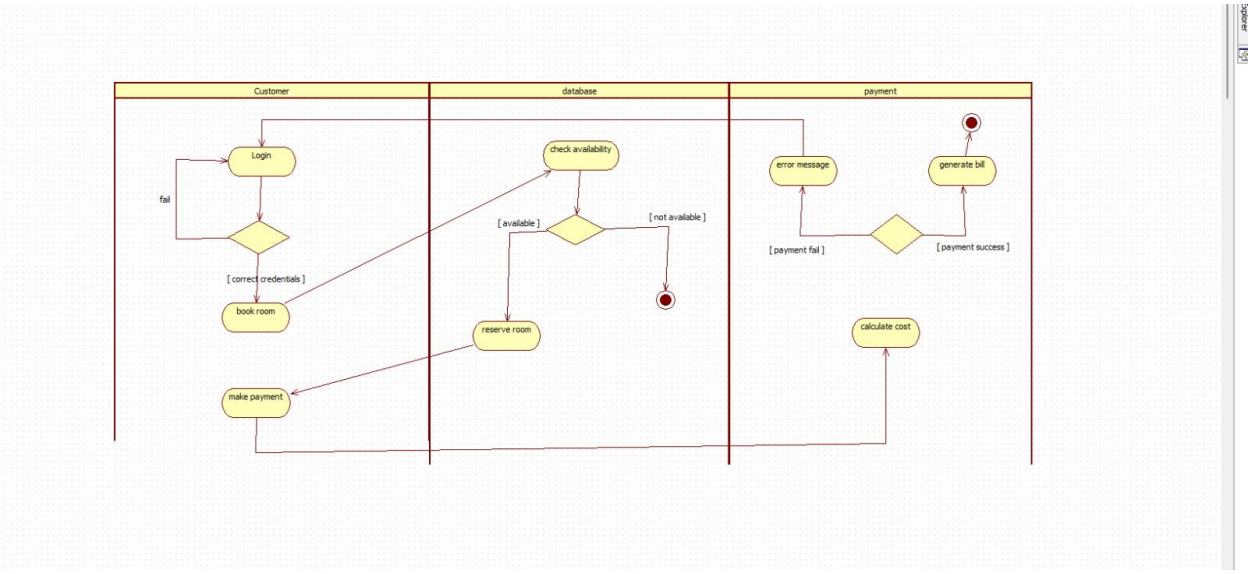
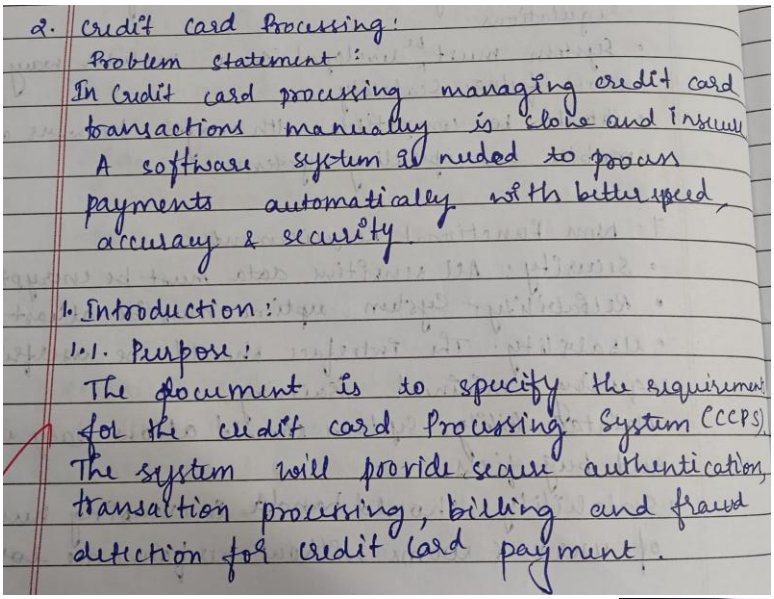


Fig 1.7 Activity Diagram for Hotel Management System

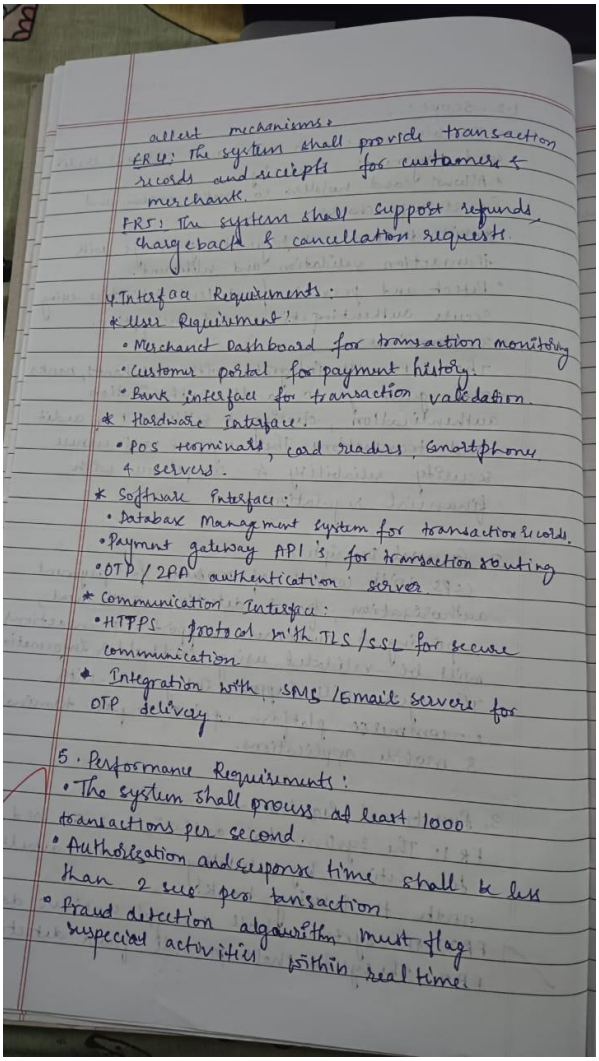
The diagram is divided into three sections, called swimlanes, representing the main actors or systems involved: the Customer, the Database, and the Payment system. The process begins with the customer's attempt to log in. If the login is successful, the customer can proceed to book a room. This action prompts the system to check the database for room availability. If a room is available, it is reserved for the customer, who is then directed to make a payment. If no room is available, the process ends. The payment step involves calculating the total cost. If the payment transaction is successful, a bill is generated, and the process is complete. However, if the payment fails, an error message is displayed, and the process terminates.

2. Credit Card Processing:



2. Credit Card Processing:
Problem statement:
In Credit card processing managing credit card transactions manually is slow and insecure. A software system is needed to process payments automatically with better speed, accuracy & security.

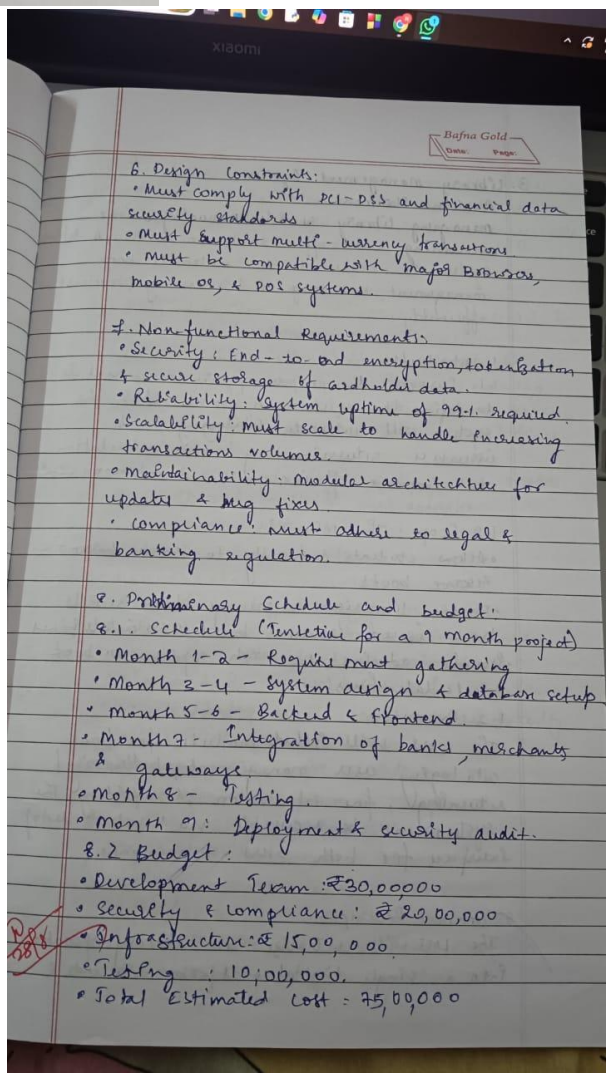
1. Introduction:
1.1. Purpose:
The document is to specify the requirements for the credit card Processing System (CCPS). The system will provide secure authentication, transaction processing, billing and fraud detection for credit card payment.



alert mechanisms.
FR4: The system shall provide transaction records and receipts for customers & merchants.
FR5: The system shall support refunds, chargeback & cancellation requests.

4. Interface Requirements:
* User Requirement:
• Merchant Dashboard for transaction monitoring
• Customer portal for payment history.
• Bank interface for transaction validation.
* Hardware Interface:
• POS terminals, card readers, Smartphone & servers.
* Software Interface:
• Database Management system for transaction records.
• Payment gateway API's for transaction routing
• OTP / 2FA authentication server
* Communication Interface:
• HTTPS protocol with TLS/SSL for secure communication
• Integration with SMS/Email servers for OTP delivery.

5. Performance Requirements:
• The system shall process at least 1000 transactions per second.
• Authorization and response time shall be less than 2 sec per transaction.
• Fraud detection algorithm must flag suspected activities within real time.



Bafna Gold

6. Design Constraints:
• Must comply with PCI-DSS and financial data security standards.
• Must support multi-currency transactions
• Must be compatible with major Browsers, mobile os, & POS systems.

7. Non-functional Requirements:
• Security: End-to-end encryption, tokenization & secure storage of cardholder data.
• Reliability: System uptime of 99.9% required.
• Scalability: Must scale to handle increasing transactions volumes.
• Maintainability: Modular architecture for updates & bug fixes
• Compliance: must adhere to legal & banking regulation.

8. Preliminary Schedule and budget:
8.1. Schedule (Tentative for a 9 month project)
• Month 1-2 - Requirement gathering
• Month 3-4 - System design & database setup
• Month 5-6 - Backend & Frontend.
• Month 7 - Integration of banks, merchants & gateways.
• Month 8 - Testing.
• Month 9: Deployment & security audit.
8.2 Budget:
• Development Team: ₹30,00000
• Security & compliance: ₹20,00,000
• Infrastructure: ₹15,00,000
• Testing: 10,00,000.
• Total Estimated cost: 75,00,000

2.3 Class Diagram

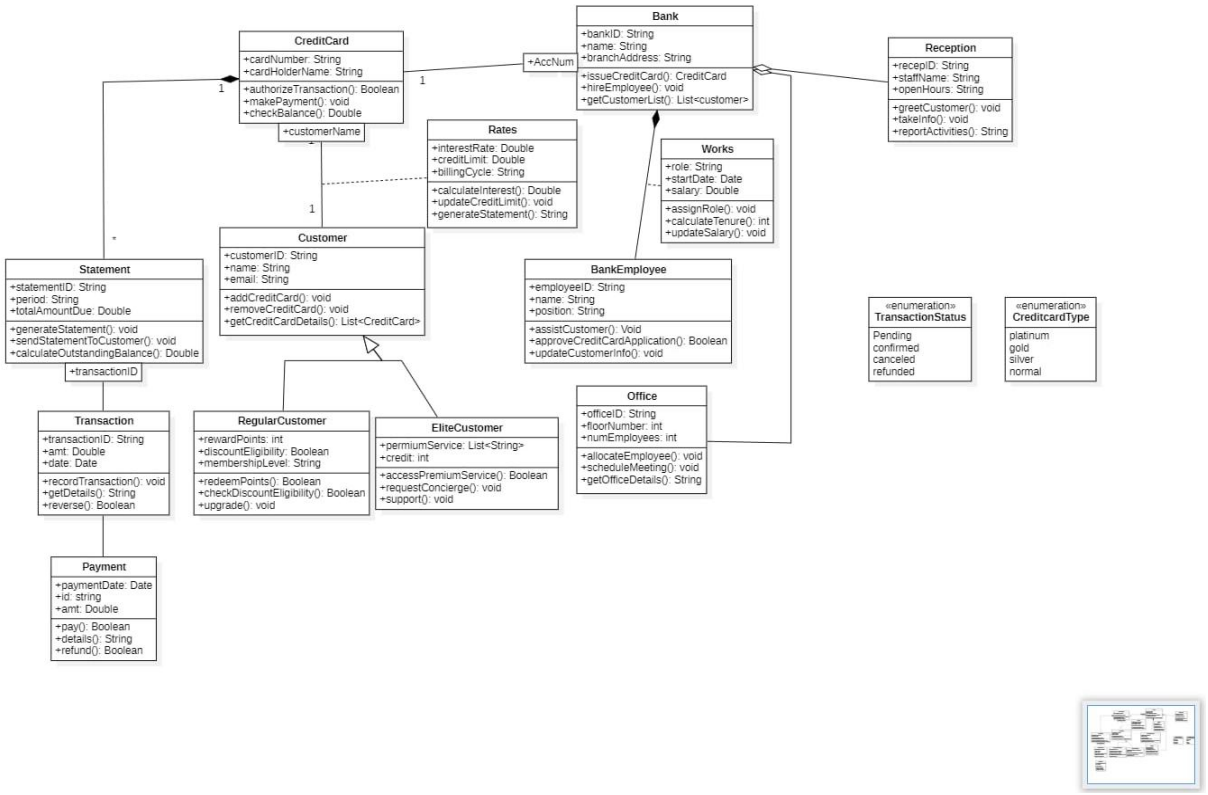


Fig 2.3 Class Diagram for Credit Card Processing

This is a Class Diagram that outlines the static structure of a banking system, focusing on credit card services. It defines various classes such as Bank, Customer, CreditCard, BankEmployee, and Transaction. The Bank class has attributes like name and address and can perform actions such as issuing credit cards and managing employees. The Customer class, which has a relationship with the Bank, can have one or more CreditCards. Customers are further specialized into RegularCustomer and EliteCustomer, each with unique attributes and behaviors. The CreditCard class contains details like card number and cardholder name and is associated with Statement and Transaction classes. This diagram effectively maps out the entities within the system, their properties, and how they are related to one another.

2.4 State Diagram

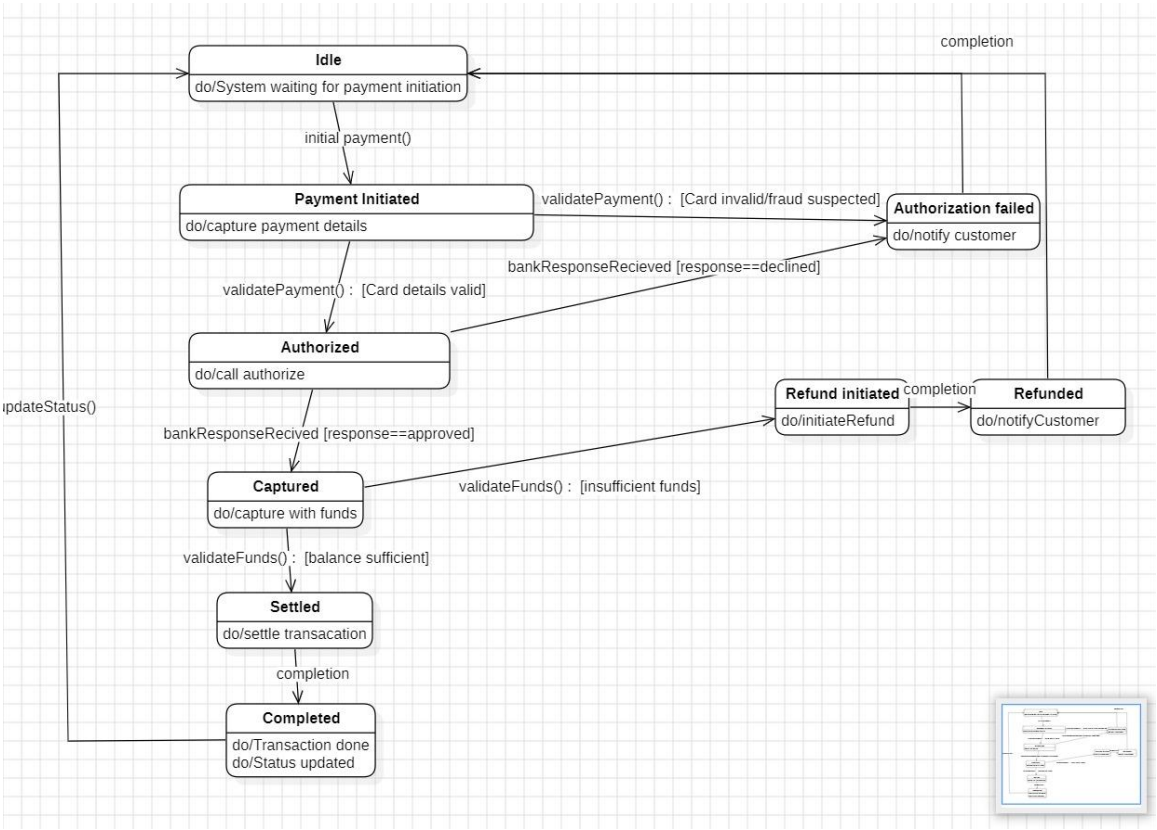


Fig 2.2 State Diagram for Credit Card Processing(1)

This diagram is a State Machine Diagram that illustrates the lifecycle of a payment transaction. The process begins in the Idle state, waiting for a payment to be initiated. Upon initiation, it transitions to the Payment Initiated state, where payment details are captured. The system then attempts to validate the payment. If the card is invalid or fraud is suspected, the state changes to Authorization failed, and the customer is notified. If the card details are valid, it moves to the Authorized state. Based on the bank's response, the transaction is either Captured (if approved) or moves to Authorization failed (if declined). From the Captured state, if funds are sufficient, the transaction becomes Settled and then Completed. If funds are insufficient, it can transition to a failure state. The diagram also shows a path for refunds, moving from a completed or settled state to Refund initiated and finally Refunded.

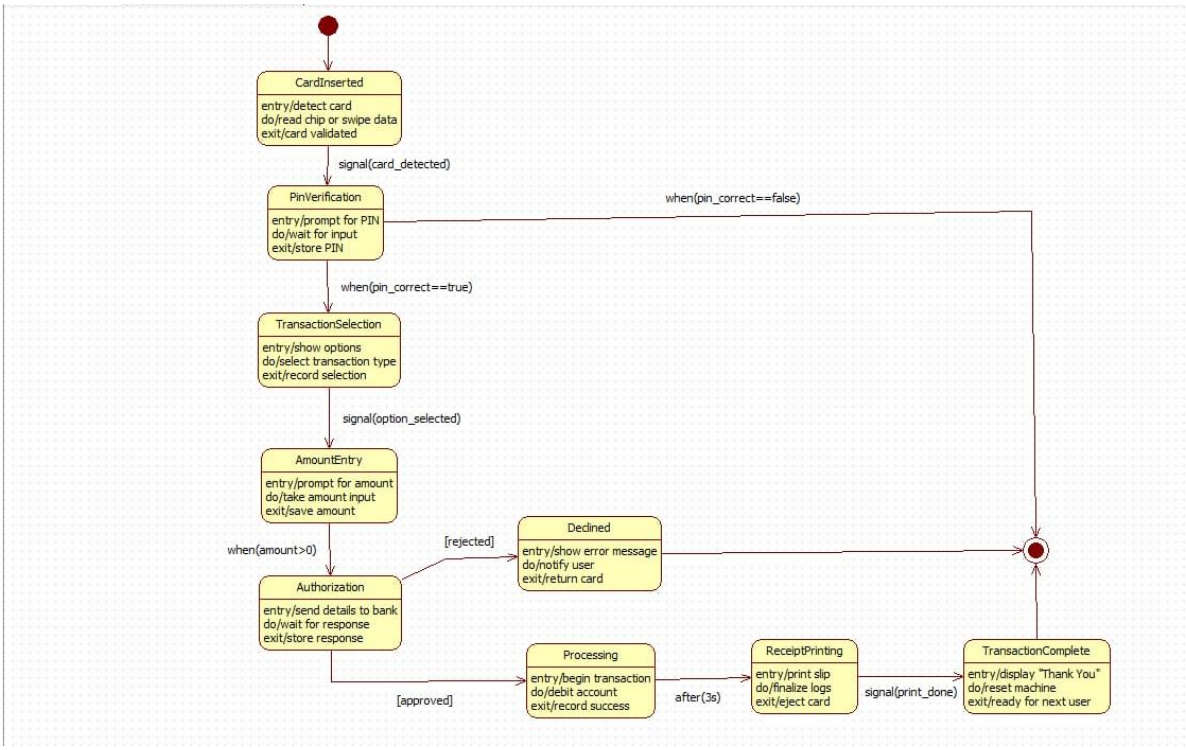


Fig 2.3 State Diagram for Credit Card Processing(2)

This State Machine Diagram details the process of an ATM or Point-of-Sale (POS) transaction. The flow starts when a CardInserted is detected. The system then transitions to PinVerification, prompting the user for a PIN. If the PIN is correct, it moves to TransactionSelection, allowing the user to choose an action like withdrawal. After the user enters an AmountEntry, the system proceeds to Authorization by sending the details to the bank. If the transaction is approved, it goes into Processing, followed by ReceiptPrinting, and finally ends in the TransactionComplete state. If the authorization is rejected at any point (e.g., incorrect PIN or bank denial), the state shifts to Declined, an error message is shown, and the process terminates.

2.5 Use Case Diagram

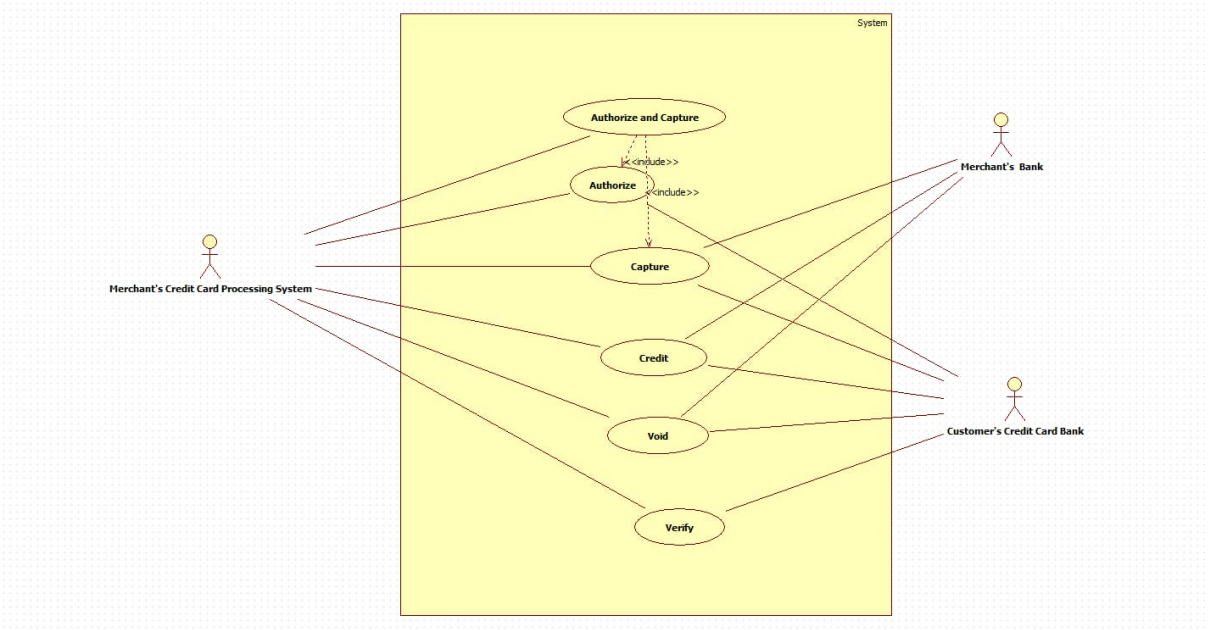


Fig 2.4 Use Case Diagram for Credit Card Processing

This is a Use Case Diagram for a Merchant's Credit Card Processing System. It identifies the primary actors interacting with the system: the Merchant's Credit Card Processing System itself, the Merchant's Bank, and the Customer's Credit Card Bank. The central system provides several key functions, or use cases, including Authorize, Capture, Credit (for refunds), Void (to cancel transactions), and Verify. The Authorize and Capture use cases are combined into a larger Authorize and Capture use case, indicating that a payment is typically authorized first and then captured. The diagram shows which actors interact with each use case; for example, both the merchant's system and the customer's bank are involved in authorization and verification processes.

2.6 Sequence Diagram

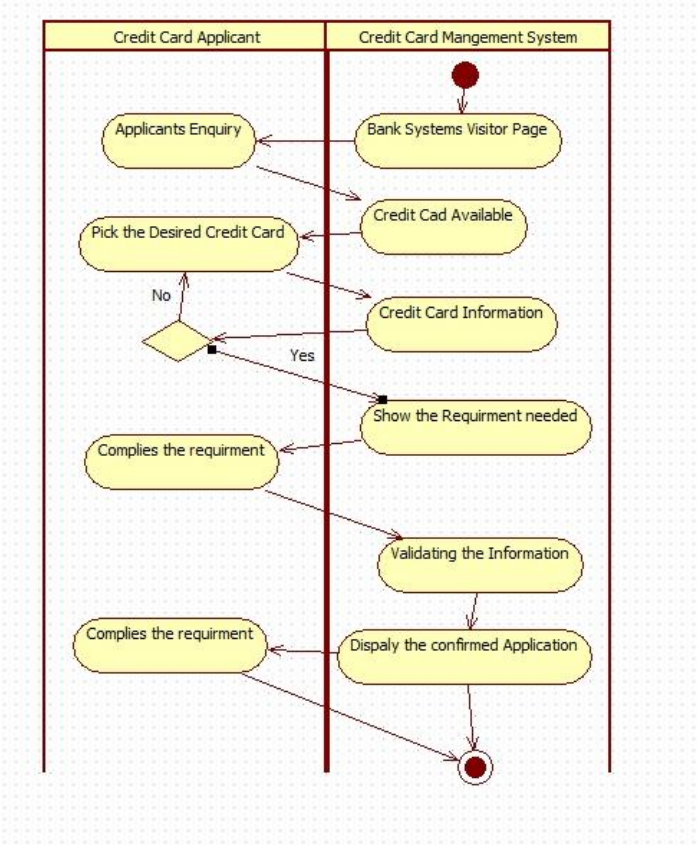


Fig 2.5 Sequence Diagram for Credit Card Processing

This Sequence Diagram illustrates the interactions between multiple components during an online credit card transaction. The participants include the Card Holder, Merchant, App Payment (payment gateway), Bank Office, Database, and Payment Provider. The sequence starts when the Card Holder makes a Purchase from the Merchant. The Merchant requests payment from the App Payment system, which returns a payment URL to the cardholder. The cardholder submits their information, which the App Payment system processes by masking the Primary Account Number (PAN) and verifying it with the Bank Office and Database. Once the information is verified, the App Payment system sends the credit card information to the Payment Provider for processing. The final payment result is relayed back through the App Payment system to both the Merchant and the Card Holder, completing the transaction sequence.

2.7 Activity Diagram

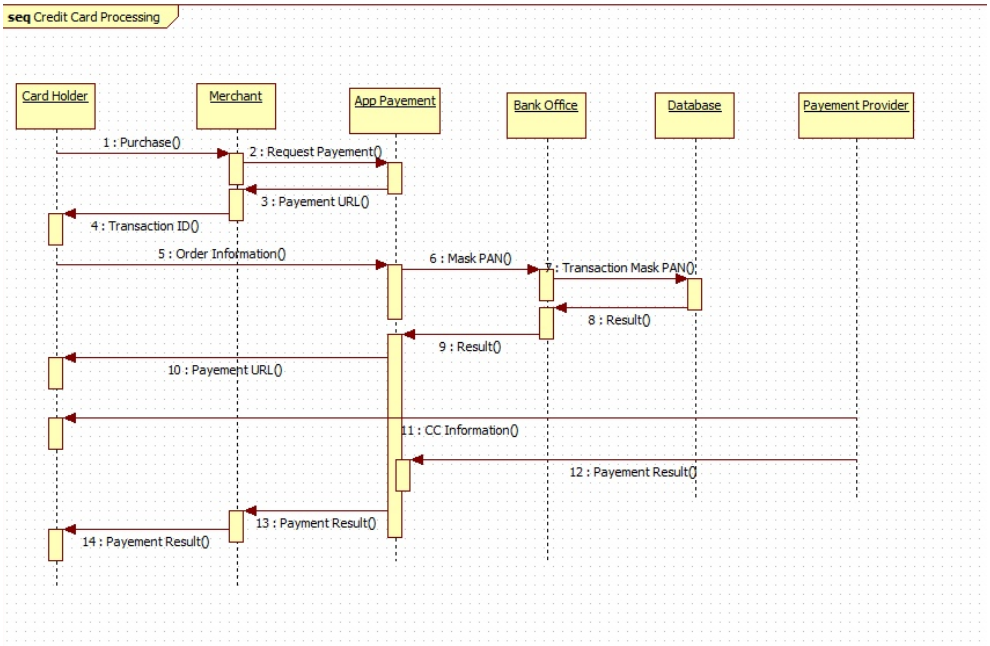
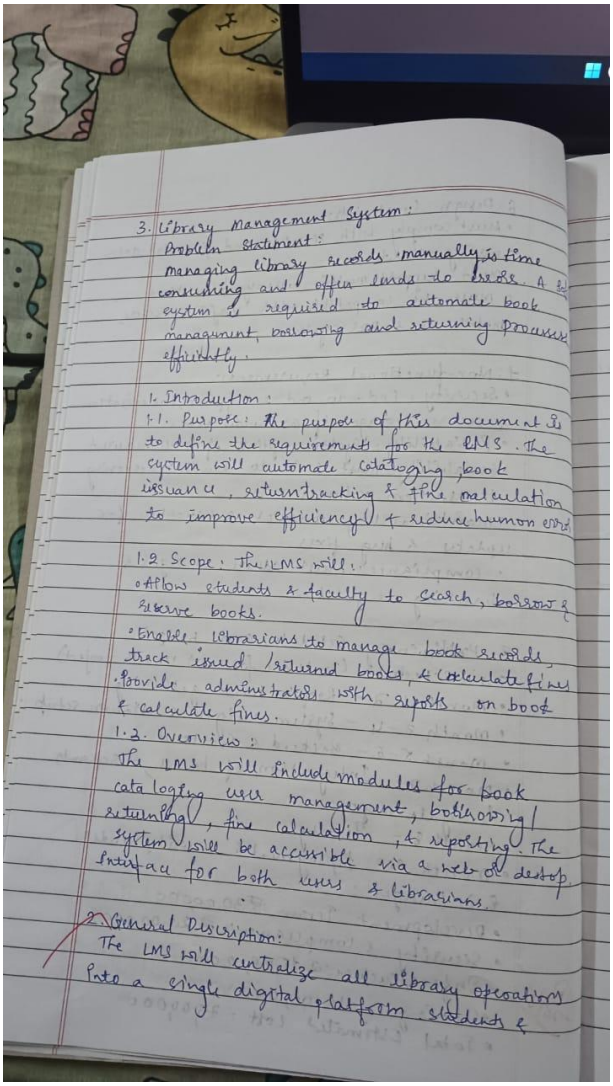


Fig 2.6 Activity Diagram for Credit Card Processing

This is an Activity Diagram that describes the workflow for applying for a credit card. It uses two swimlanes to distinguish between the actions of the Credit Card Applicant and the Credit Card Management System. The process begins with the applicant making an enquiry on the bank's system. The system shows available credit cards, and the applicant picks one. The system then displays the requirements. A decision point follows: if the applicant does not meet the requirements, they may need to pick a different card. If they do, they proceed to comply with the requirements by submitting information. The system then validates this information and displays the confirmed application to the applicant for final review before the process concludes.

3. Library Management System



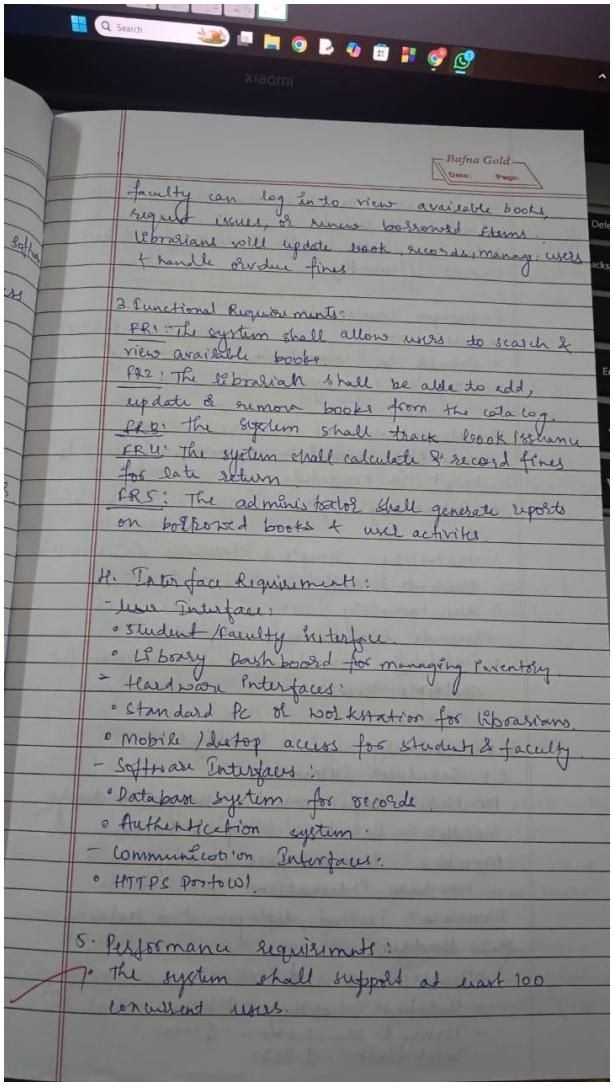
3. Library Management System:
Problem Statement:
Managing library records manually is time consuming and often leads to errors. A system is required to automate book management, borrowing and returning processes efficiently.

1. Introduction:
1.1. Purpose: The purpose of this document is to define the requirements for the LMS. The system will automate cataloging, book issuance, return tracking & fine calculation to improve efficiency & reduce human errors.

1.2 Scope: The LMS will:
- Allow students & faculty to search, borrow & reserve books.
- Enable librarians to manage book records, track issued/returned books, & calculate fines.
- Provide administrators with reports on book & calculate fines.

1.2. Overview:
The LMS will include modules for book cataloging, user management, borrowing/returning, fine calculation, & reporting. The system will be accessible via a web or desktop interface for both users & librarians.

2. General Description:
The LMS will centralize all library operations into a single digital platform. Students &



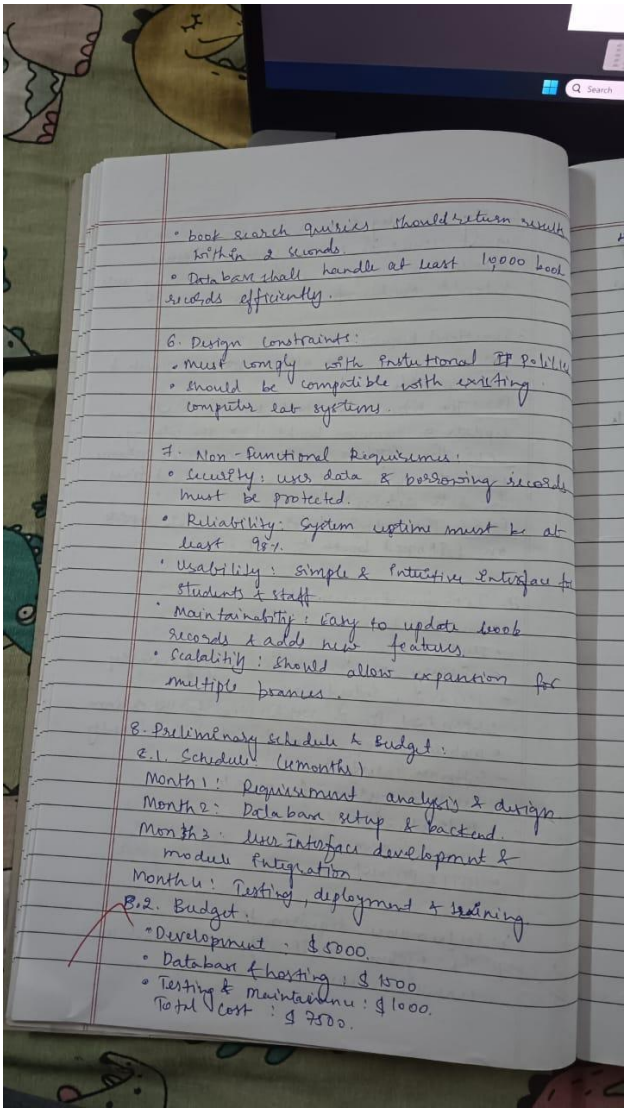
Bafna Gold

faculty can log in to view available books, request issues, & renew borrowed items. Librarians will update book records, manage users & handle overdue fines.

3. Functional Requirements:
FR1: The system shall allow users to search & view available books.
FR2: The librarian shall be able to add, update & remove books from the catalog.
FR3: The system shall track book issuance.
FR4: The system shall calculate & record fines for late return.
FR5: The administrator shall generate reports on borrowed books & user activity.

4. Interface Requirements:
- User Interface:
  - Student/Faculty interface.
  - Library dashboard for managing inventory.
- Hardware Interfaces:
  - Standard PC or workstation for librarians.
  - Mobile/desktop access for students & faculty.
- Software Interfaces:
  - Database system for records.
  - Authentication system.
- Communication Interfaces:
  - HTTPS protocol.

5. Performance requirements:
- The system shall support at least 100 concurrent users.



- book search queries should return results within 2 seconds
- Database shall handle at least 10000 book records efficiently.

6. Design constraints:
- must comply with institutional IT policies
- should be compatible with existing computer lab systems.

7. Non-functional Requirements:
- Security: user data & borrowing records must be protected.
- Reliability: System uptime must be at least 98%.
- Usability: simple & intuitive interface for students & staff
- Maintainability: Easy to update book records & add new features.
- Scalability: should allow expansion for multiple branches.

8. Preliminary schedule & Budget:
8.1. Schedule (4 months)
Month 1: Requirement analysis & design
Month 2: Database setup & backend.
Month 3: User Interface development & module integration.
Month 4: Testing, deployment & training.
8.2. Budget:
- Development: $5000.
- Database & hosting: $500
- Testing & Maintainance: $1000.
Total cost: $7500.

2.3 Class Diagram

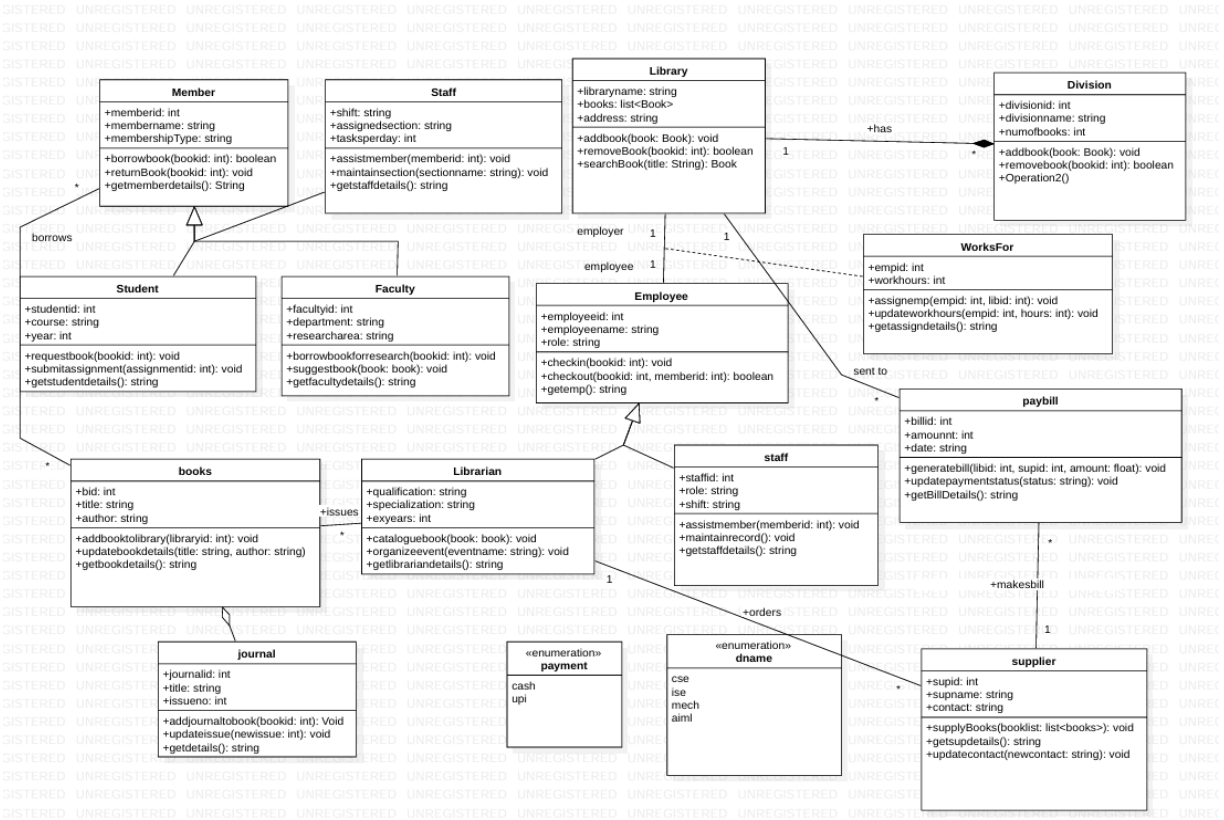


Fig 3.1 Class Diagram for Library Management System

This is a comprehensive Class Diagram representing the structure of a Library Management System. It details the different entities (classes) within the system, their attributes (data), and their methods (behaviors). Key classes include Library, Member, Book, Staff, and Supplier. The diagram shows relationships between these classes, such as inheritance, where Student and Faculty are types of Member. It also illustrates associations, for example, a Member borrows a Book, and a Librarian (a type of Staff) issues a Book. This diagram serves as a blueprint for the system, defining how different components are organized and interact with each other at a structural level.

3.4 State Diagram

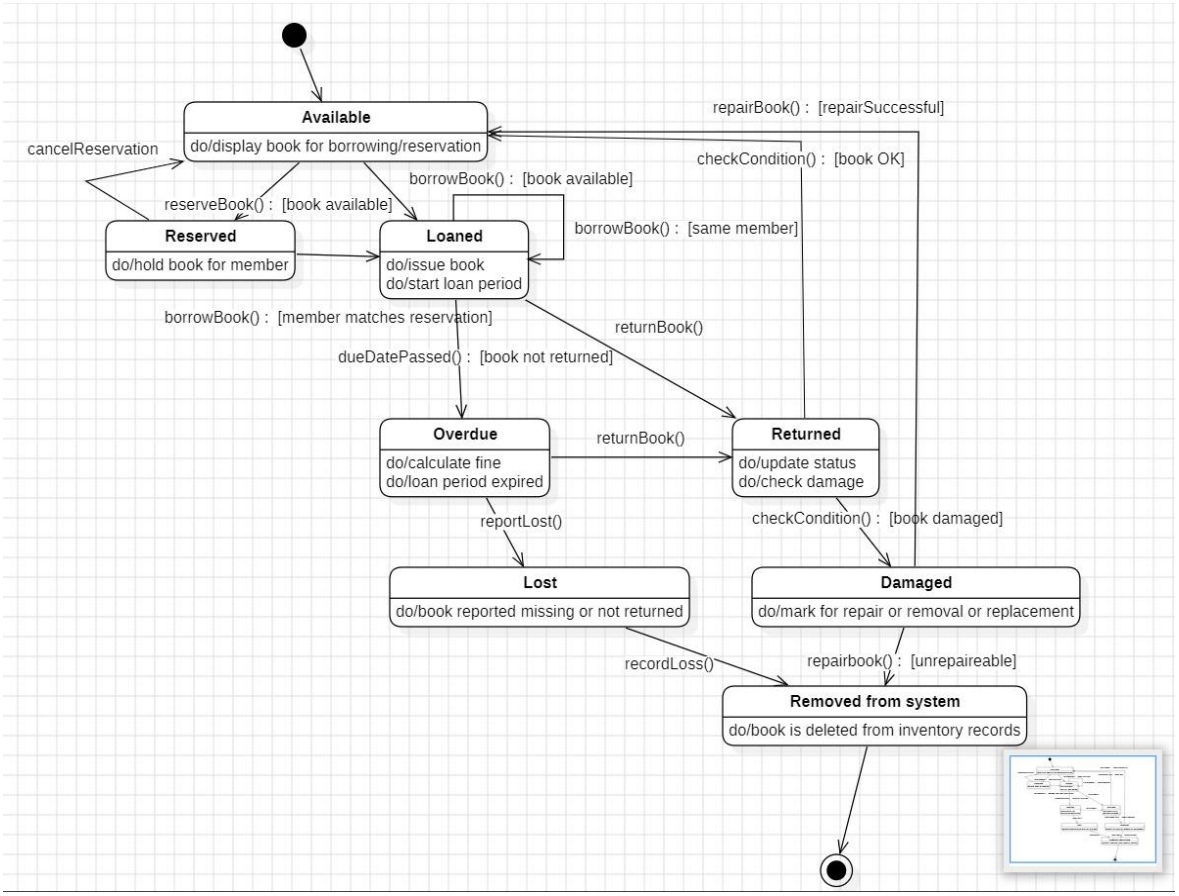


Fig 3.2 State Diagram for Library Management System

This State Machine Diagram illustrates the complete lifecycle of a book within the library system. It tracks the various states a book can be in, starting from Available on the shelf. From there, a book can be Reserved or directly Loaned. If a loaned book is not returned on time, it becomes Overdue. When a book is returned, its state changes to Returned, and its condition is checked. Depending on the condition, it either goes back to Available or is marked as Damaged. The diagram also accounts for situations where a book is Lost or damaged beyond repair, in which case it is Removed from system, marking the end of its lifecycle.

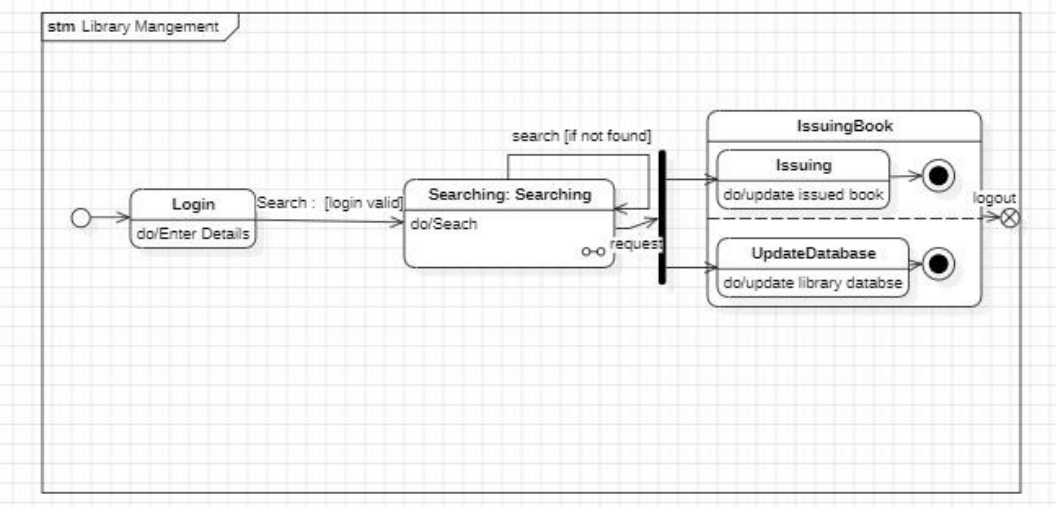


Fig 3.3 Advance State Diagram for Library Management System (1)

This diagram is a high-level State Machine Diagram for the Library Management system, focusing on a user's session. The process begins with a Login state. Upon valid authentication, the system transitions to a Searching state, where the user can look for books. From the searching state, the user can make a request, which moves the system into the IssuingBook composite state. This state includes the internal processes of Issuing the book and UpdateDatabase. After the issuing process is complete, or if the user chooses to, the system transitions to a final logout state, ending the session.

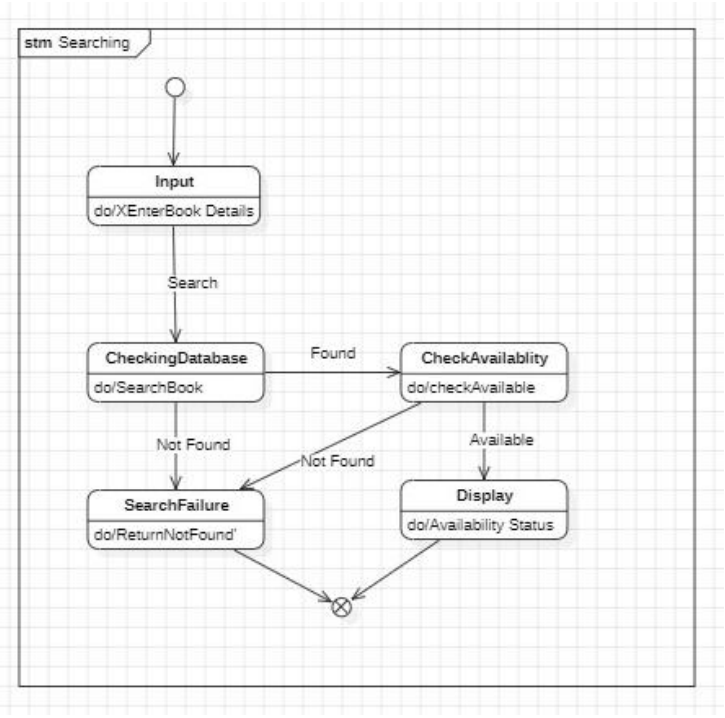


Fig 3.4 Advance State Diagram for Library Management System (2)

This is a detailed State Machine Diagram that focuses specifically on the "Searching" process within the library system. The workflow starts in the Input state, where the user enters the details of the book they are looking for. The system then transitions to CheckingDatabase to search its records. If the book is found, the system moves to CheckAvailability. If it's not found in the database, the state becomes SearchFailure. From the availability check, if the book is available, its status is shown in the Display state. If it is not available, this also leads to a failure or status update. The diagram shows how the system handles a search query from input to final output.

3.5 Use Case Diagram

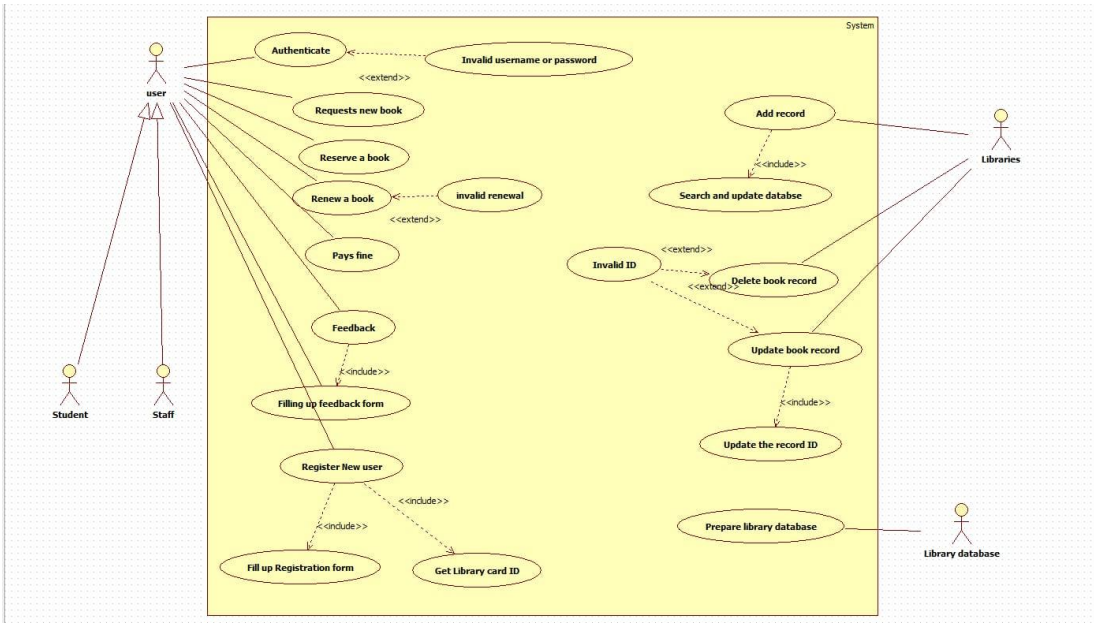


Fig 3.5 Use Case Diagram for Library Management System

This is a Use Case Diagram that provides a functional overview of the library system from the perspective of its users (actors). The primary actors are the general user (which can be a Student or Staff), the Librarian, and the Library database. The diagram outlines all the actions or use cases available, such as Authenticate, Reserve a book, Renew a book, and Pays fine for regular users. For administrative users like the Librarian, use cases include Add record, Search and update database, and Delete book record. The diagram uses relationships like "extend" and "include" to show dependencies and optional interactions, for instance, an Invalid username or password message extends the Authenticate use case.

3.6 Sequence Diagram

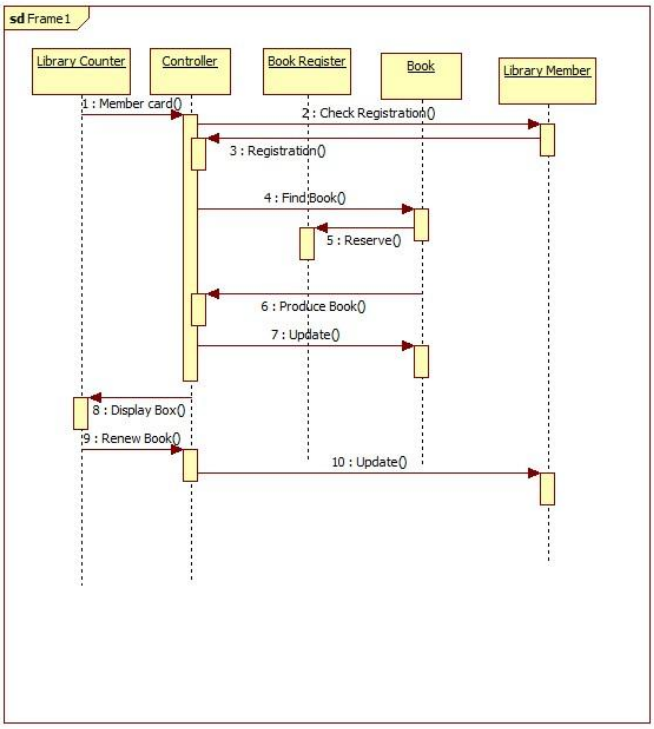


Fig 3.6 Sequence Diagram for Library Management System

This Sequence Diagram illustrates the step-by-step interaction between different components of the system during a book reservation or renewal process at a library counter. The participants (lifelines) include the Library Counter, a central Controller, Book Register, Book, and Library Member. The sequence begins when a member presents their card at the counter. The Controller then orchestrates the entire process by sending messages to the other components in a specific order to check the member's registration, find and reserve the book, and update the book's status. The diagram clearly shows the flow of communication and the order in which operations are performed to fulfill the user's request.

3.7 Activity Diagram

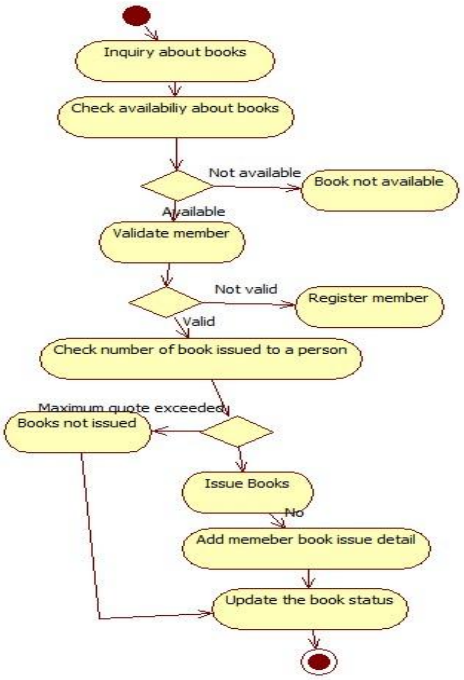
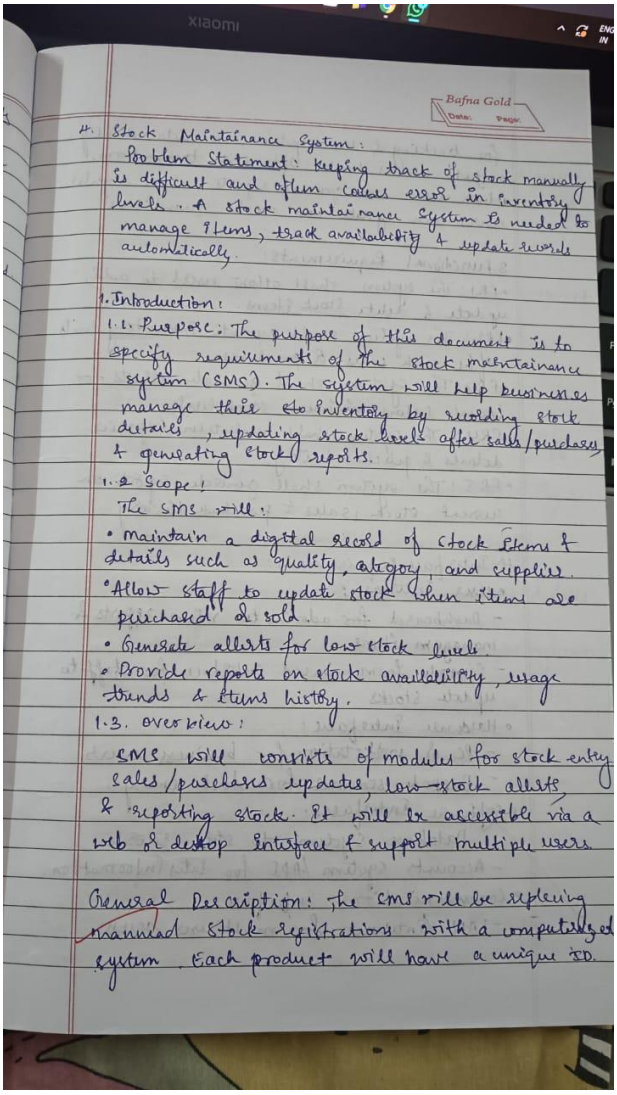


Fig 3.7 Activity Diagram for Library Management System

This Activity Diagram details the business logic for the process of issuing a book to a member. The workflow starts with an Inquiry about books and proceeds to Check availability. A decision point follows: if the book is not available, the process terminates. If it is available, the system then validates the member's status. If the member is not valid, they may be prompted to register. For a valid member, the system checks the number of books they have already issued. Another decision checks if they have exceeded their borrowing limit. If the limit is not exceeded, the book is issued, details are recorded, the book's status is updated, and the process completes successfully. If the limit is exceeded, the book is not issued.

4. Stock Maintenance System



4. Stock Maintainance System:

Problem Statement: Keeping track of stock manually is difficult and often causes error in inventory levels. A stock maintainance system is needed to manage items, track availability & update records automatically.

1. Introduction:

1.1. Purpose: The purpose of this document is to specify requirements of the stock maintainance system (SMS). The system will help businesses manage their stock inventory by recording stock details, updating stock levels after sales/purchases & generating stock reports.

1.2 Scope:

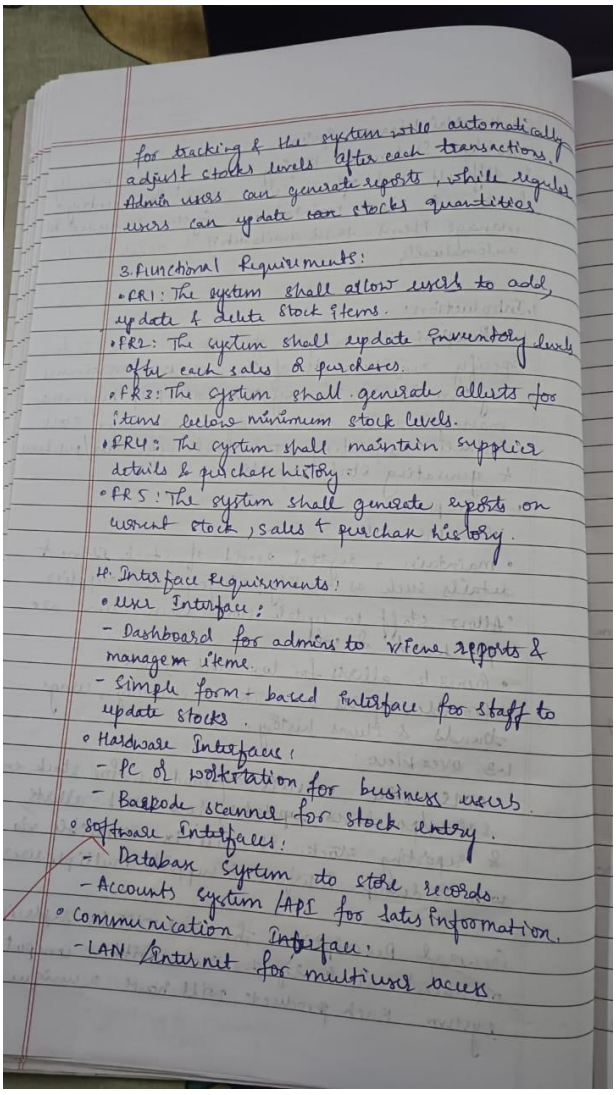
The SMS will:

- Maintain a digital record of stock items & details such as quality, category, and supplier.
- Allow staff to update stock when items are purchased & sold.
- Generate alerts for low stock levels.
- Provide reports on stock availability, usage trends & items history.

1.3. Overview:

SMS will consists of modules for stock entry, sales/purchases updates, low-stock alerts, & reporting stock. It will be accessible via a web & desktop interface & support multiple users.

General Description: The SMS will be replacing manual stock registrations with a computerized system. Each product will have a unique ID.



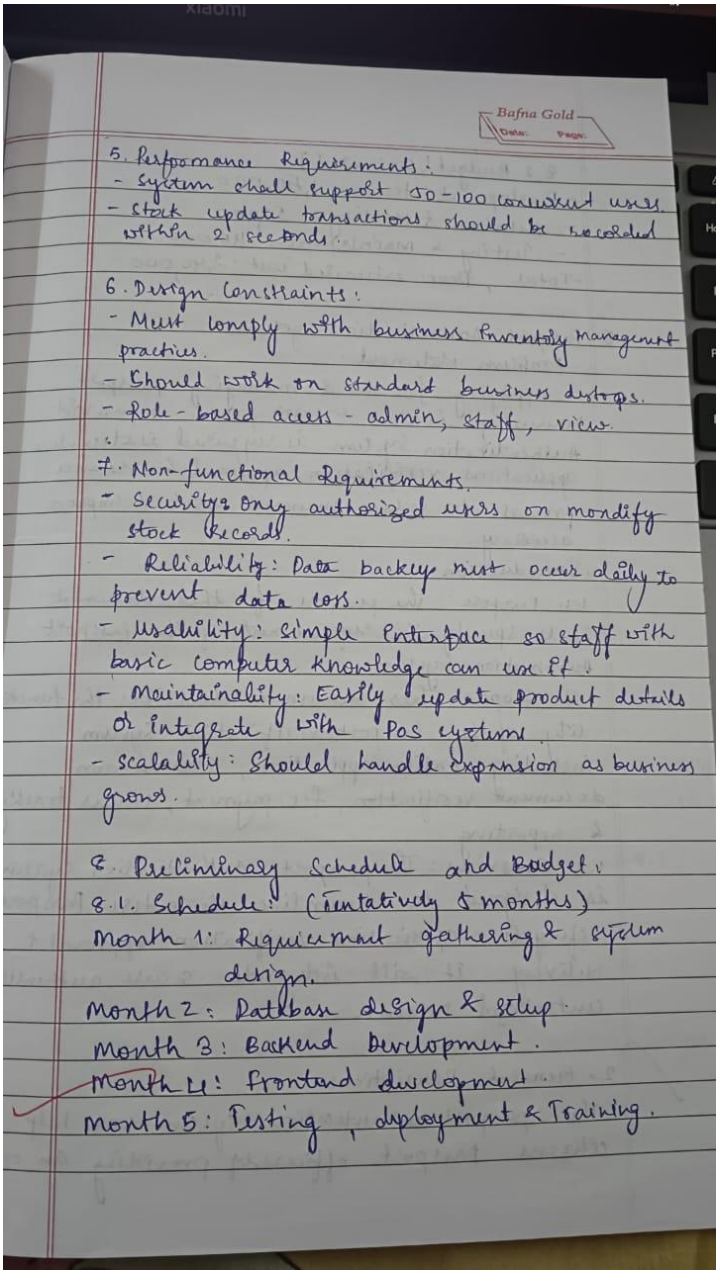
for tracking & the system will automatically adjust stocks levels after each transactions. Admin users can generate reports, while regular users can update can stocks quantities.

3. Functional Requirements:

- FR1: The system shall allow users to add, update & delete stock items.
- FR2: The system shall update inventory levels after each sales & purchases.
- FR3: The system shall generate alerts for items below minimum stock levels.
- FR4: The system shall maintain supplier details & purchase history.
- FR5: The system shall generate reports on current stock, sales & purchase history.

4. Interface Requirements:

- User Interface:
  - Dashboard for admins to view reports & manage items.
  - Simple form-based interface for staff to update stocks.
- Hardware Interfaces:
  - PC or workstation for business users.
  - Barcode scanner for stock entry.
- Software Interfaces:
  - Database system to store records.
  - Accounts system/API for sales information.
- Communication Interface:
  - LAN/Internet for multiuser access.



5. Performance Requirements:
- System shall support 50-100 concurrent users.
- Stock update transactions should be recorded within 2 seconds.

6. Design Constraints:
- Must comply with business inventory management practices.
- Should work on standard business desktops.
- Role-based access - admin, staff, view.

7. Non-functional Requirements:
- Security: Only authorized users on modify stock records.
- Reliability: Data backup must occur daily to prevent data loss.
- Usability: simple interface so staff with basic computer knowledge can use it.
- Maintainability: Easily update product details or integrate with Pos systems.
- scalability: Should handle expansion as business grows.

8. Preliminary Schedule and Budget:
8.1. Schedule: (Tentatively 5 months)
Month 1: Requirement gathering & system design.
Month 2: Database design & setup.
Month 3: Backend Development.
Month 4: Frontend development.
Month 5: Testing, deployment & Training.

4.3 Class Diagram

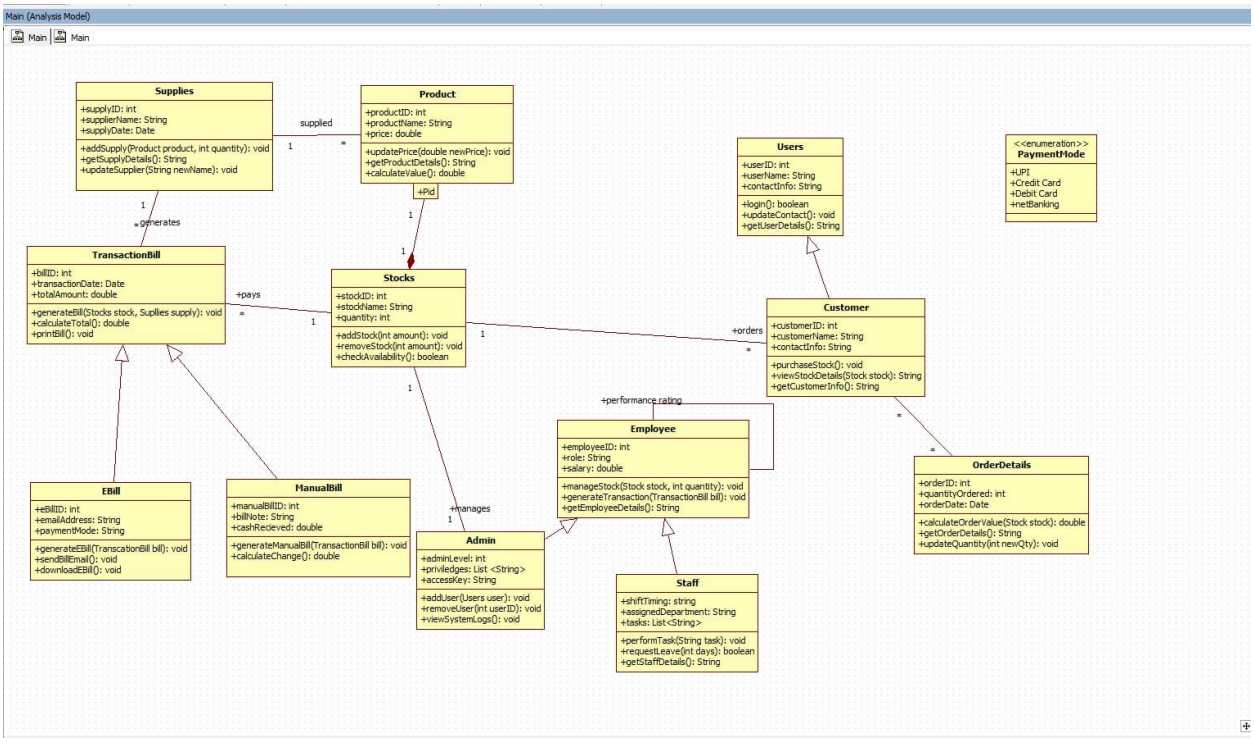


Fig 4.1 Class Diagram for Stock Maintenance System

This is a Class Diagram that provides a detailed blueprint of an inventory or stock management system. It defines the main components (classes) such as Product, Stocks, Suppliers, Customer, and Employee. The diagram outlines the properties (attributes) and functionalities (methods) of each class. It also illustrates the relationships between them; for example, a Customer places an OrderDetails, an Employee manages Stocks, and a Supplier is associated with TransactionBill and supplies Product. The inheritance relationship is also shown, with Admin and Staff being specialized types of Employee, and EBill and ManualBill being types of TransactionBill.

4.2 State Diagram

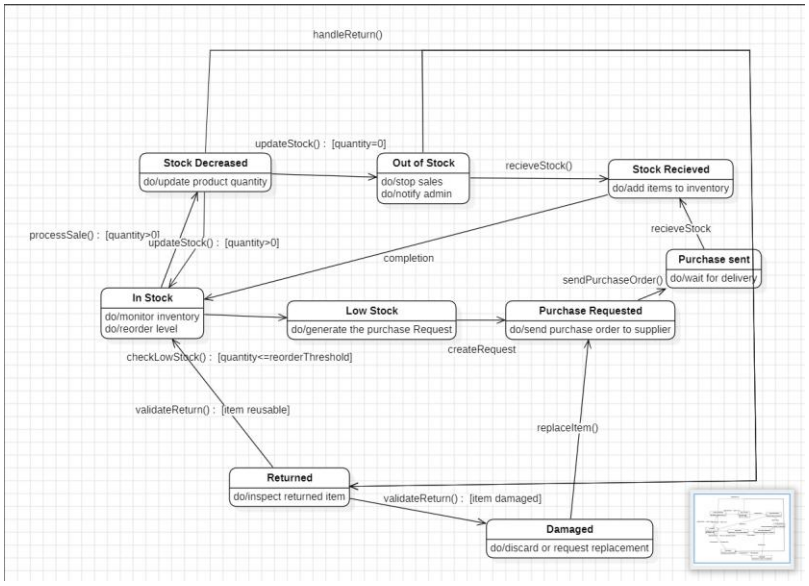


Fig 4.2 State Diagram for Stock Maintenance System(1)

This State Machine Diagram visualizes the dynamic behavior and lifecycle of stock for a particular item. The stock begins in the In Stock state. A sale (processSale) transitions it to Stock Decreased. If the quantity reaches zero, it moves to Out of Stock, which triggers a notification to the admin. When new inventory arrives (recieveStock), the state becomes Stock Recieved and then returns to In Stock. The diagram also handles reordering: if the stock level falls to a certain threshold, it enters the Low Stock state, which leads to a Purchase Requested and Purchase sent sequence to replenish inventory. Furthermore, it manages returns, with states like Returned and Damaged to handle items coming back from customers.

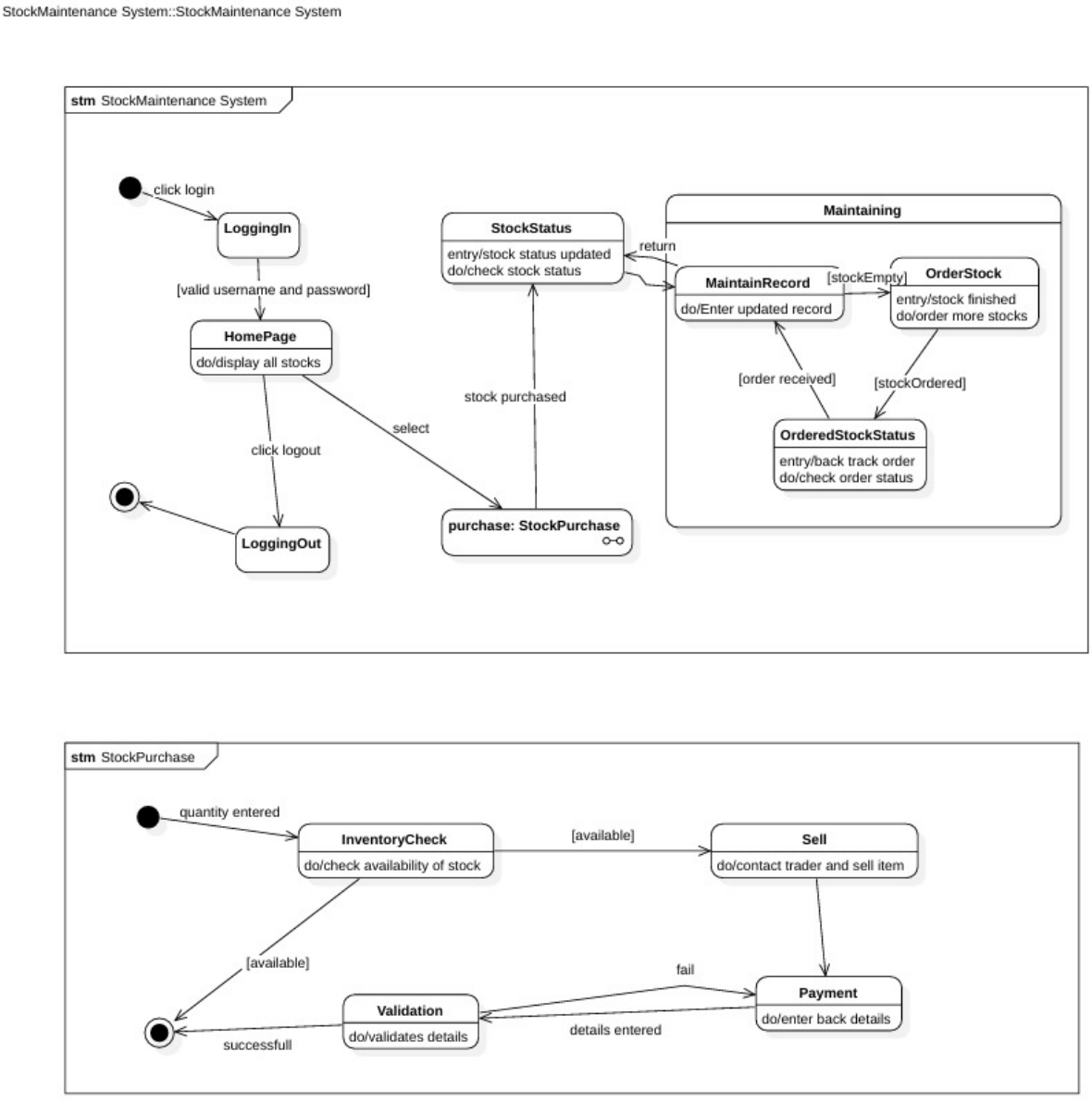


Fig 4.3 State Diagram for Stock Maintenance System(2)

These two diagrams are State Machine Diagrams that illustrate the user interaction and processes within the Stock Maintenance System. The first diagram shows the overall system flow, starting from a user LoggingIn. Upon successful login, the user reaches the HomePage. From here, they can select a stock to view its StockStatus or initiate a purchase. The Maintaining section is a composite state that includes actions like MaintainRecord and OrderStock. The second diagram provides a detailed view of the StockPurchase process. It starts with an InventoryCheck to verify availability. If available, it moves to Validation of details. Upon successful validation, the process proceeds to Sell and Payment, completing the purchase transaction.

4.3 Use Case Diagram

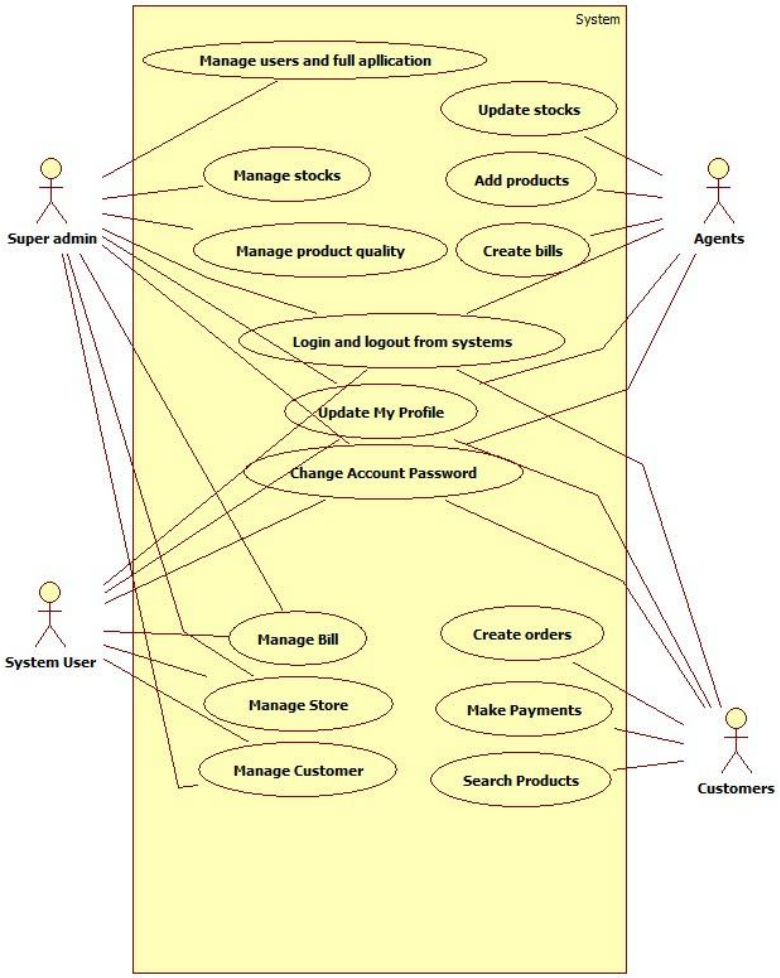


Fig 4.4 Use Case Diagram for Stock Maintenance System

This is a Use Case Diagram that outlines the functionalities of the system and the actors who interact with it. It identifies four main actors: Super admin, System User, Agents, and Customers. The Super admin has the highest level of access, with use cases like Manage users and full application. Agents are responsible for inventory-related tasks such as Update stocks and Add products. System Users have more general permissions like managing bills and customers. Customers can perform actions like Create orders, Make Payments, and Search Products. The diagram provides a clear, high-level view of what the system does and who can do what.

4.4 Sequence Diagram

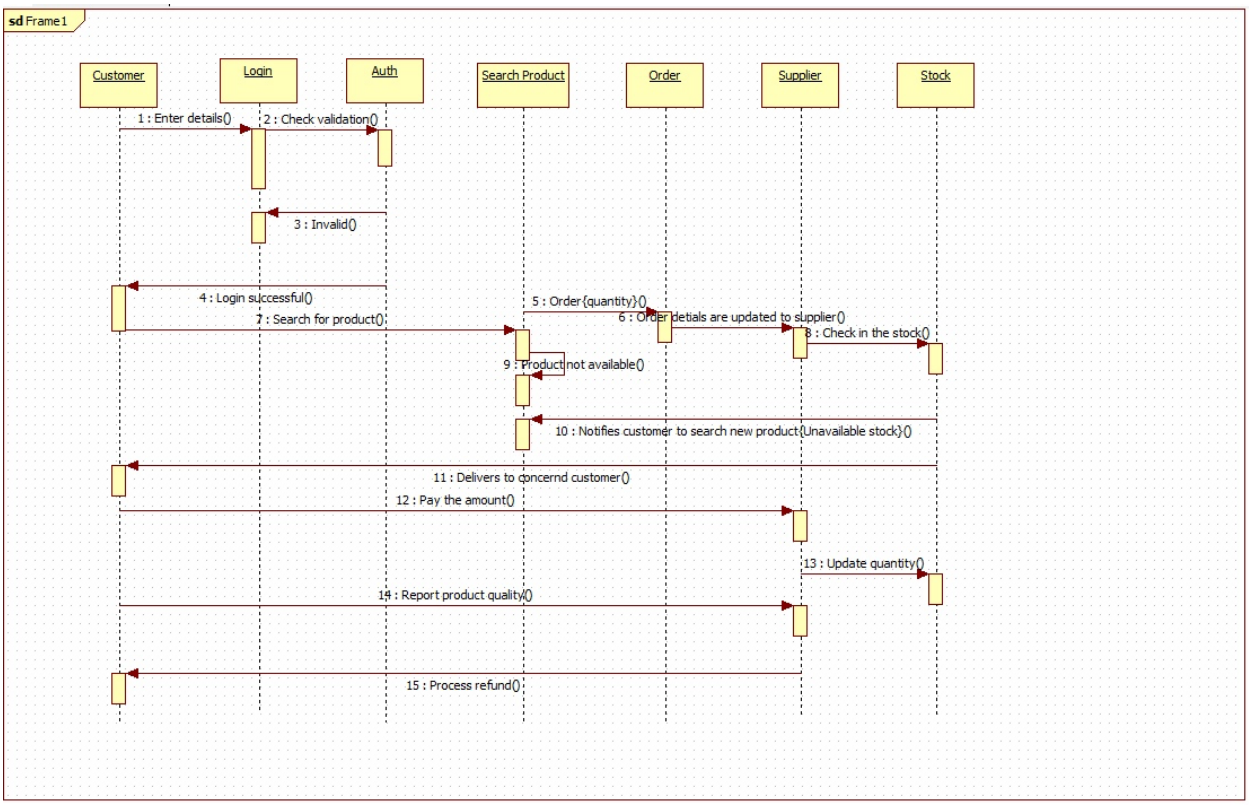


Fig 4.5 Sequence Diagram for Stock Maintenance System

This Sequence Diagram illustrates the interactions that occur when a customer orders a product. The participants are Customer, Login, Auth (Authentication), Search Product, Order, Supplier, and Stock. The sequence begins with the Customer entering their details to log in. After successful authentication, the customer searches for a product. The system checks the Stock. If the product is available, the customer places an Order, and the details are updated with the Supplier. The diagram also shows subsequent steps like the supplier delivering the product, the customer making a payment, and potentially reporting on product quality, which updates the supplier's records. It effectively shows the message flow between different system components over time.

4.5 Activity Diagram

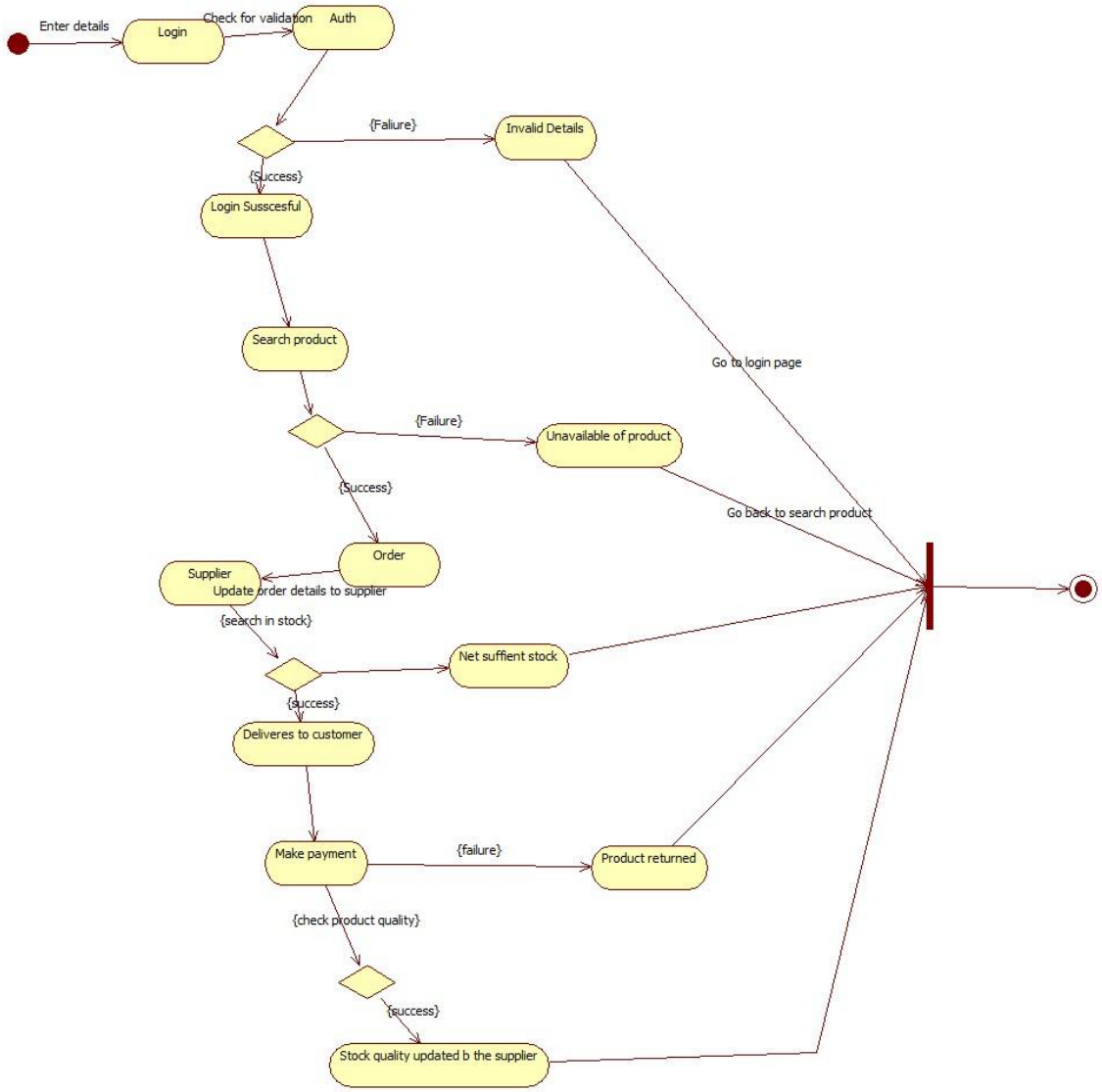
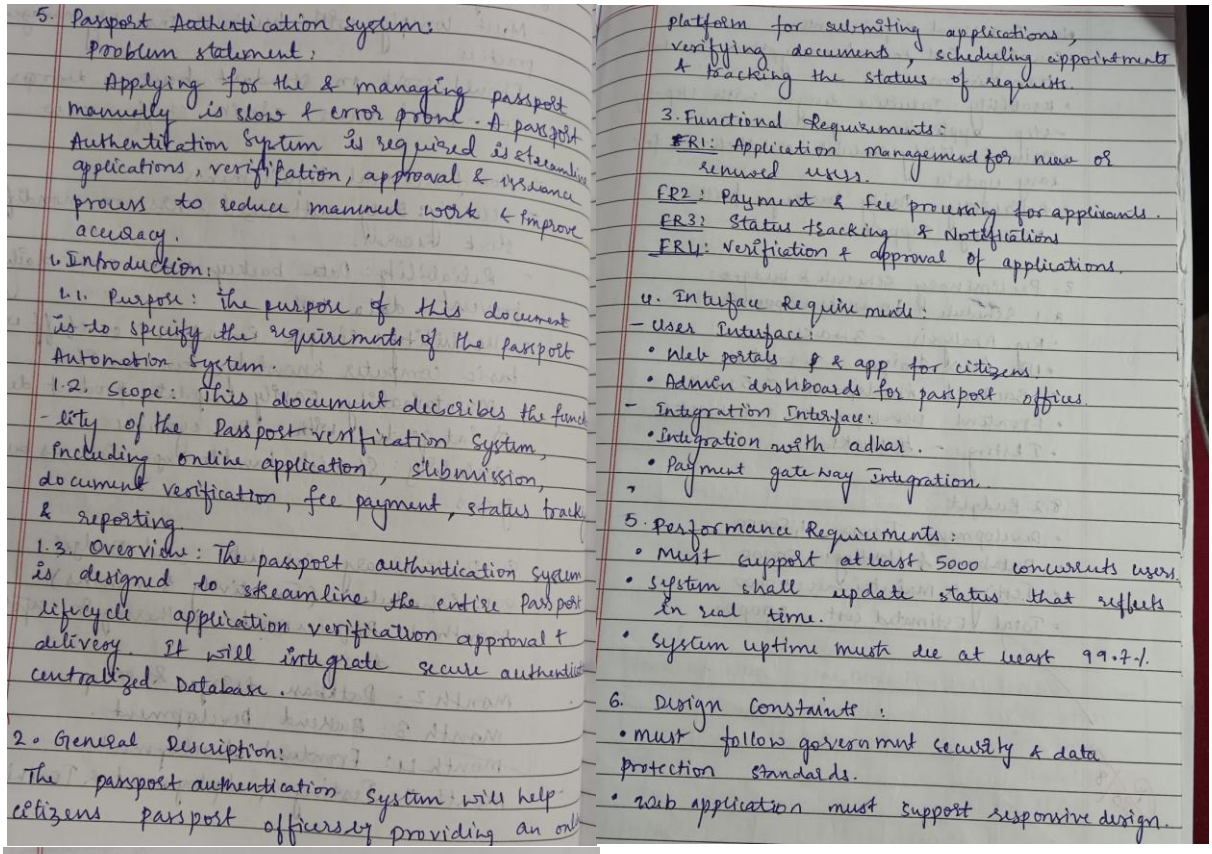


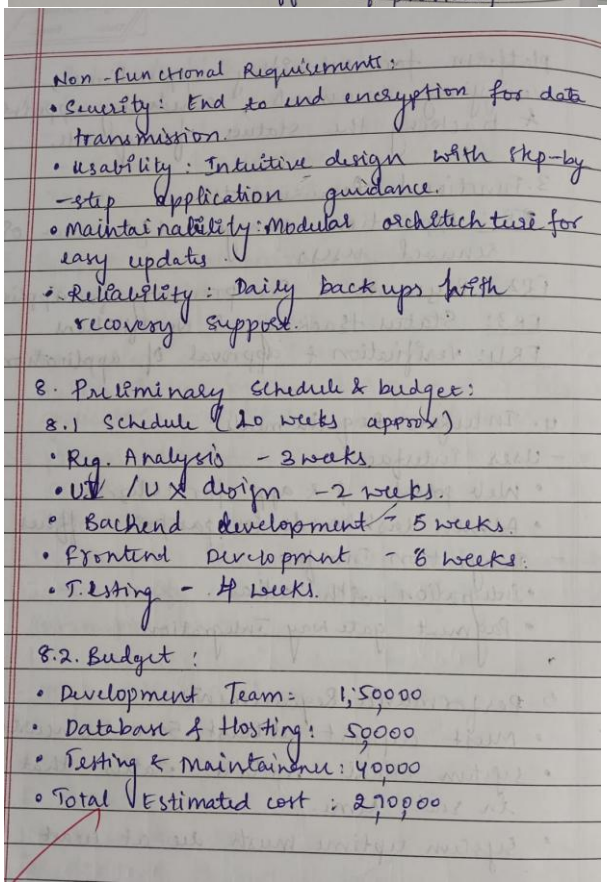
Fig 4.5 Activity Diagram for Stock Maintenance System

This Activity Diagram maps out the end-to-end workflow for a customer purchasing an item. The process starts with the user entering their details for Login and Authentication. A decision node checks for success or failure. If successful, the flow continues to Search product. Another decision checks if the product is available. If it is, the customer can place an Order. The system then checks if there is sufficient stock. If so, the product is delivered, and the customer proceeds to Make payment. The final steps involve a quality check, which may result in updating the stock quality with the supplier, after which the entire process comes to an end. The diagram clearly shows the various paths and decision points in the transaction process.

5. Passport Authentication System:



5. Passport Authentication System:
Problem statement:
Applying for the & managing passport manually is slow & error prone. A passport Authentication System is required is streamline applications, verification, approval & issuance process to reduce manual work & improve accuracy.
1. Introduction:
1.1. Purpose: The purpose of this document is to specify the requirements of the passport Automation System.
1.2. Scope: This document describes the functionality of the Passport verification System, including online application, submission, document verification, fee payment, status tracking & reporting.
1.3. Overview: The passport authentication system is designed to streamline the entire passport lifecycle application verification approval & delivery. It will integrate secure authentication centralized Database.
2. General Description:
The passport authentication System will help citizens passport officers by providing an online platform for submitting applications, verifying documents, scheduling appointments & tracking the status of requests.
3. Functional Requirements:
FR1: Application Management for new or renewed users.
FR2: Payment & fee processing for applicants.
FR3: Status tracking & Notifications
FR4: Verification & approval of applications.
4. Interface Requirements:
- User Interface:
• Web portals & app for citizens.
• Admin dashboards for passport offices.
- Integration Interface:
• Integration with adhar.
• Payment gateway Integration.
5. Performance Requirements:
• Must support at least 5000 concurrents users.
• System shall update status that reflects in real time.
• System uptime must be at least 99.7%.
6. Design Constraints:
• must follow government security & data protection standards.
• Web application must support responsive design.



Non-Functional Requirements:
• Security: End to end encryption for data transmission.
• Usability: Intuitive design with step-by-step application guidance.
• Maintainability: Modular architecture for easy updates.
• Reliability: Daily backups with recovery support.
8. Preliminary Schedule & budget:
8.1 Schedule (20 weeks approx)
• Req. Analysis - 3 weeks
• UI / UX design - 2 weeks.
• Backend development - 5 weeks.
• Frontend Development - 6 weeks.
• Testing - 4 weeks.
8.2. Budget:
• Development Team: 1,50,000
• Database & Hosting: 50,000
• Testing & Maintainance: 40,000
• Total Estimated cost: 2,70,000

5.2 Class Diagram

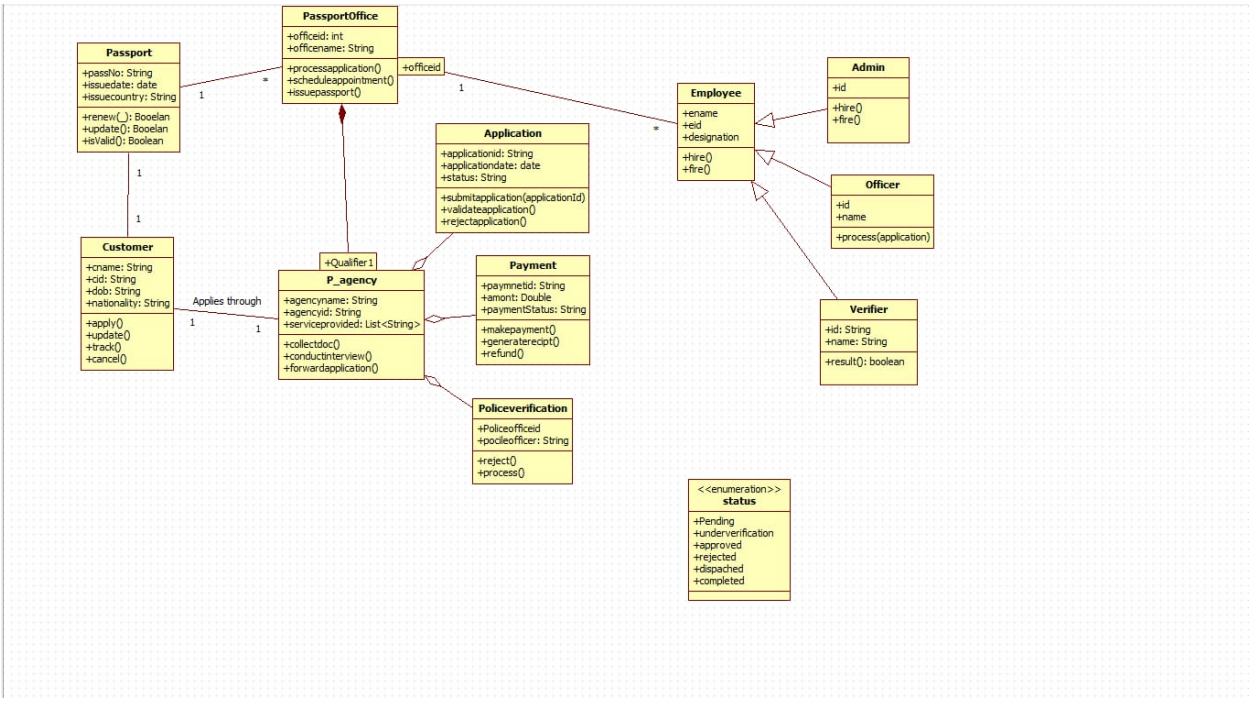


Fig 5.1 Class Diagram for Passport Authentication System

The Class Diagram illustrates the static structure of the system by defining the different objects (classes), their attributes, and their relationships. Key classes include Customer, Passport, Application, PassportOffice, and various types of Employee like Officer and Verifier. It shows, for example, that a Customer can have one Passport and can apply for an Application. A PassportOffice employs various Employee roles and processes applications. This diagram essentially serves as the blueprint for the system's database and object-oriented programming structure.

5.3 State Diagram

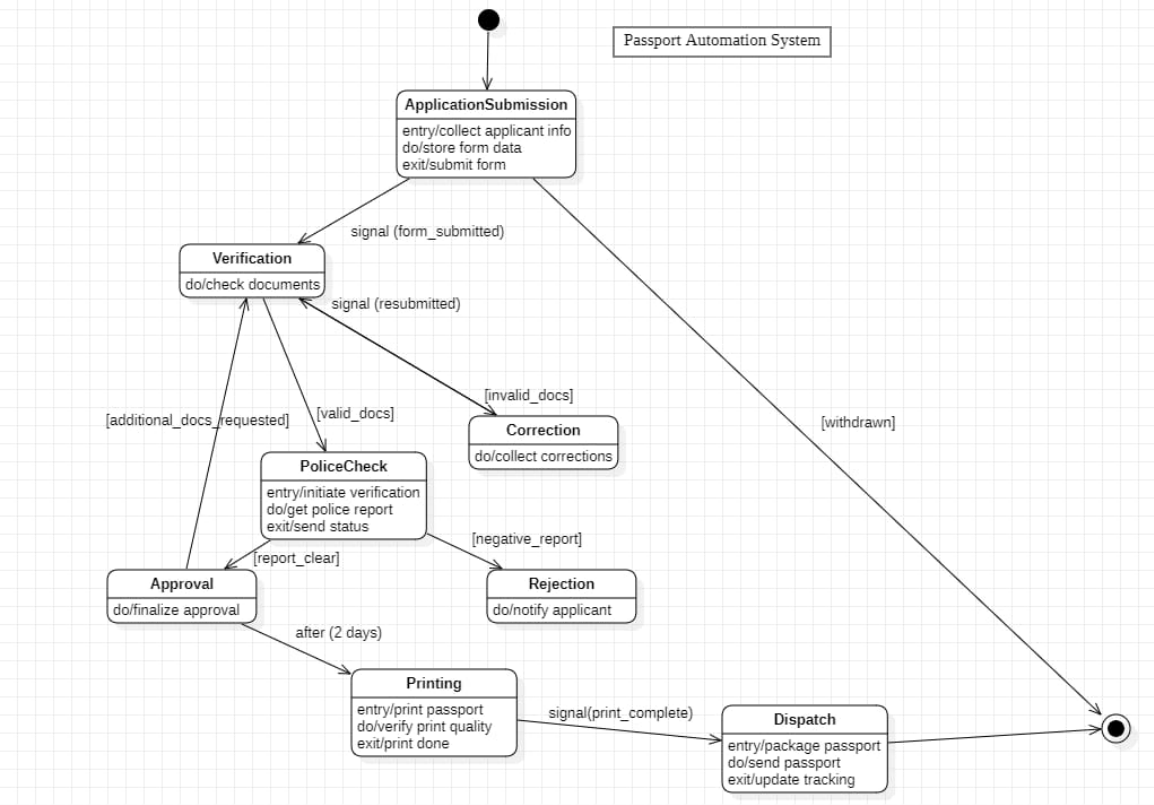


Fig 5.2 State Diagram for Passport Authentication System

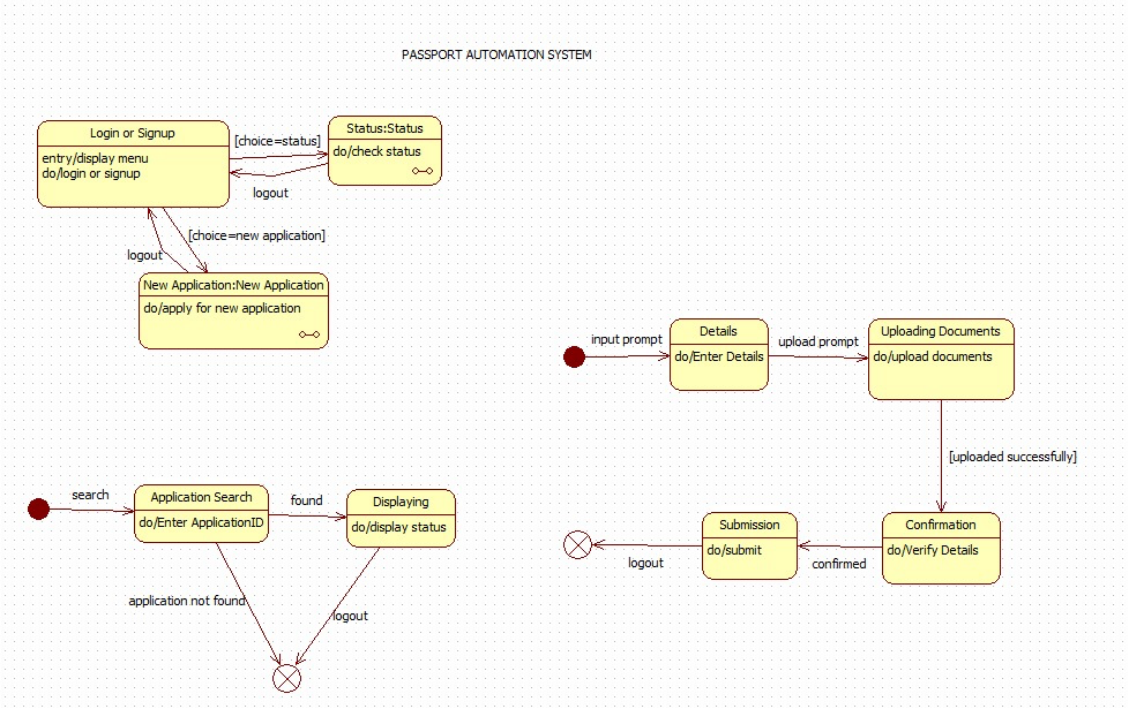


Fig 5.3 State Diagram for Passport Authentication System

The first State Machine Diagram depicts the lifecycle of a passport application from beginning to end. It shows the various states an application can be in, such as ApplicationSubmission, Verification, PoliceCheck, Approval, Printing, and Dispatch. The arrows represent transitions between these states, triggered by events like form_submitted, valid_docs, or negative_report. This diagram is crucial for understanding the dynamic behavior and the overall workflow of the passport processing journey.

The second State Machine Diagram focuses on the user interface and interaction flows. It is composed of several smaller diagrams showing different user activities. One part details the process for a user to Login or Signup, check Status, or start a New Application. Another part outlines the steps for submitting a new application, which includes entering Details, Uploading Documents, Confirmation, and final Submission. The last part shows how a user can search for an application by ID to display its status.

5.4 Use Case Diagram

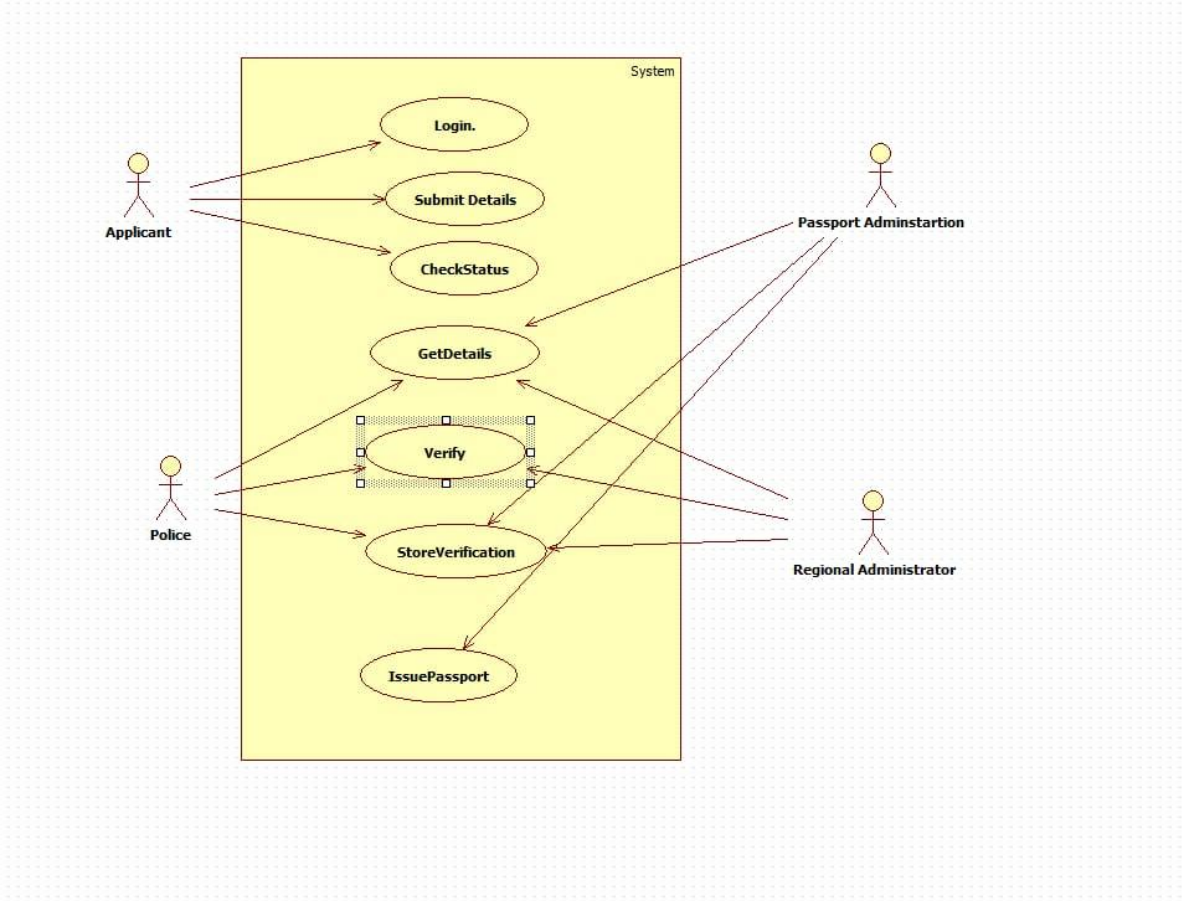


Fig 5.4 Use Case Diagram for Passport Authentication System

The Use Case Diagram provides a high-level overview of the system's functionalities from the perspective of different users (actors). The actors identified are the Applicant, Police, Passport Administrator, and Regional Administrator. It outlines the key actions or "use cases" each actor can perform. For instance, an Applicant can Submit Details and CheckStatus, while the Police actor is involved in StoreVerification. This diagram helps in understanding the system's requirements and who interacts with which function.

5.5 Sequence Diagram

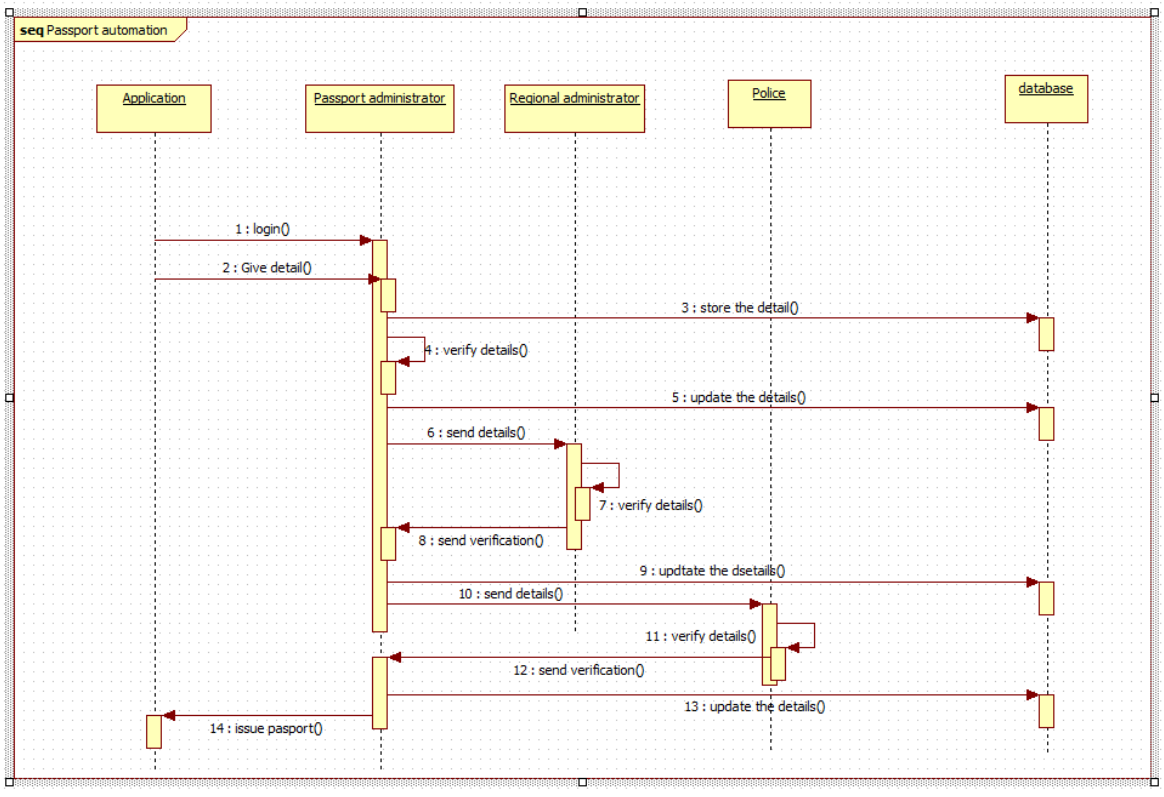


Fig 5.5 Sequence Diagram for Passport Authentication System

The Sequence Diagram details the interactions between different components of the system over time for a specific scenario. It shows the chronological sequence of messages exchanged between the Application, Passport administrator, Regional administrator, Police, and the database. The diagram illustrates the step-by-step process of how details are submitted, stored, verified by different authorities in a specific order, and how the database is updated at each stage, leading up to the final issue passport command.

5.6 Activity Diagram

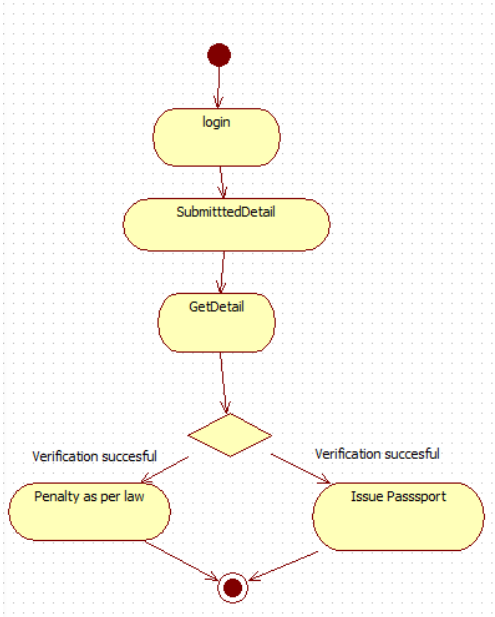


Fig 5.6 Activity Diagram for Passport Authentication System

The Activity Diagram illustrates a specific workflow within the system, focusing on the sequence of actions and decisions. This particular diagram shows the process starting from login, followed by SubmittedDetail and GetDetail. After the details are retrieved, a decision point is reached based on the outcome of the verification. If the verification is successful, the system proceeds to Issue Passport; otherwise, it leads to a "Penalty as per law". This highlights the conditional logic in the verification process.