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:

SHREYASH SHAURYA
1BM23CS354

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 Shreyash Shaurya (1BM23CS354) during the 5th Semester August 2025-December 2025.

Signature of the Faculty Incharge:
Ropashree S (Assistant Professor)

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

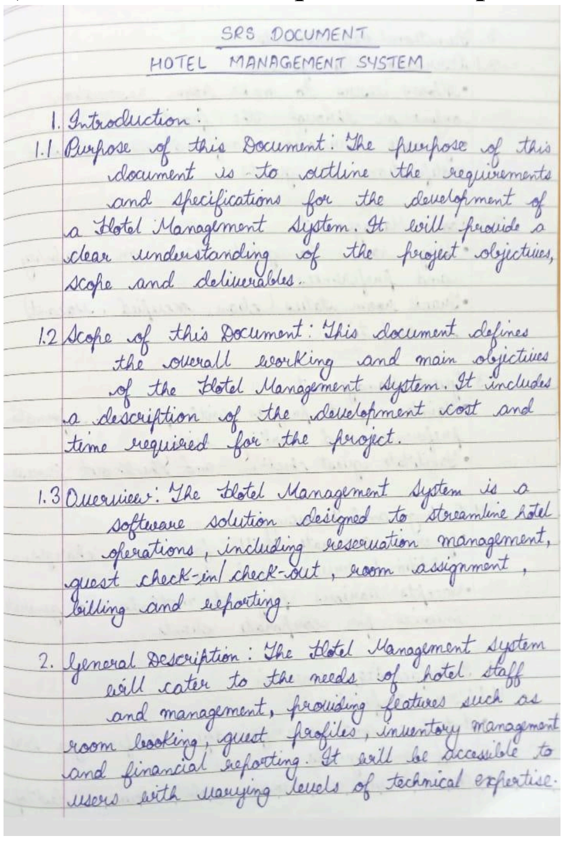
Table of Contents

1. Hotel Management System

1) Problem Statement

Managing hotel operations manually leads to inefficiencies, errors, and delays in handling reservations, guest management, billing, and room allocation. Hotels often struggle with maintaining accurate records, preventing overbooking, ensuring timely guest check-in/check-out, and generating financial reports. A centralized, automated Hotel Management System is required to streamline these operations, enhance service quality, and improve overall administrative control.

2) SRS-Software Requirements Specification



SRS DOCUMENT

HOTEL MANAGEMENT SYSTEM

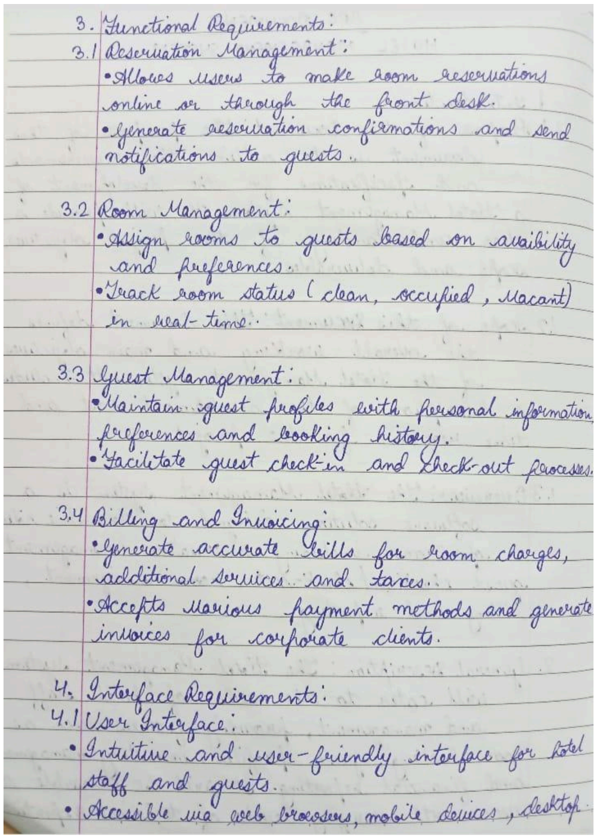
1. Introduction:

1.1 Purpose of this Document: The purpose of this document is to outline the requirements and specifications for the development of a Hotel Management System. It will provide a clear understanding of the project objectives, scope and deliverables.

1.2 Scope of this Document: This document defines the overall working and main objectives of the Hotel Management System. It includes a description of the development cost and time required for the project.

1.3 Overview: The Hotel Management System is a software solution designed to streamline hotel operations, including reservation management, guest check-in/check-out, room assignment, billing and reporting.

2. General Description: The Hotel Management System will cater to the needs of hotel staff and management, providing features such as room booking, guest profiles, inventory management and financial reporting. It will be accessible to users with varying levels of technical expertise.



3. Functional Requirements:

3.1 Reservation Management:
- Allows users to make room reservations online or through the front desk.
- Generate reservation confirmations and send notifications to guests.

3.2 Room Management:
- Assign rooms to guests based on availibility and preferences.
- Track room status (clean, occupied, vacant) in real-time.

3.3 Guest Management:
- Maintain guest profiles with personal information, preferences and booking history.
- Facilitate guest check-in and check-out processes.

3.4 Billing and Invoicing:
- Generate accurate bills for room charges, additional services and taxes.
- Accepts various payment methods and generate invoices for corporate clients.

4. Interface Requirements:

4.1 User Interface:
- Intuitive and user-friendly interface for hotel staff and guests.
- Accessible via web browsers, mobile devices, desktop

4.2 Integration Interfaces:
- Integration with payment gateways for secure transactions.
- Integration with third-party booking platforms for seamless reservation management.

5. Performance Requirements:

5.1 Response Time:
- The system should respond to user actions within 2 seconds.

5.2 Scalability:
- Handle a minimum of 1000 concurrent users during peak hours.

5.3 Data Integrity:
- Ensure data consistency and accuracy across all modules.

6. Design Constraints

6.1 Hardware Limitations:
- The system should be compatible with standard hotel hardware (computers, printers, POS terminals)

6.2 Software Dependencies:
- Utilize a relational database management system (eg., MySQL) for data storage.
- Use programming languages and frameworks conduciu to UML modeling (eg., Java, Spring Boot).

7. Non-Functional Attributes:

7.1 Security:
- Implement robust authentication and authoriza mechanisms to protect sensitive data.

7.2 Reliability:
- Ensure high availability and fault tolerance to minimize system downtime.

7.3 Scalability:
- Design the system to accomodate future growt and expansion.

7.4 Portability:
- Support multiple platforms and devices for user accessibility.

7.5 Usability:
- The system shall have a user-friendly interface with clear navigation.

7.6 Reusability:
- The system shall use modular core design to facilitate future enchancements and maintenance

7.7 Compatibility:
- The system shall be compatible with common web browsers (Chrome, Firefox, Safari).

7.8 Data Integrity:
- The system shall ensure accurate and consistent data storage and retrieval.

8. Preliminary Schedule and Budget: The development of the Hotel Management System is estimated to take 6 months with a budget of $100,000. This includes project planning, development, testing and deployment phases.

3) Class Diagram

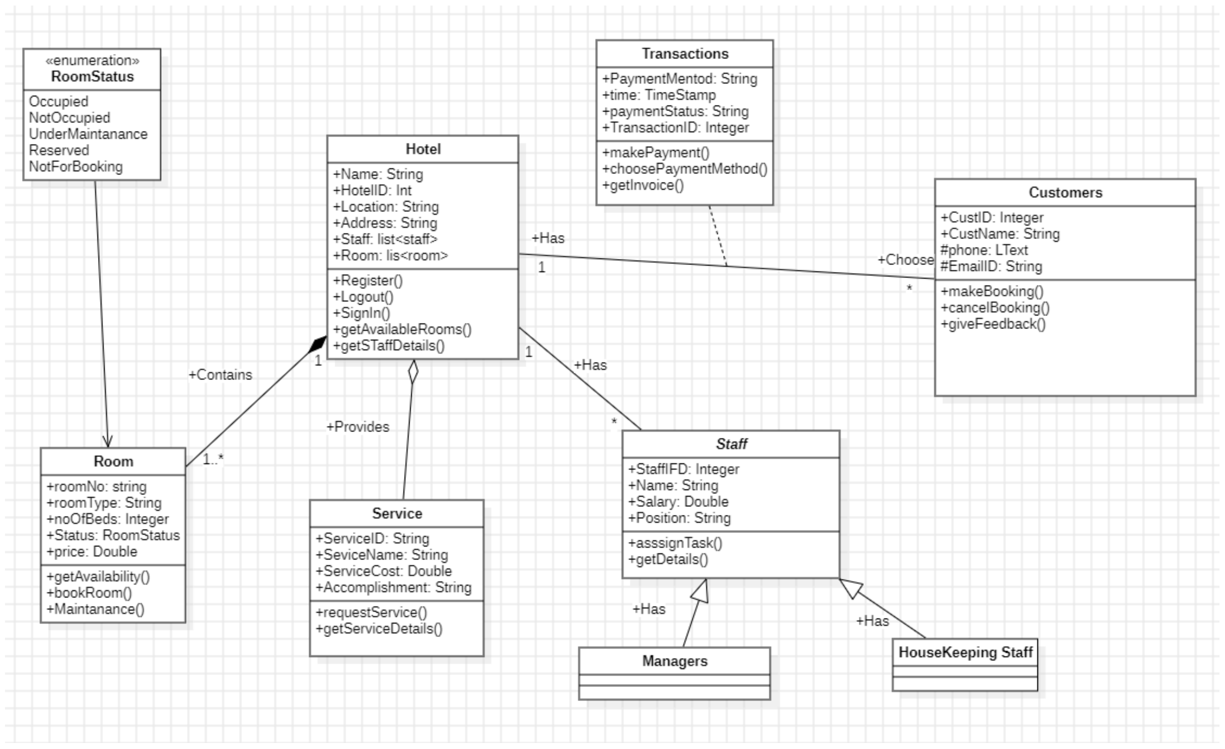


Fig 1.1: Class Diagram for Hotel Management System

The class diagram for the Hotel Management System represents how core entities like Guest, Reservation, Room, and Invoice structurally interact to support hotel operations. It captures the permanent data relationships needed for booking, billing, and room allocation. The associations show how a single reservation may involve multiple services, and composition ensures data integrity for invoices and payments. This structure allows efficient querying, updates, and extension of functionality. It forms the backbone on which all hotel workflows operate.

4) State Diagram

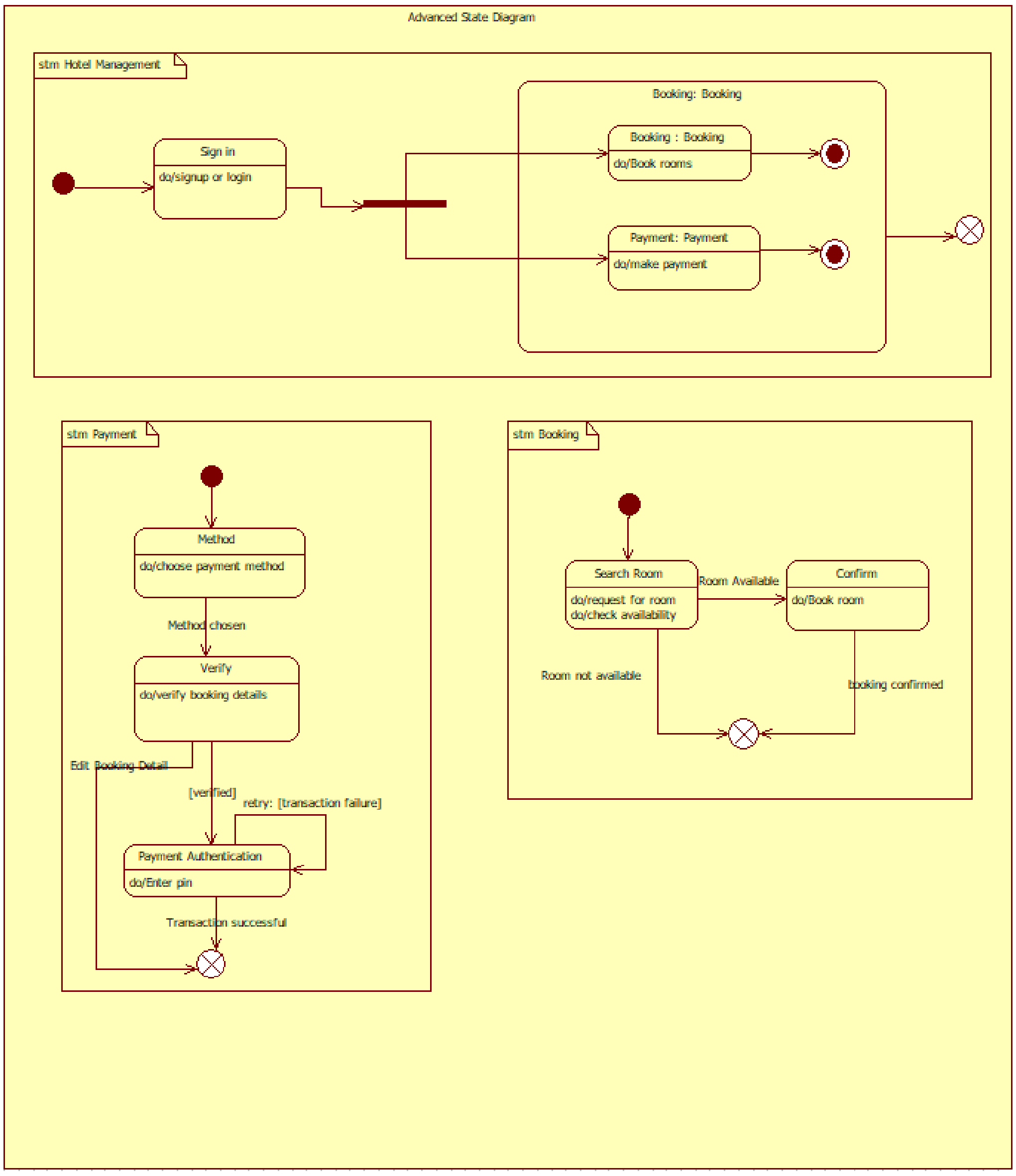


Fig 1.2: State Diagram for Hotel Management System

The state diagram models the lifecycle of a reservation as it moves through various operational phases of a hotel. It shows all valid states—from being created, awaiting payment, checked-in, occupied, and checked-out—along with transitions caused by user actions or system events. This representation ensures no invalid state jumps occur and highlights exceptional flows like cancellations or no-shows. It helps verify that the reservation logic behaves consistently under different scenarios.

5) Use Case Diagram

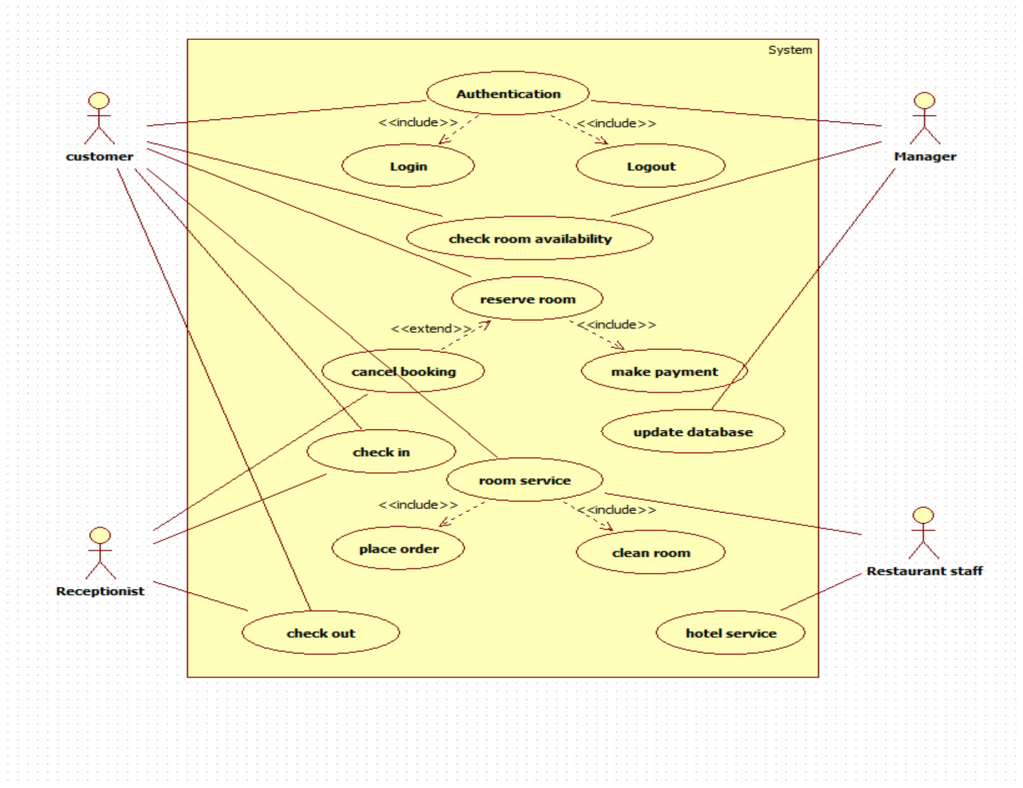


Fig 1.3: Use Case Diagram for Hotel Management System

The use-case diagram outlines how different users such as guests, front desk staff, and administrators interact with the hotel system. It maps the functional expectations, including booking rooms, checking in guests, generating invoices, and updating room statuses. Relationships like include and extend emphasize optional or reusable behaviors within the process. This diagram provides a clear boundary of responsibilities and interactions the system must support.

6) Sequence Diagram

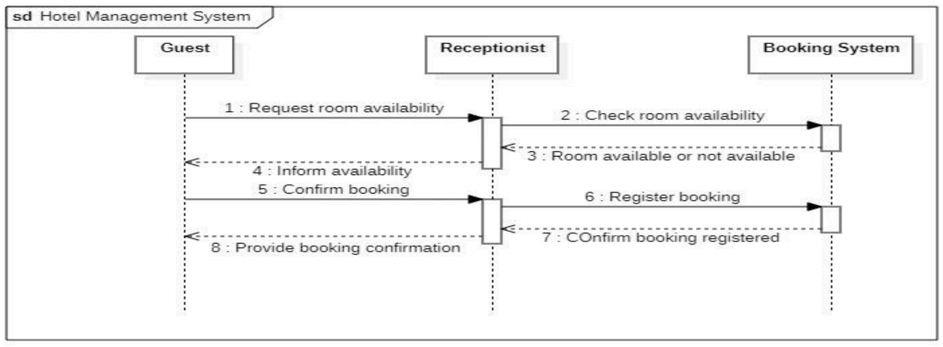

sd Hotel Management System
Guest
Receptionist
Booking System
1 : Request room availability
2 : Check room availability
3 : Room available or not available
4 : Inform availability
5 : Confirm booking
6 : Register booking
7 : COnfirm booking registered
8 : Provide booking confirmation

Fig 1.4: Sequence Diagram for Hotel Management System

- The sequence diagram shows the exact runtime order in which different components of the hotel system communicate to complete operations like booking or check-in. It details how controllers, services, and repositories exchange messages and how external systems such as payment gateways fit into the interaction. The diagram exposes timing, dependencies, and potential bottlenecks, making it valuable for validating the implementation flow.

7) Activity Diagram

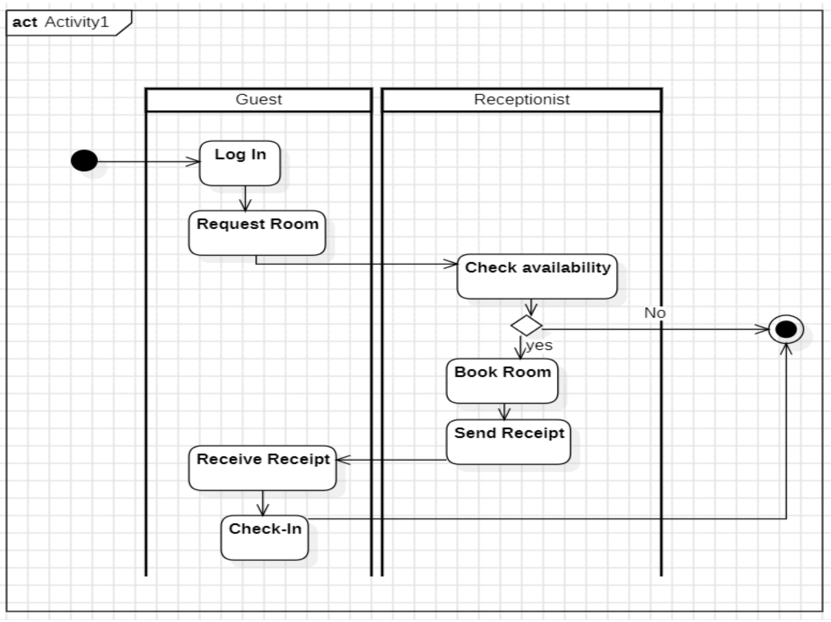


Fig 1.5: Activity Diagram for Hotel Management System

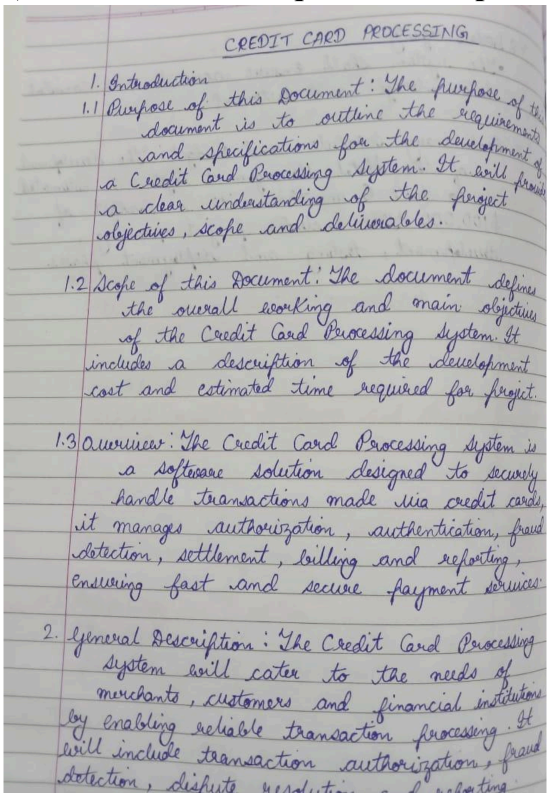
The activity diagram describes the operational workflow for a process such as a room booking or guest check-in. It visualizes the sequence of steps, decision points, loops, and parallel tasks involved. This helps clarify the internal logic and ensures that all possible paths—successful, failed, or exceptional—are accounted for. It is essential for validating the completeness and efficiency of the designed workflow.

2. Credit Card Processing System

1) Problem Statement

Traditional credit card transaction handling involves complex, error-prone manual processes for authorization, settlement, fraud detection, and record maintenance. With the increasing volume of digital payments, banks and merchants need a secure and efficient system that can process transactions in real time, reduce fraud risk, and maintain reliable financial records. A Credit Card Processing System is needed to automate these processes and ensure secure, fast, and accurate payment handling.

2) SRS-Software Requirements Specification



CREDIT CARD PROCESSING

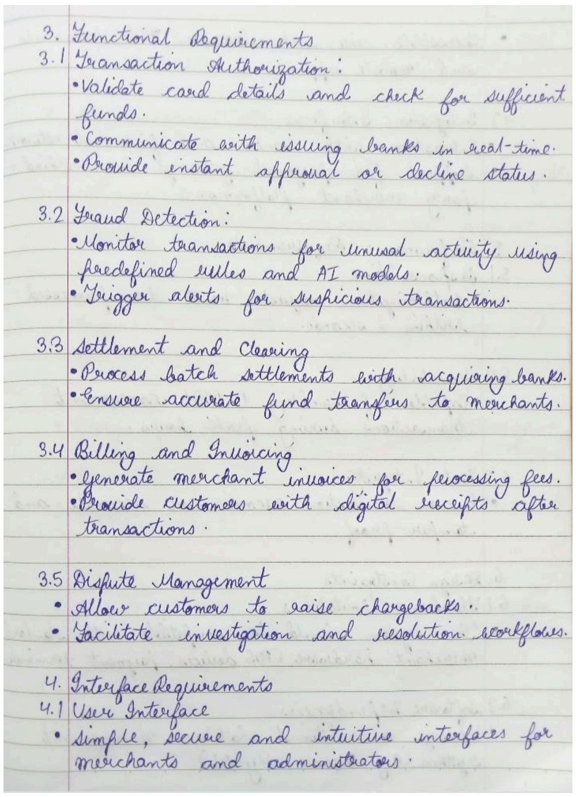
1. Introduction

1.1 Purpose of this Document: The purpose of the document is to outline the requirements and specifications for the development of a Credit Card Processing System. It will provide a clear understanding of the project objectives, scope and deliverables.

1.2 Scope of this Document: The document defines the overall working and main objectives of the Credit Card Processing System. It includes a description of the development cost and estimated time required for project.

1.3 Overview: The Credit Card Processing System is a software solution designed to securely handle transactions made via credit cards, it manages authorization, authentication, fraud detection, settlement, billing and reporting, ensuring fast and secure payment services.

2. General Description: The Credit Card Processing System will cater to the needs of merchants, customers and financial institutions by enabling reliable transaction processing. It will include transaction authorization, fraud detection, dispute resolution and reporting



3. Functional Requirements

3.1 Transaction Authorization:
- Validate card details and check for sufficient funds.
- Communicate with issuing banks in real-time.
- Provide instant approval or decline status.

3.2 Fraud Detection:
- Monitor transactions for unusual activity using predefined rules and AI models.
- Trigger alerts for suspicious transactions.

3.3 Settlement and Clearing
- Process batch settlements with acquiring banks.
- Ensure accurate fund transfers to merchants.

3.4 Billing and Invoicing
- Generate merchant invoices for processing fees.
- Provide customers with digital receipts after transactions.

3.5 Dispute Management
- Allow customers to raise chargebacks.
- Facilitate investigation and resolution workflows.

4. Interface Requirements

4.1 User Interface
- Simple, secure and intuitive interfaces for merchants and administrators.

- Accessible via web browsers, POS terminals and mobile apps.

4.2 Integration Interfaces
- Integration with banks and financial networks
- Support for payment gateways and third-party merchant platforms.

5. Performance Requirements
5.1 Response Time
- Authorization requests must be processed within 2 seconds.

5.2 Scalability
- Capable of handling 10,000+ concurrent transactions during peak hours.

5.3 Data Integrity
- Ensure transaction records are accurate and tamper-proof.

6. Design Constraints
6.1 Hardware Limitations
- The system should be compatible with standard merchant hardware (POS devices, payment terminals)

6.2 Software Dependencies
- Utilize a relational database management system (e.g., Oracle, MySQL).

- Employ secure communication protocols (e.g., SSL/TLS, PCI-DSS compliance).

7. Non-Functional Attributes
7.1 Security
- Implement end-to-end encryption and tokenization of card data.
- Comply with PCI-DSS standards for payment security.

7.2 Reliability
- Ensure 99.99% uptime with redundancy and failover mechanisms.

7.3 Scalability
- Design for expansion to accomodate future growth.

7.4 Portability
- Support multiple deployment environments (cloud, on-premise)

7.5 Usability
- Provide a user-friendly interface with clear transaction reporting.

7.6 Reusability
- Modular design to allow reuse in future payment systems.

7.7 Compatibility
- Compatible with common browsers, operating systems and POS standards

7.8 Data Integrity
- Ensure accurate transaction recording and reconciliation across all modules.

8. Preliminary Schedule and Budget
The development of the Credit Card Processing System is estimated to take 8 months with a budget of $250,000. This includes project planning, development, compliance audits, testing and deployment phases.

3) Class Diagram

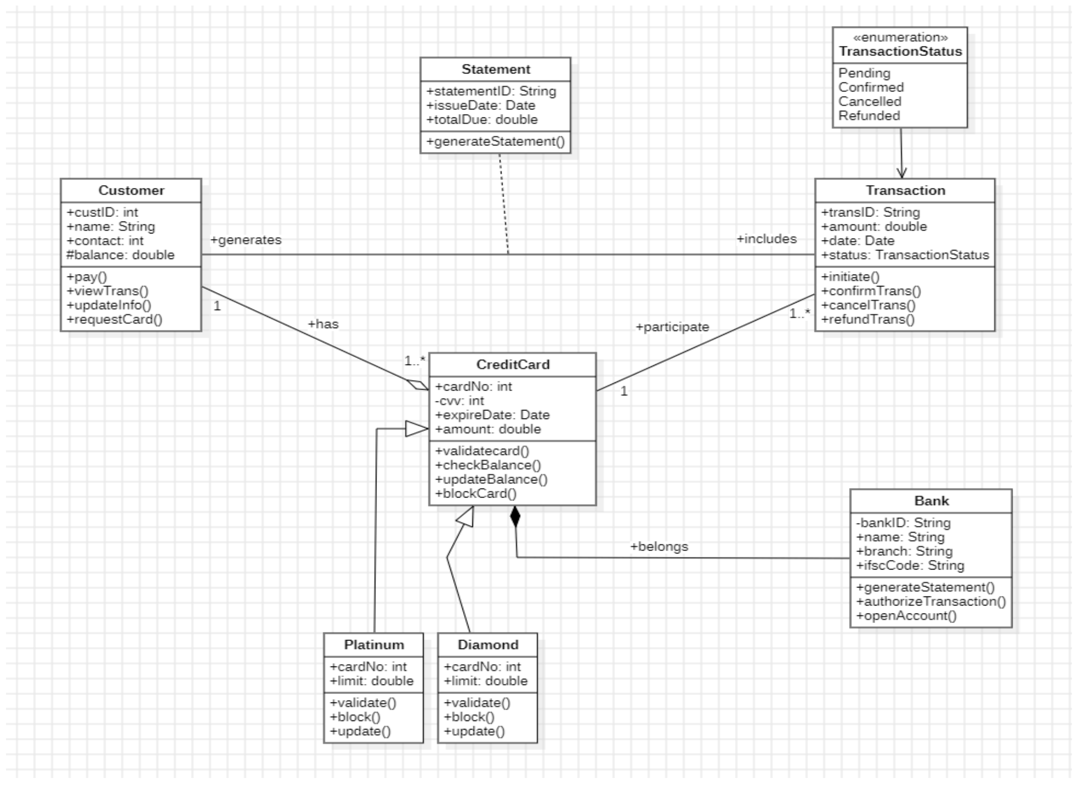


Fig 2.1: Class Diagram for Credit Card Processing System

The class diagram organizes all business objects involved in a credit card transaction, such as Card, Transaction, AuthorizationRequest, and Issuer. It shows how these elements depend on each other to carry out validation, authorization, fraud checking, and settlement. The relationships ensure sensitive financial data flows in a controlled manner. This structure creates a robust foundation for implementing secure transaction processing.

4) State Diagram

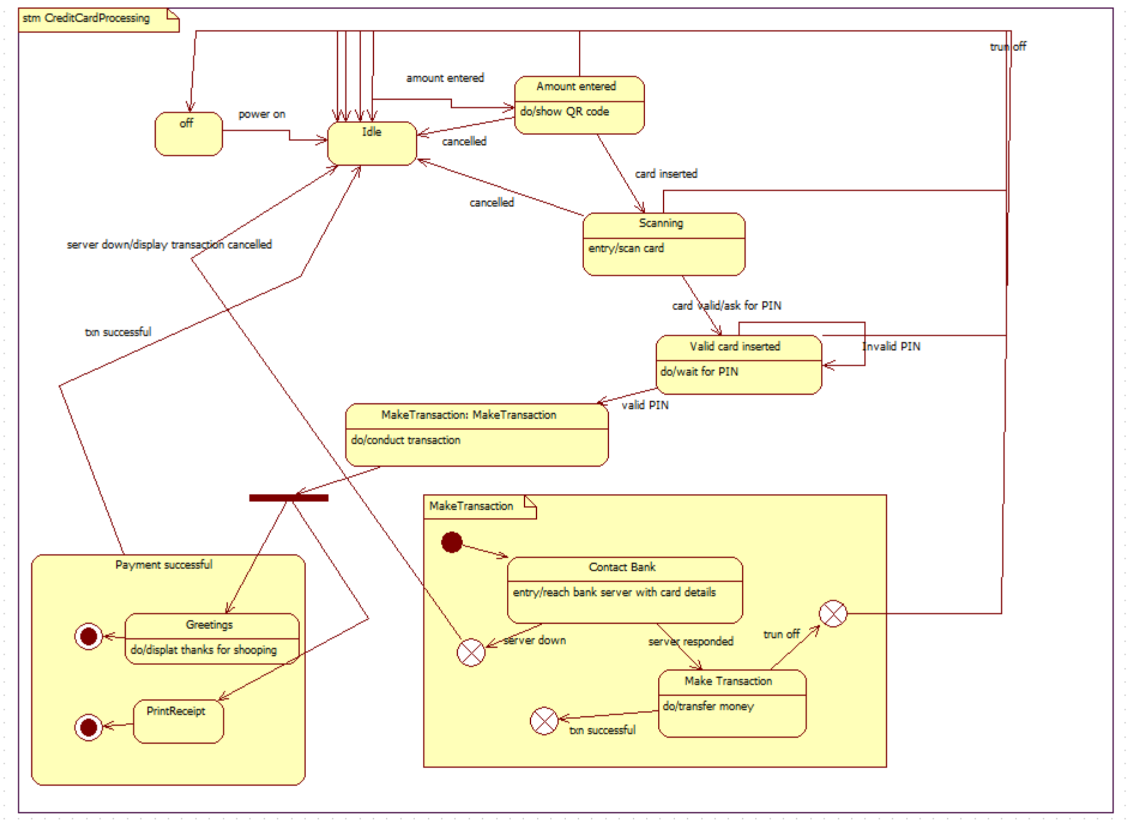


Fig 2.2: State Diagram for Credit Card Processing System

The state diagram captures every possible stage a transaction can pass through, from initiation to authorization, capture, settlement, or decline. It reflects real-world constraints such as fraud verification and issuer responses that affect the flow. By visualizing all permissible transitions, the diagram ensures the transaction engine behaves consistently under various conditions. It also highlights error states that the system must be able to recover from.

5) Use Case Diagram

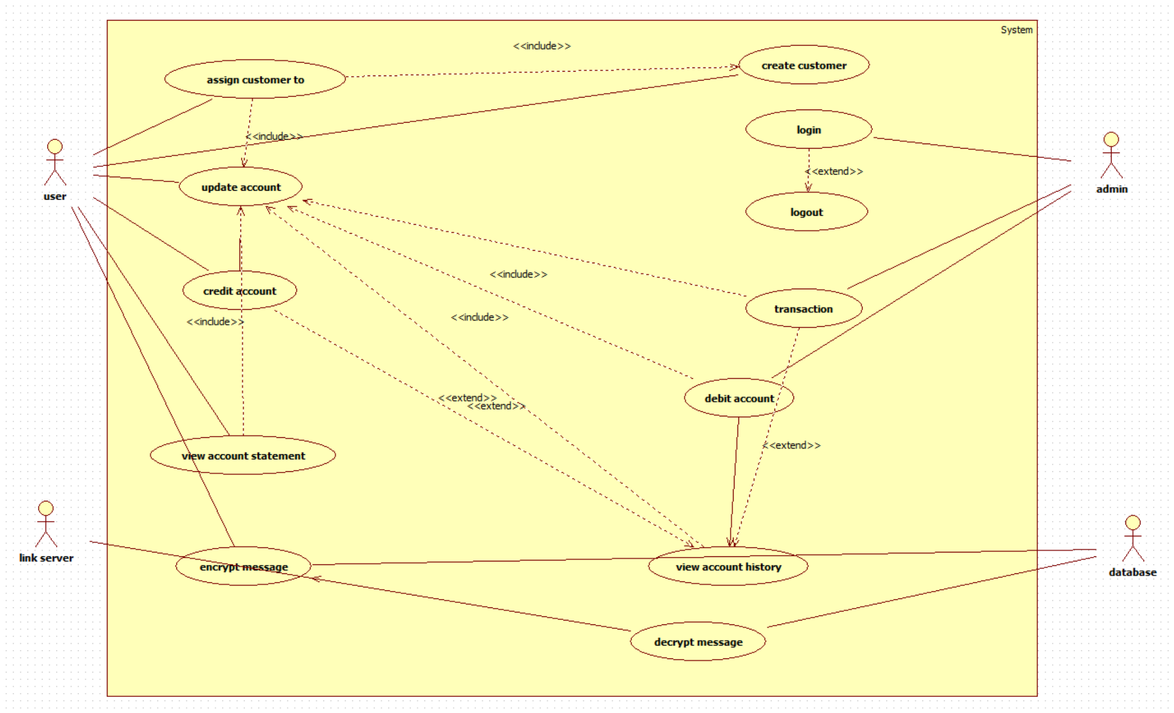


Fig 2.3: Use Case Diagram for Credit Card Processing System

The use case diagram explains how merchants, cardholders, processors, and banks interact with the payment system. It formalizes essential activities such as authorizing payments, handling chargebacks, and settling funds. Inclusion and extension relationships demonstrate shared logic (e.g., identity checks) and optional flows like 3-D Secure challenges. It provides a top-level view of what functionality the system must handle.

6) Sequence Diagram

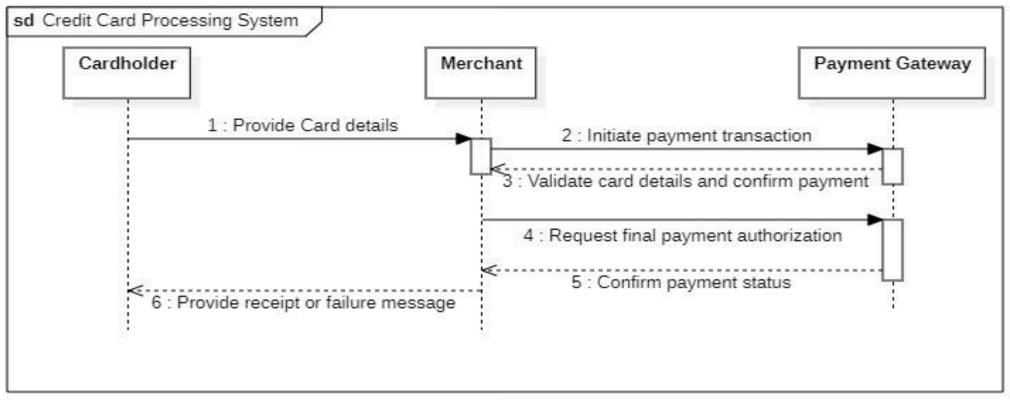

sd Credit Card Processing System
Cardholder
Merchant
Payment Gateway
1 : Provide Card details
2 : Initiate payment transaction
3 : Validate card details and confirm payment
4 : Request final payment authorization
5 : Confirm payment status
6 : Provide receipt or failure message

Fig 2.4: Sequence Diagram for Credit Card Processing System

The sequence diagram illustrates the chain of communication involved in processing a transaction—from merchant terminal to acquirer, network, and issuer. It clarifies how data flows, which validations occur first, and how responses are propagated back. The diagram helps identify time-critical steps and dependencies between services. This is crucial for verifying compliance and performance requirements.

7) Activity Diagram

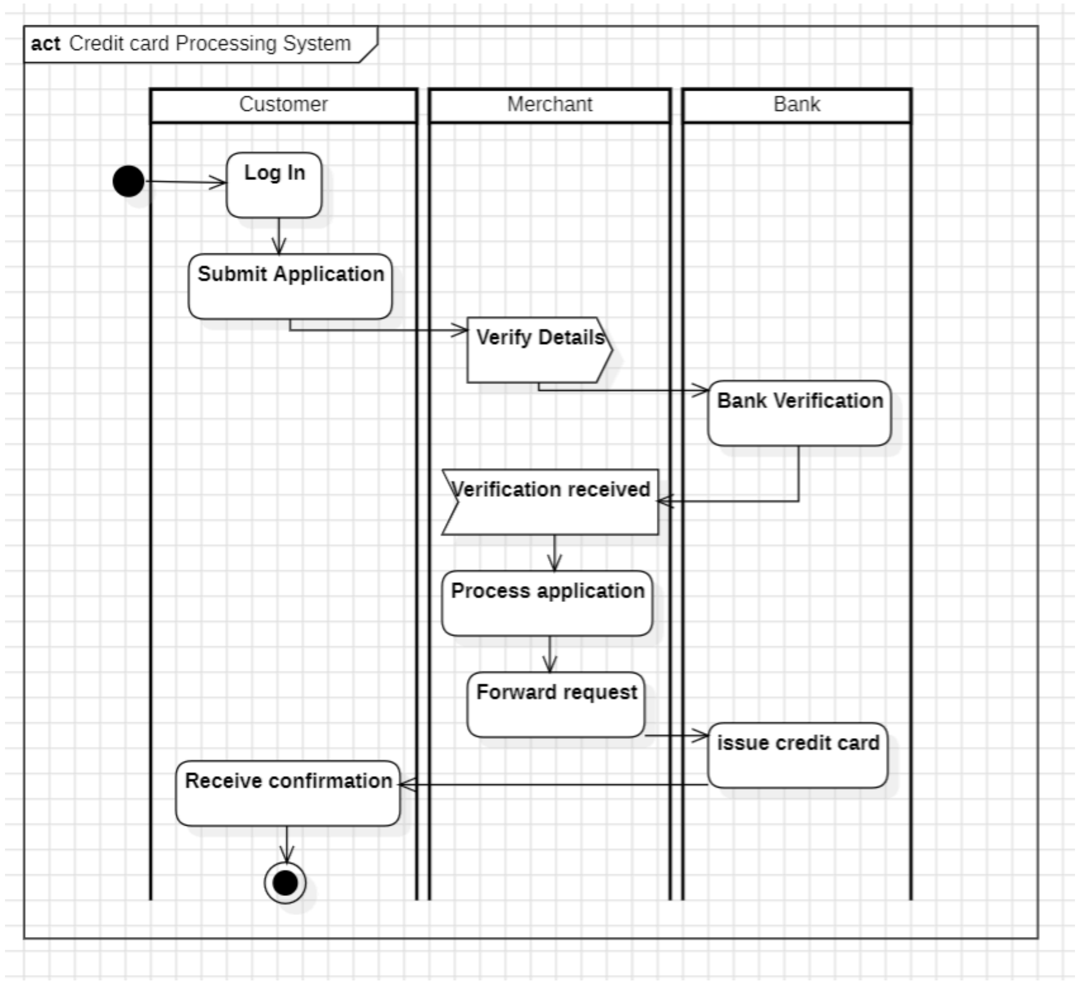


Fig 2.5: Activity Diagram for Credit Card Processing System

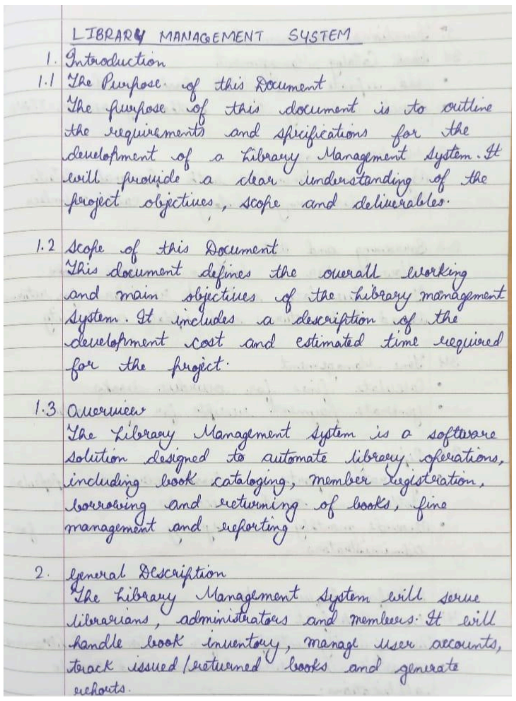
The activity diagram reflects the workflow of a credit card authorization, including fraud checks, issuer requests, and result evaluation. It maps out how the system decides whether a transaction is approved or declined. The diagram ensures that every branch of the process is logically covered. This is essential for validating financial decision logic.

3. Library Management System

1) Problem Statement

Libraries face challenges in manually tracking book inventory, managing member accounts, issuing and returning books, calculating fines, and maintaining accurate usage records. Manual processes often lead to misplaced books, delays in finding availability, and poor user experience. A computerized Library Management System is required to automate cataloging, borrowing, returns, fine management, and reporting, ensuring efficient library operations and improved accessibility for users.

2) SRS-Software Requirements Specification



LIBRARY MANAGEMENT SYSTEM

1. Introduction

1.1 The Purpose of this Document

The purpose of this document is to outline the requirements and specifications for the development of a Library Management System. It will provide a clear understanding of the project objectives, scope and deliverables.

1.2 Scope of this Document

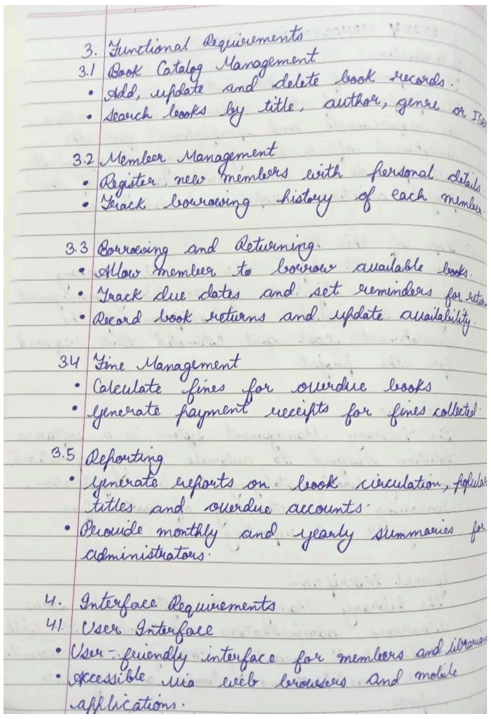
This document defines the overall working and main objectives of the library management System. It includes a description of the development cost and estimated time required for the project.

1.3 Overview

The Library Management System is a software solution designed to automate library operations, including book cataloging, member registration, borrowing and returning of books, fine management and reporting.

2. General Description

The Library Management System will serve librarians, administrators and members. It will handle book inventory, manage user accounts, track issued/returned books and generate reports.



3. Functional Requirements

3.1 Book Catalog Management
- Add, update and delete book records.
- Search books by title, author, genre or IS

3.2 Member Management
- Register new members with personal details
- Track borrowing history of each member.

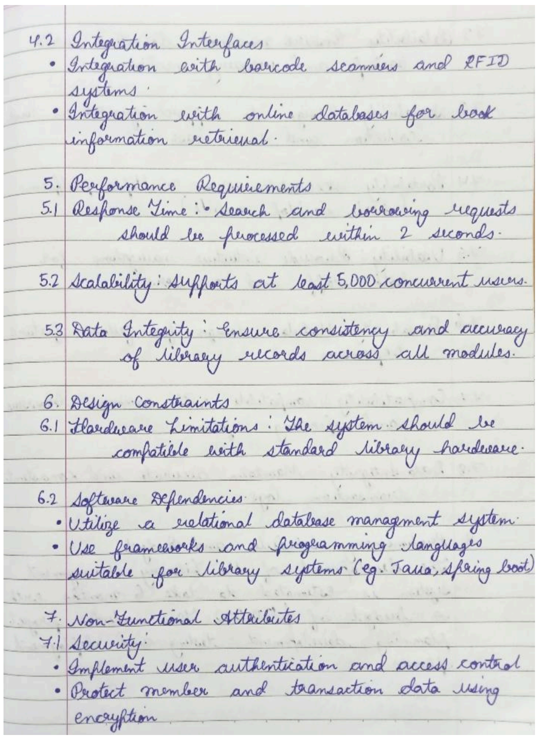
3.3 Borrowing and Returning
- Allow member to borrow available books.
- Track due dates and set reminders for retu
- Record book returns and update availability.

3.4 Fine Management
- Calculate fines for overdue books
- Generate payment receipts for fines collected.

3.5 Reporting
- Generate reports on book circulation, popula titles and overdue accounts.
- Provide monthly and yearly summaries for administrators.

4. Interface Requirements

4.1. User Interface
- User-friendly interface for members and librarian
- Accessible via web browsers and mobile applications.



4.2 Integration Interfaces
- Integration with barcode scanners and RFID systems.
- Integration with online databases for book information retrieval.

5. Performance Requirements
5.1 Response Time: • Search and borrowing requests should be processed within 2 seconds.

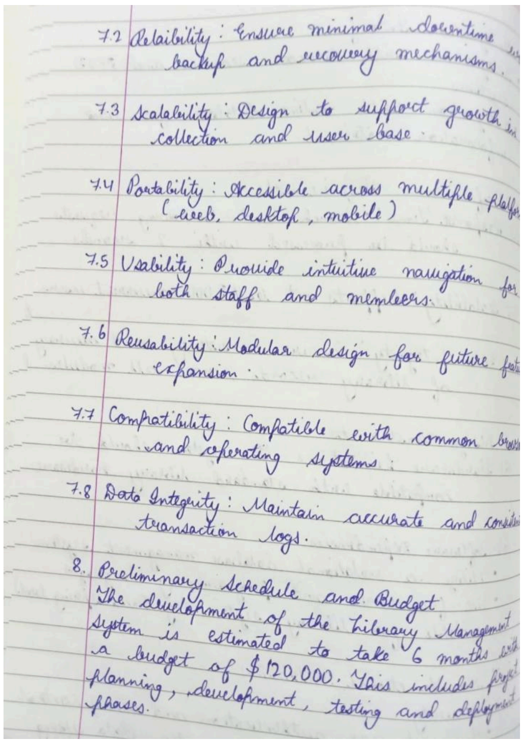
5.2 Scalability: Supports at least 5,000 concurrent users.

5.3 Data Integrity: Ensure consistency and accuracy of library records across all modules.

6. Design Constraints
6.1 Hardware Limitations: The system should be compatible with standard library hardware.

6.2 Software Dependencies
- Utilize a relational database management system.
- Use frameworks and programming languages suitable for library systems (eg. Java, spring boot)

7. Non-Functional Attributes
7.1 Security:
- Implement user authentication and access control
- Protect member and transaction data using encryption



7.2 Relaibility: Ensure minimal downtime w backup and recovery mechanisms.

7.3 Scalability: Design to support growth in collection and user base.

7.4 Portability: Accessible across multiple platfo (web, desktop, mobile)

7.5 Usability: Provide intuitive navigation for both staff and members.

7.6 Reusability: Modular design for future featu expansion.

7.7 Compatibility: Compatible with common brows and operating systems.

7.8 Data Integrity: Maintain accurate and consist transaction logs.

8. Preliminary Schedule and Budget
The development of the Library Management System is estimated to take 6 months with a budget of $120,000. This includes project planning, development, testing and deployment phases.

3) Class Diagram

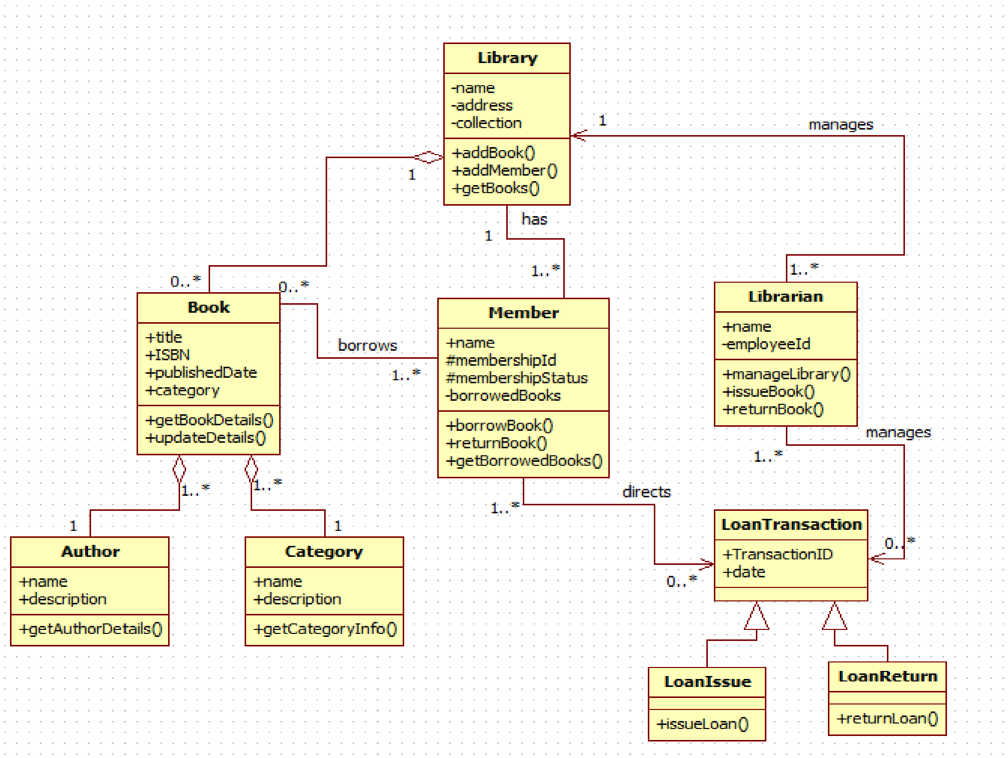


Fig 3.1: Class Diagram for Library Management System

The class diagram models key library entities such as Book, Copy, Member, Loan, and Fine, and represents how they interact to support circulation. It establishes the structural relationships necessary for tracking borrowing, reservations, and penalties. By defining these data structures clearly, the system can manage availability and maintain accurate lending records. This ensures the design supports long-term consistency and scalability.

4) State Diagram

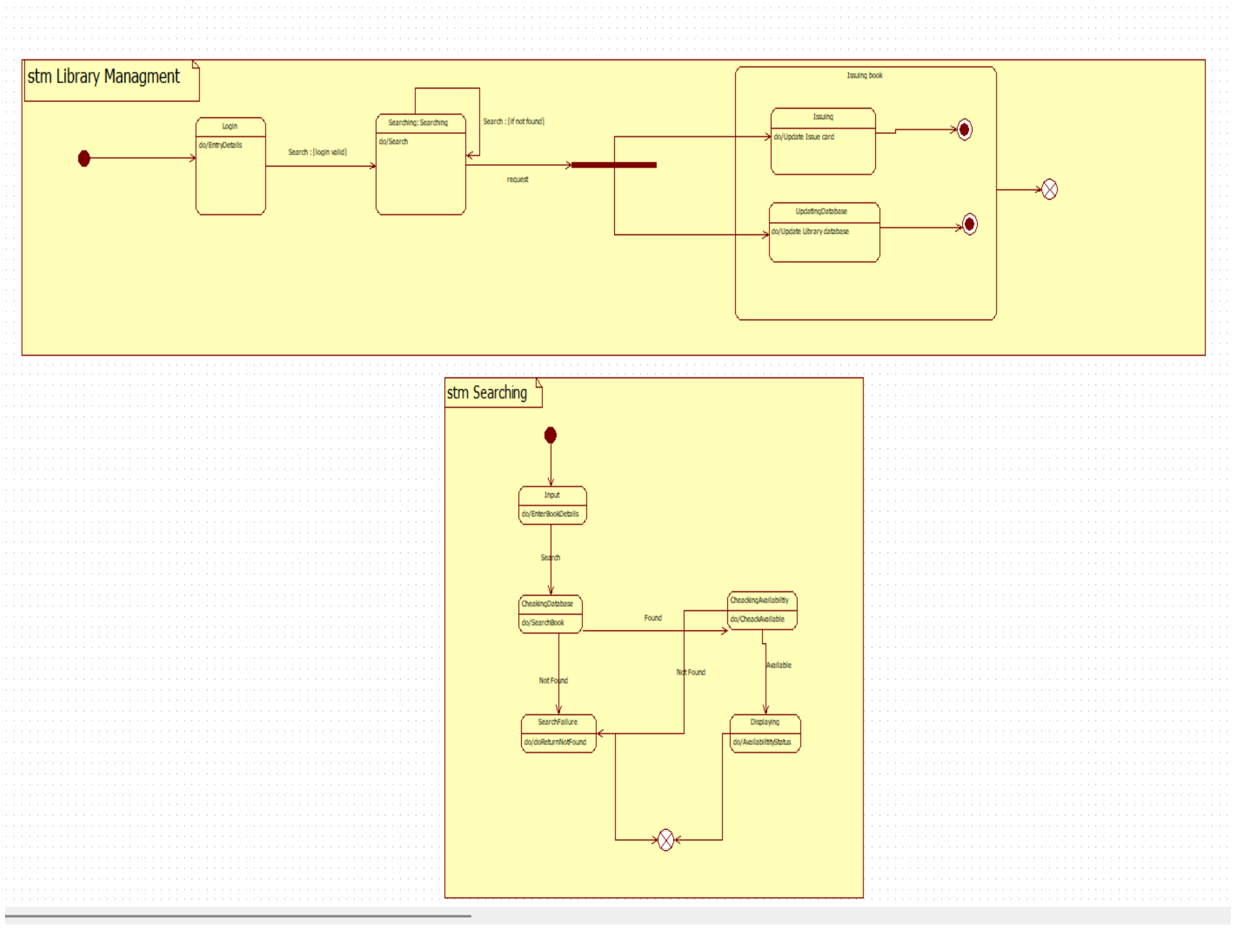


Fig 3.2: State Diagram for Library Management System

The state diagram illustrates how a library book copy transitions through states like available, reserved, checked-out, overdue, or lost. Each state change corresponds to specific member actions or system rules. The representation ensures that items cannot enter invalid states and tracks lifecycle behavior accurately. It is essential for proper inventory and circulation management.

5) Use Case Diagram

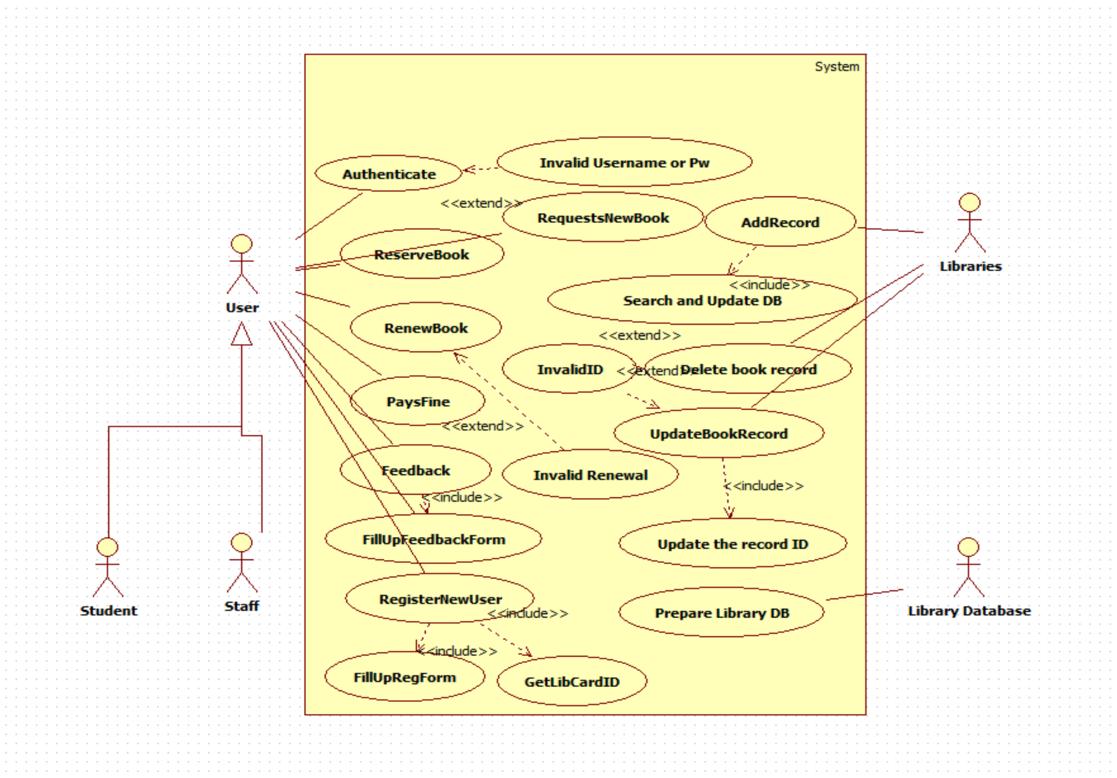


Fig 3.3: Use Case Diagram for Library Management System

The use-case diagram shows how librarians and members interact with the system to perform tasks such as searching books, issuing loans, returning items, and paying fines. It outlines functional expectations without exposing internal logic. The relationships clarify optional behaviors like calculating fines during returns. This gives a concise overview of all services the library must support

6) Sequence Diagram

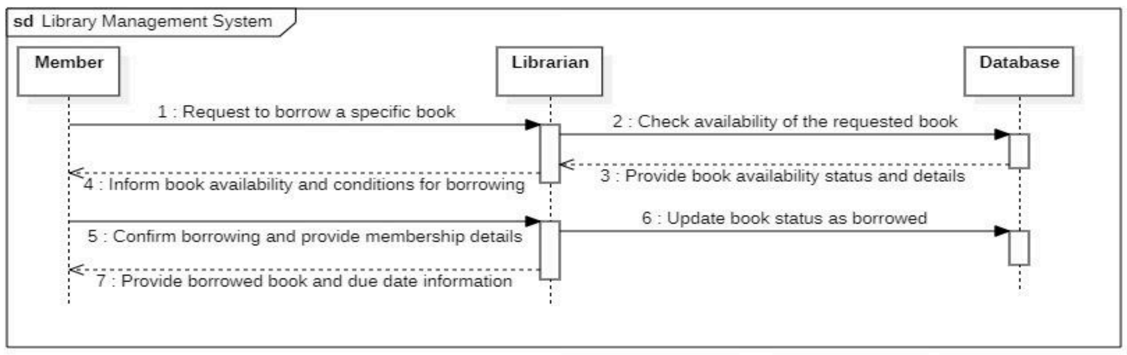

sd Library Management System
Member
Librarian
Database
1 : Request to borrow a specific book
2 : Check availability of the requested book
3 : Provide book availability status and details
4 : Inform book availability and conditions for borrowing
5 : Confirm borrowing and provide membership details
6 : Update book status as borrowed
7 : Provide borrowed book and due date information

Fig 3.4: Sequence Diagram for Library Management System

The sequence diagram captures the coordinated steps involved when a member borrows or returns a book. It outlines how the interface, catalog service, loan handler, and notification system communicate. This helps validate the correctness of transaction flow and ensures all necessary operations trigger in proper order. It is key for checking operational dependencies.

7) Activity Diagram

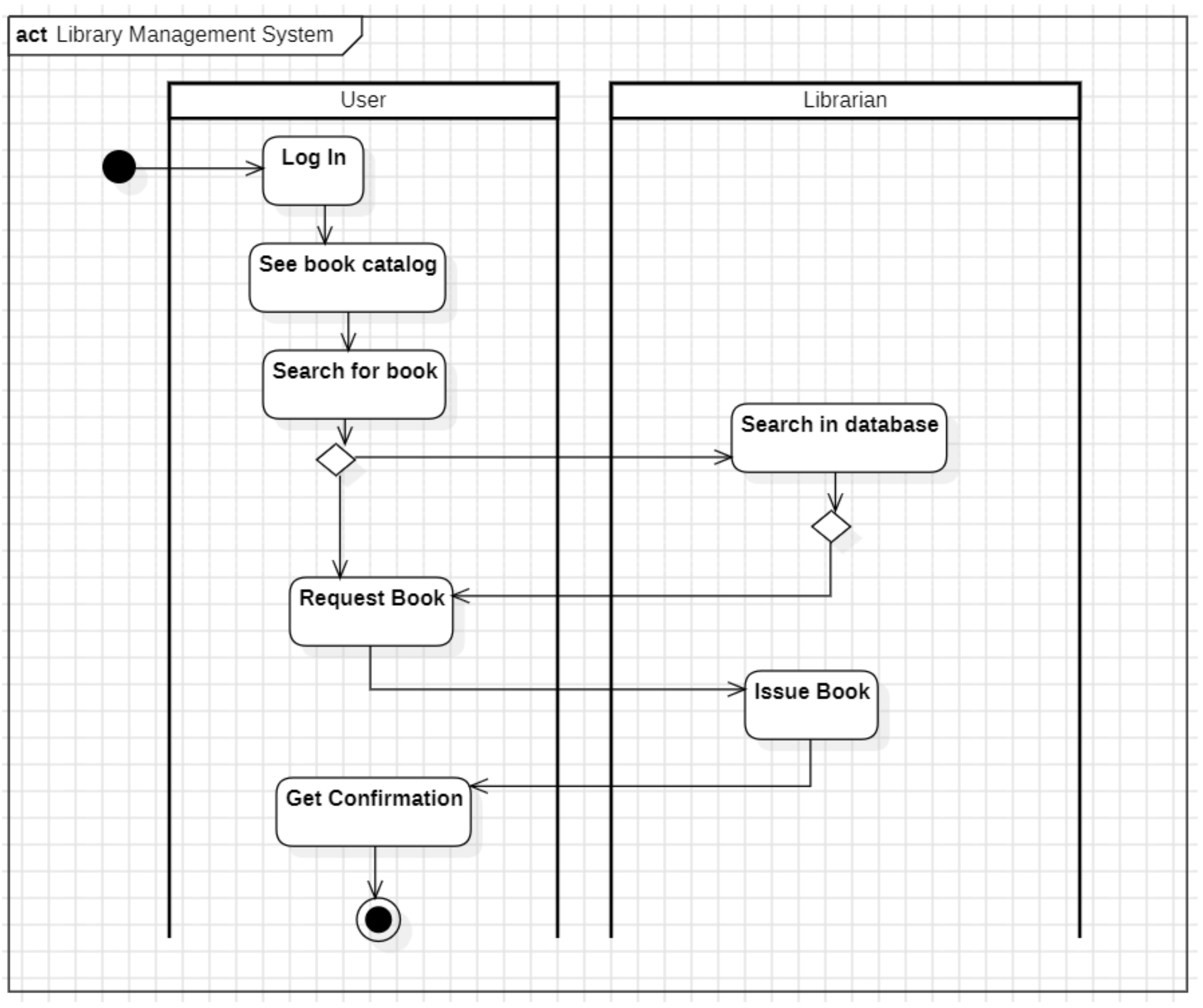


Fig 3.5: Activity Diagram for Library Management System

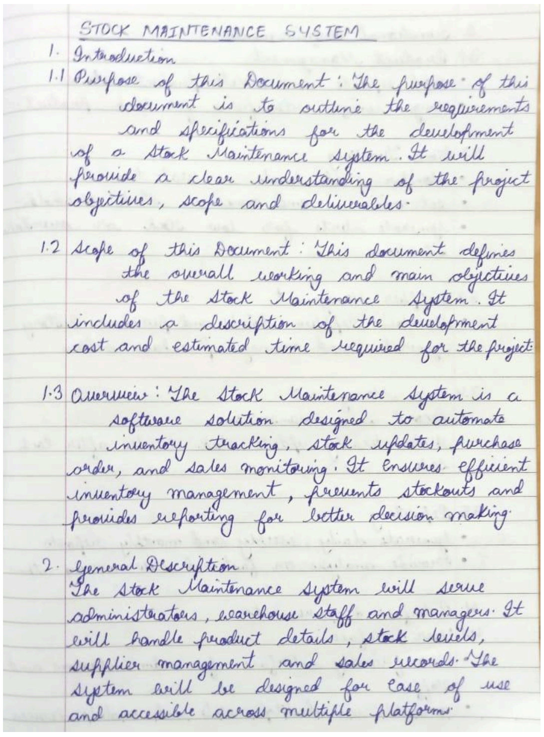
The activity diagram describes workflows like issuing or returning books, highlighting decisions such as fine applicability. It visualizes both simple and branching paths to ensure no incomplete scenarios exist. This helps document procedural logic that librarians follow. It improves clarity in how tasks progress from start to completion.

4. Stock Maintenance System

1) Problem Statement

Businesses relying on manual inventory tracking often encounter issues such as stock discrepancies, delays in restocking, overstocking, and inaccuracies in sales or purchase records. These problems lead to financial losses and poor decision-making. A Stock Maintenance System is required to automate stock tracking, supplier management, sales updates, and reporting to ensure real-time visibility, accuracy, and efficient inventory control.

2) SRS-Software Requirements Specification



STOCK MAINTENANCE SYSTEM

1. Introduction

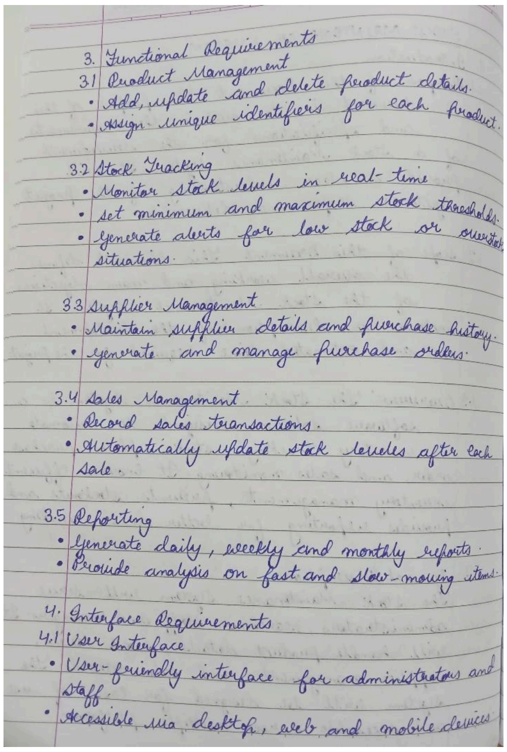
1.1 Purpose of this Document: The purpose of this document is to outline the requirements and specifications for the development of a Stock Maintenance System. It will provide a clear understanding of the project objectives, scope and deliverables.

1.2 Scope of this Document: This document defines the overall working and main objectives of the Stock Maintenance System. It includes a description of the development cost and estimated time required for the project.

1.3 Overview: The Stock Maintenance System is a software solution designed to automate inventory tracking, stock updates, purchase order, and sales monitoring. It ensures efficient inventory management, prevents stockouts and provides reporting for better decision making.

2. General Description

The Stock Maintenance System will serve administrators, warehouse staff and managers. It will handle product details, stock levels, supplier management and sales records. The system will be designed for ease of use and accessible across multiple platforms.



3. Functional Requirements

3.1 Product Management
- Add, update and delete product details.
- Assign unique identifiers for each product.

3.2 Stock Tracking
- Monitor stock levels in real-time
- set minimum and maximum stock thresholds.
- Generate alerts for low stock or overstock situations.

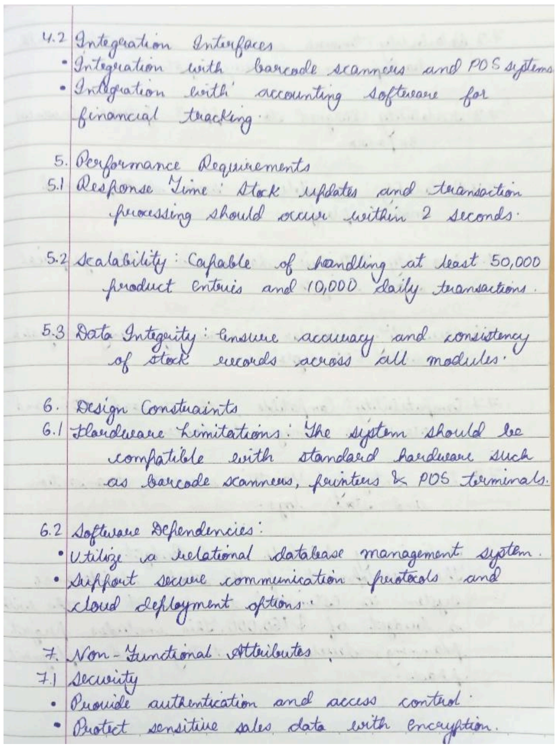
3.3 Supplier Management
- Maintain supplier details and purchase history.
- Generate and manage purchase orders.

3.4 Sales Management
- Record sales transactions.
- Automatically update stock leveles after each sale.

3.5 Reporting
- Generate daily, weekly and monthly reports.
- Provide analysis on fast and slow-moving items.

4. Interface Requirements

4.1 User Interface
- User-friendly interface for administrators and staff.
- Accessible via desktop, web and mobile devices



4.2 Integration Interfaces
- Integration with barcode scanners and POS systems
- Integration with accounting software for financial tracking.

5. Performance Requirements
5.1 Response Time: Stock updates and transaction processing should occur within 2 seconds.

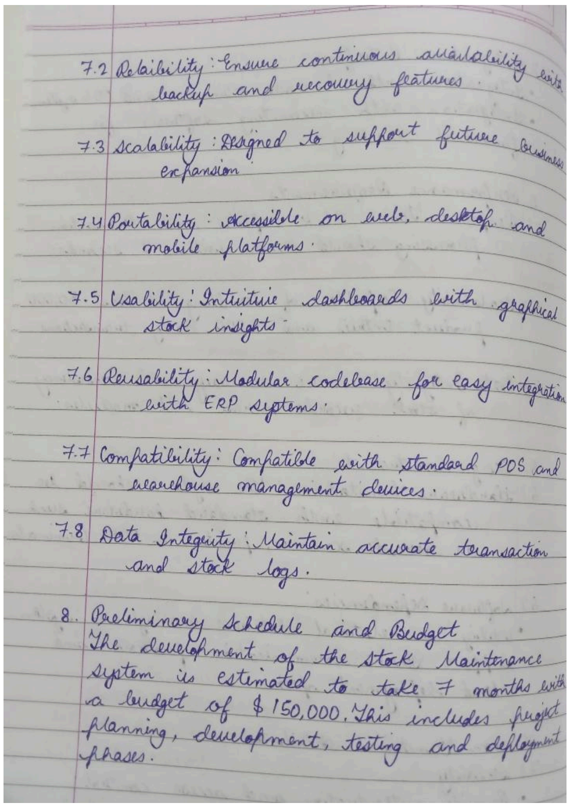
5.2 Scalability: Capable of handling at least 50,000 product entries and 10,000 daily transactions.

5.3 Data Integrity: Ensure accuracy and consistency of stock records across all modules.

6. Design Constraints
6.1 Hardware Limitations: The system should be compatible with standard hardware such as barcode scanners, printers & POS terminals.

6.2 Software Dependencies:
- Utilize a relational database management system.
- Support secure communication protocols and cloud deployment options.

7. Non-Functional Attributes
7.1 Security
- Provide authentication and access control.
- Protect sensitive sales data with encryption.



7.2 Relaibility: Ensure continuous availability with backup and recovery features.

7.3 Scalability: Designed to support future business expansion

7.4 Portability: Accessible on web, desktop and mobile platforms.

7.5 Usability: Intuitive dashboards with graphical stock insights.

7.6 Reusability: Modular codebase for easy integration with ERP systems.

7.7 Compatibility: Compatible with standard POS and warehouse management devices.

7.8 Data Integrity: Maintain accurate transaction and stock logs.

8. Preliminary Schedule and Budget
The development of the Stock Maintenance System is estimated to take 7 months with a budget of $150,000. This includes project planning, development, testing and deployment phases.

3) Class Diagram

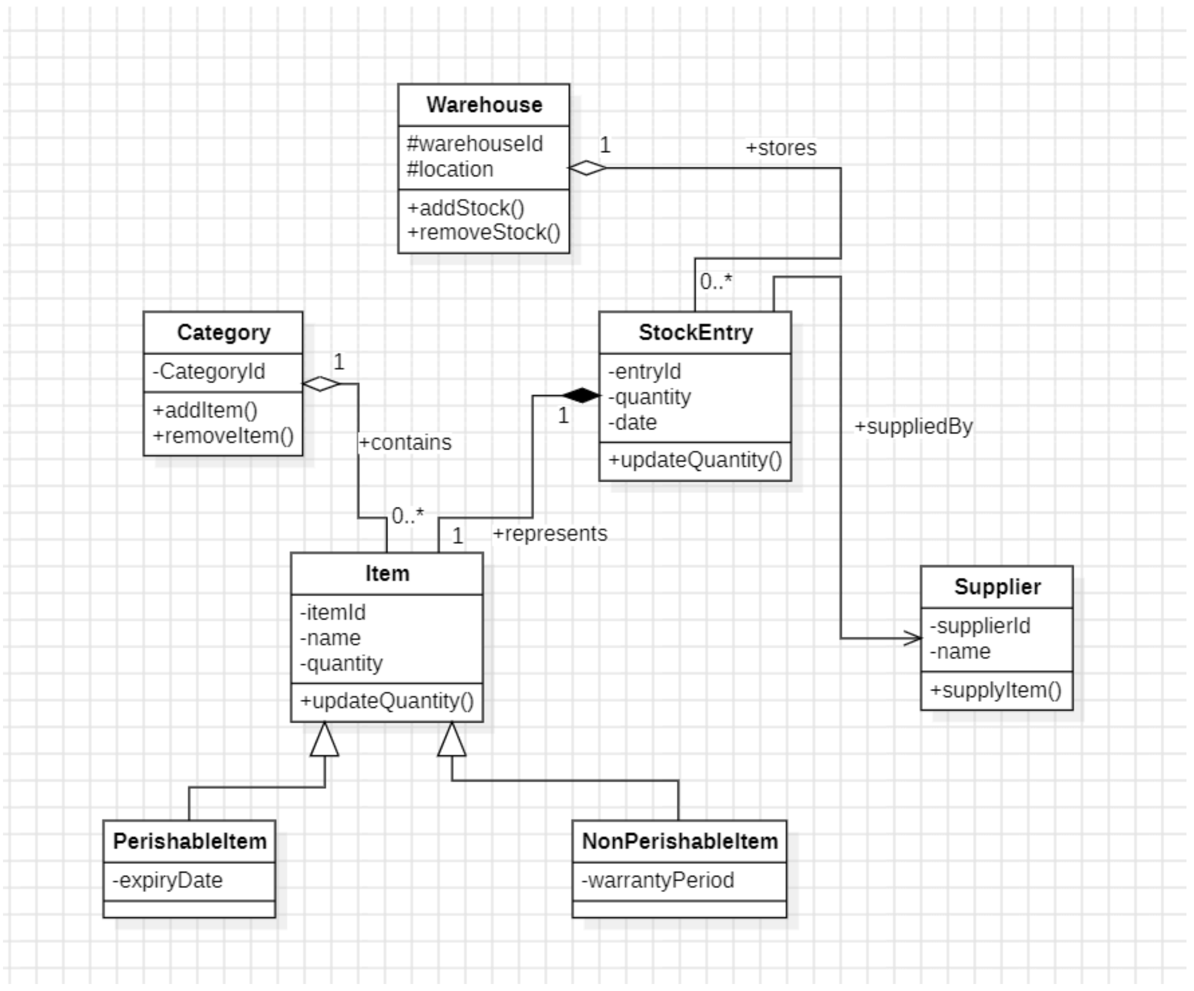


Fig 4.1: Class Diagram for Stock Maintenance System

The class diagram defines the structure of products, stock levels, suppliers, warehouses, and orders. It organizes how these components relate to track inventory accurately. The relationships determine how stock is added, consumed, or reordered. This serves as the foundation for managing stock consistency across operations.

4) State Diagram

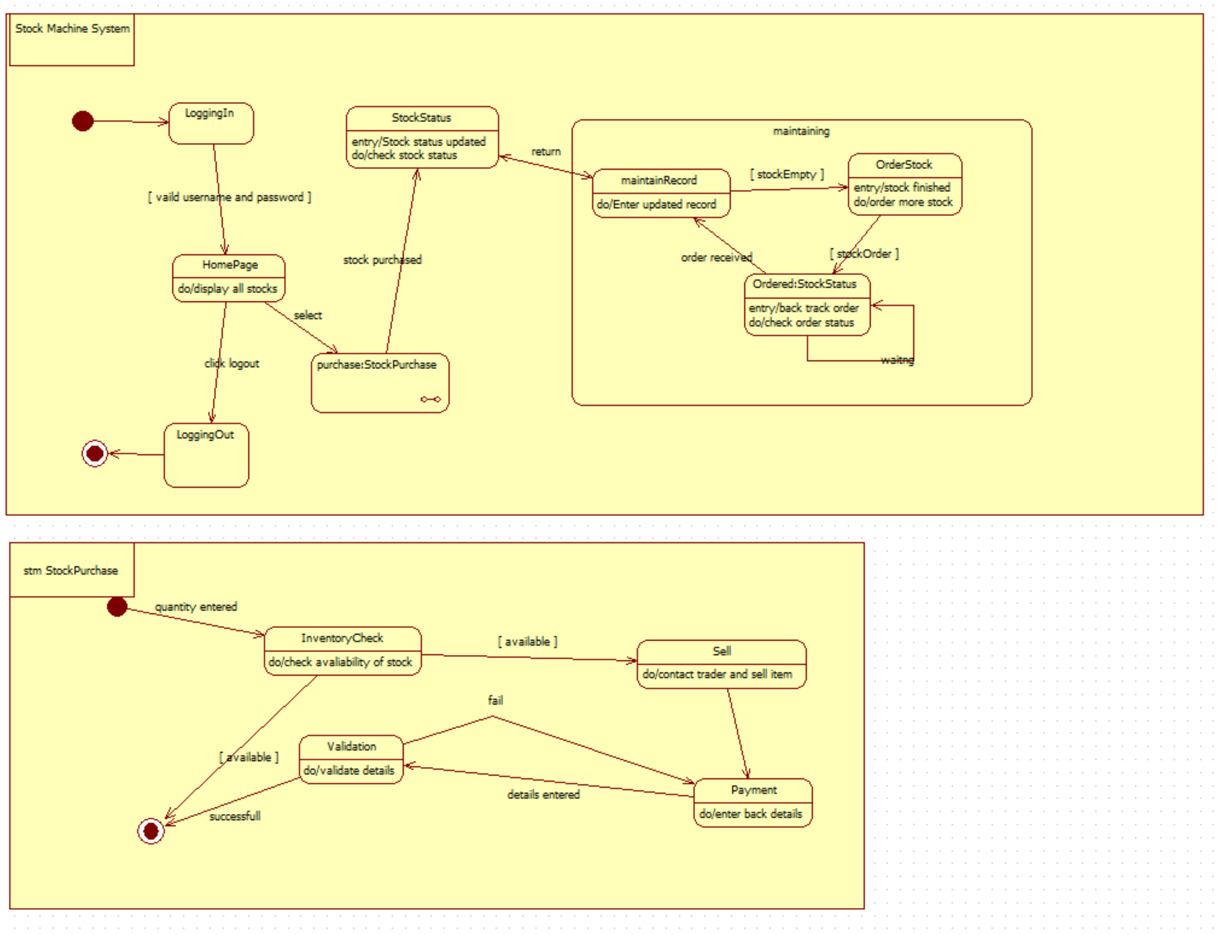


Fig 4.2: State Diagram for Stock Maintenance System

The state diagram shows how a purchase order or stock item progresses through states like pending, ordered, received, processed, or expired. Each transition depends on supplier actions, stock audits, or internal validations. Modeling these states ensures that stock is always accounted for accurately. It helps prevent inconsistencies or missing stock updates.

5) Use Case Diagram

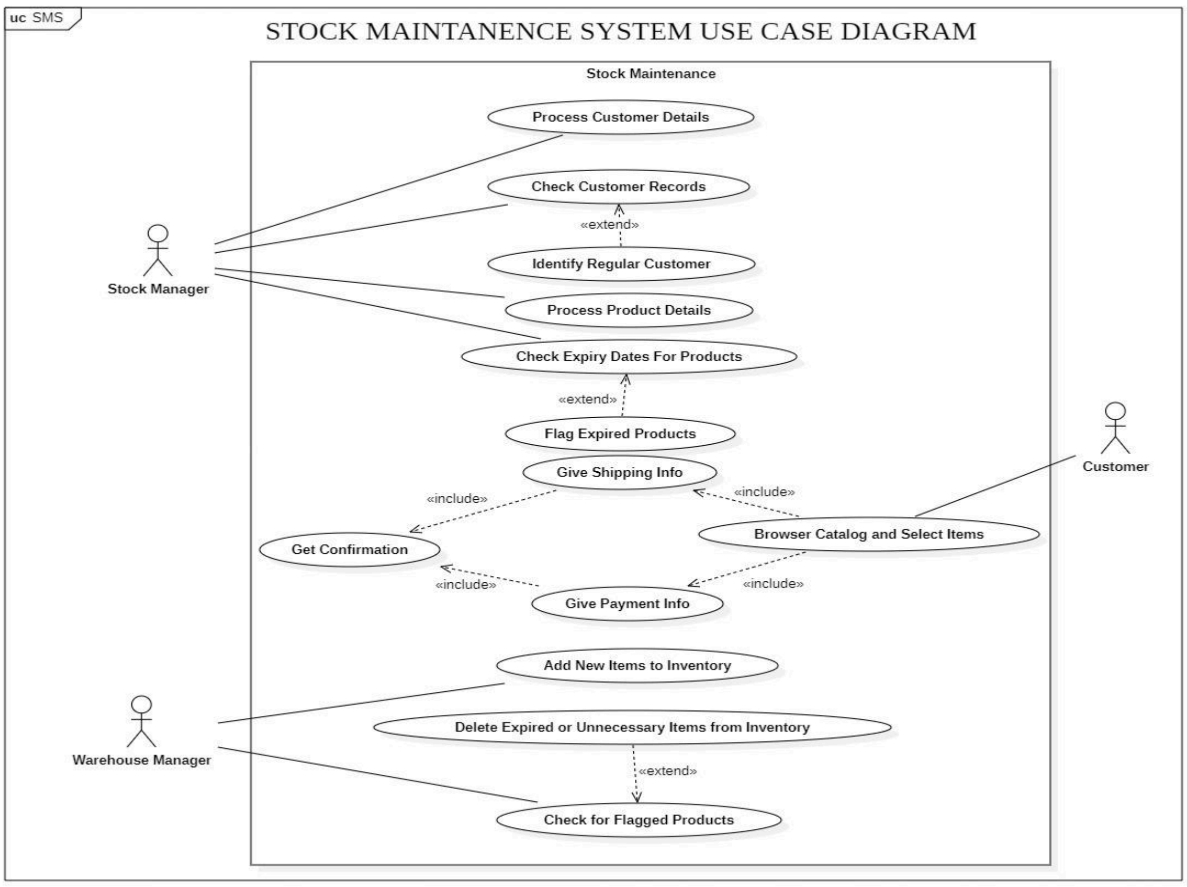
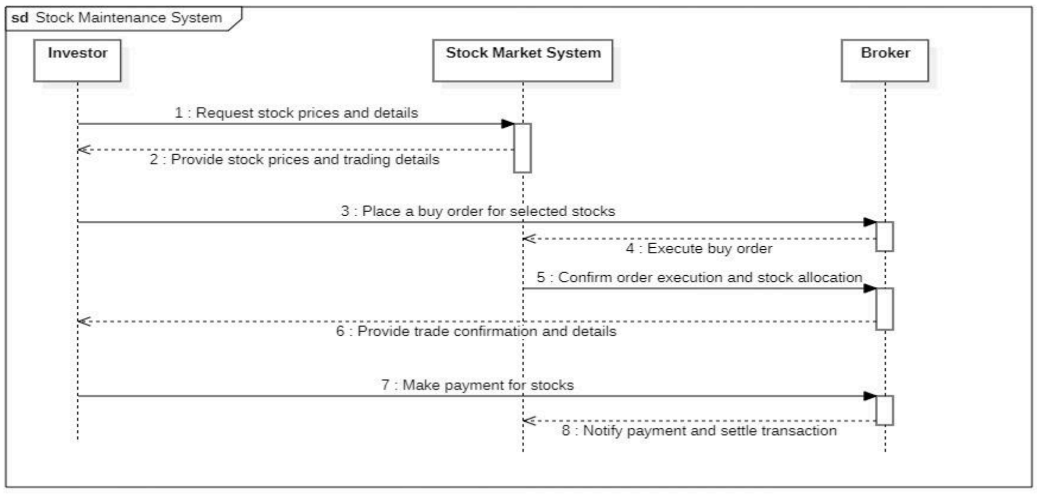


Fig 4.3: Use Cases Diagram for Stock Maintenance System

The use case diagram captures how warehouse staff, purchasers, and suppliers interact with the system. It outlines activities like creating POs, receiving shipments, adjusting stock, and reviewing reports. Include/extend relationships identify reusable workflows such as low-stock alerts. It clarifies the function boundaries for inventory control.

6) Sequence Diagram



sd Stock Maintenance System
Investor
Stock Market System
Broker
1 : Request stock prices and details
2 : Provide stock prices and trading details
3 : Place a buy order for selected stocks
4 : Execute buy order
5 : Confirm order execution and stock allocation
6 : Provide trade confirmation and details
7 : Make payment for stocks
8 : Notify payment and settle transaction

Fig 4.4: Sequence Diagram for Stock Maintenance System

The sequence diagram outlines communication flow during stock updates, such as auto-reorder or goods receipt. It shows how monitoring services, repositories, supplier APIs, and warehouses interact in order. This identifies bottlenecks and verifies that stock adjustments occur correctly. It validates that operational logic is correctly sequenced.

7) Activity Diagram

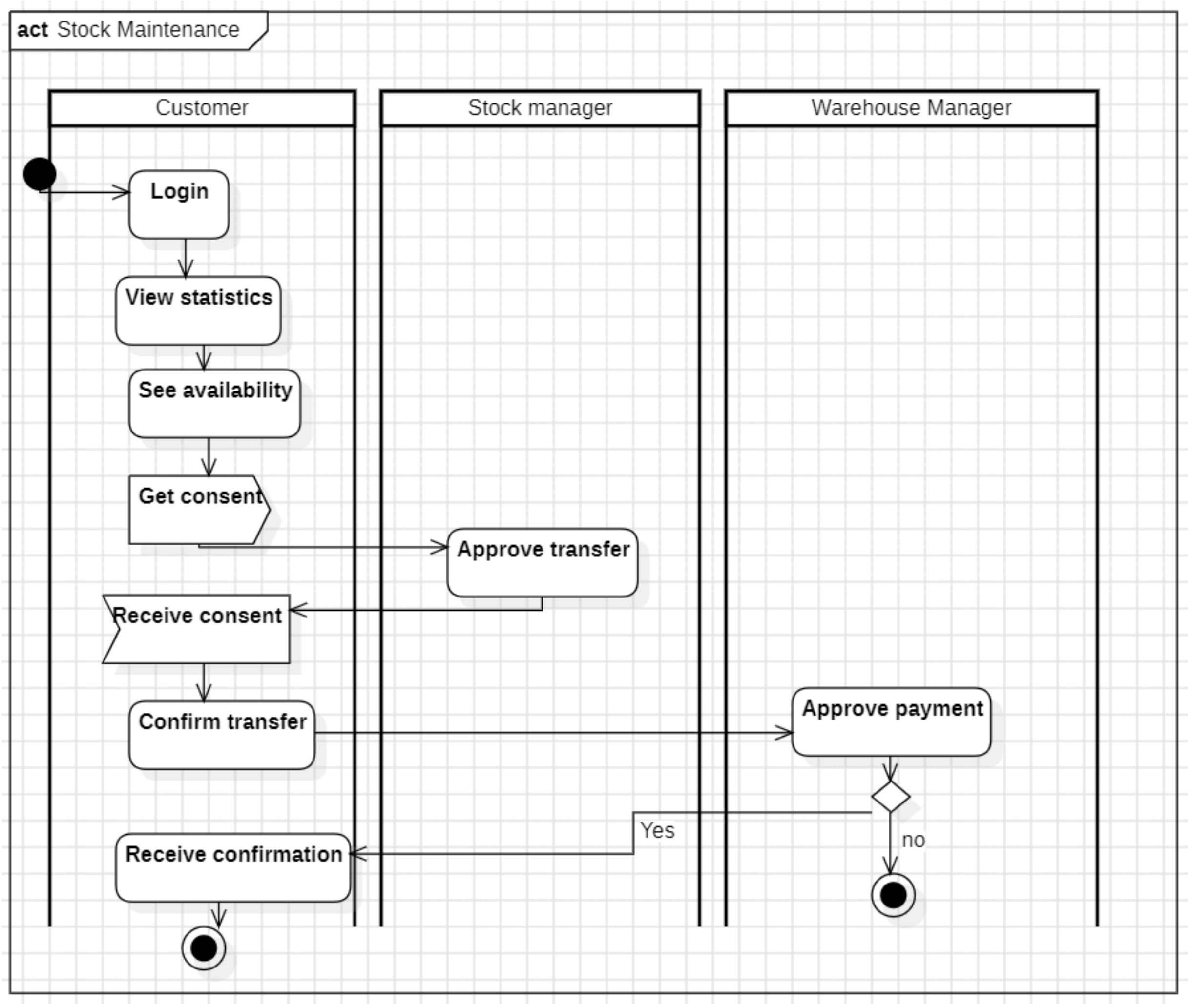


Fig 4.5: Activity Diagram for Stock Maintenance System

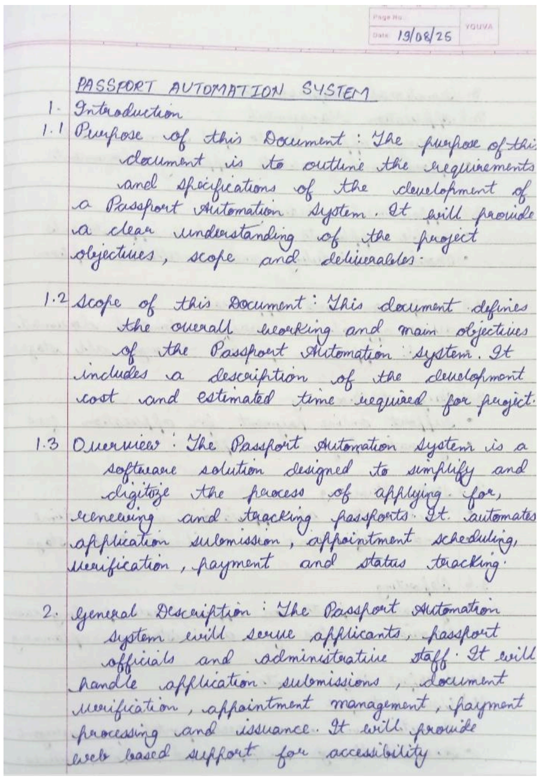
The activity diagram maps out workflows like stock receiving or dispatch. It highlights decision points such as quality checks, shortages, or rejections. This ensures the process is well-defined and supports all practical scenarios. It provides a clear understanding of how inventory tasks progress from start to finish.

5. Passport Automation System

1) Problem Statement

The traditional passport application and issuance process often suffers from long queues, manual paperwork, delays in verification, and lack of transparency for applicants. Tracking application status manually is time-consuming and inefficient for both applicants and government officials. A Passport Automation System is required to digitize application submission, verification, payment, scheduling, and status tracking to ensure faster processing, transparency, and improved public service delivery.

2) SRS-Software Requirements Specification



Page No.
Date: 19/08/25
YOUVA

PASSPORT AUTOMATION SYSTEM

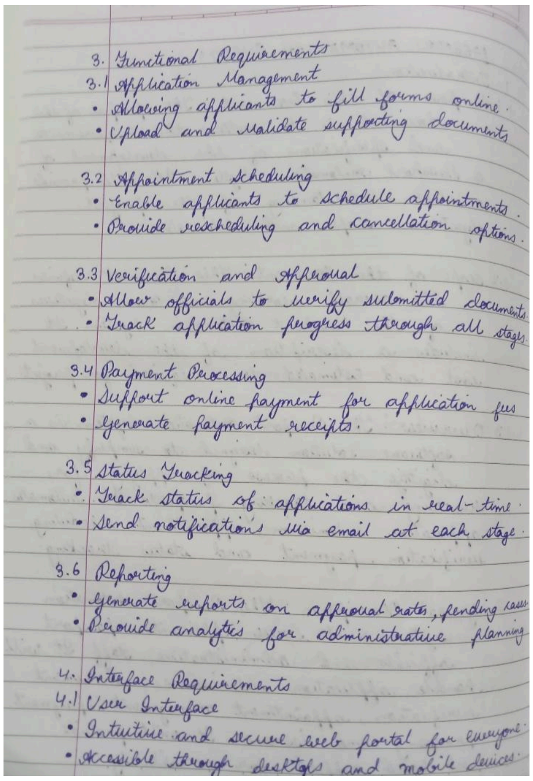
1. Introduction

1.1 Purpose of this Document : The purpose of this document is to outline the requirements and specifications of the development of a Passport Automation System. It will provide a clear understanding of the project objectives, scope and deliverables.

1.2 Scope of this Document : This document defines the overall working and main objectives of the Passport Automation system. It includes a description of the development cost and estimated time required for project.

1.3 Overview : The Passport Automation System is a software solution designed to simplify and digitize the process of applying for, renewing and tracking passports. It automates application submission, appointment scheduling, verification, payment and status tracking.

2. General Description : The Passport Automation system will serve applicants, passport officials and administrative staff. It will handle application submissions, document verification, appointment management, payment processing and issuance. It will provide web based support for accessibility.



3. Functional Requirements

3.1 Application Management
- Allowing applicants to fill forms online.
- Upload and validate supporting documents

3.2 Appointment Scheduling
- Enable applicants to schedule appointments.
- Provide rescheduling and cancellation options.

3.3 Verification and Approval
- Allow officials to verify submitted documents.
- Track application progress through all stages.

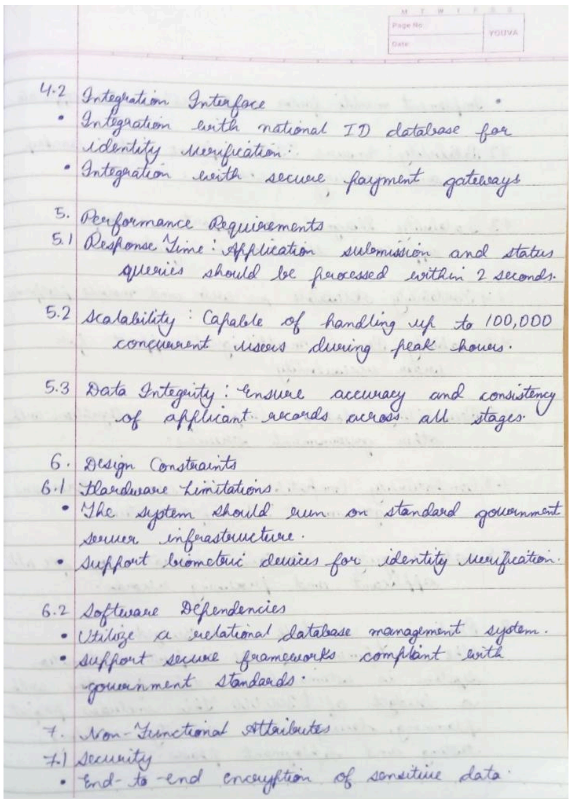
3.4 Payment Processing
- Support online payment for application fees
- Generate payment receipts.

3.5 Status Tracking
- Track status of applications in real-time.
- Send notifications via email at each stage.

3.6 Reporting
- Generate reports on approval rates, pending cases
- Provide analytics for administrative planning

4. Interface Requirements

4.1 User Interface
- Intuitive and secure web portal for everyone.
- Accessible through desktops and mobile devices.



4.2 Integration Interface
- Integration with national ID database for identity verification.
- Integration with secure payment gateways

5. Performance Requirements
5.1 Response Time: Application submission and status queries should be processed within 2 seconds.

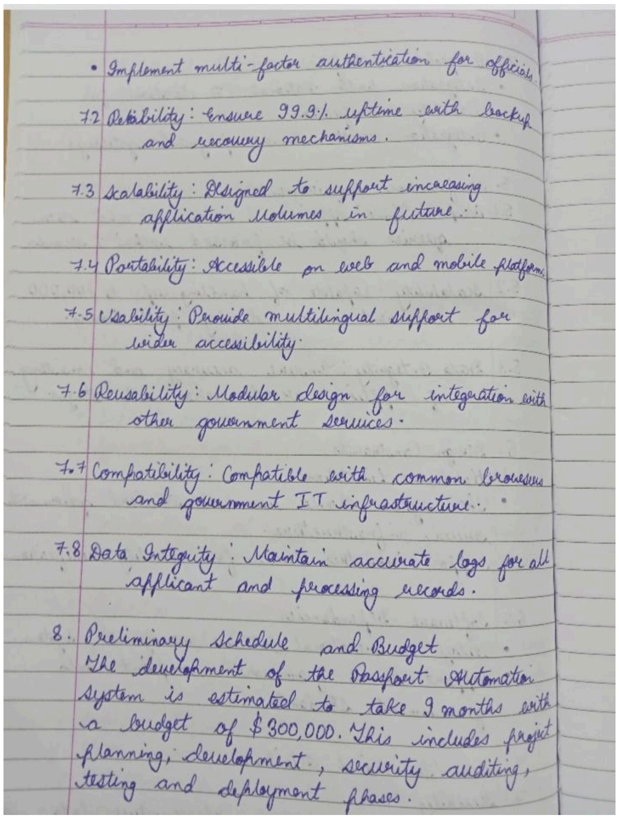
5.2 Scalability: Capable of handling up to 100,000 concurrent users during peak hours.

5.3 Data Integrity: Ensure accuracy and consistency of applicant records across all stages.

6. Design Constraints
6.1 Hardware Limitations.
- The system should run on standard government server infrastructure.
- Support biometric devices for identity verification.

6.2 Software Dependencies
- Utilize a relational database management system.
- Support secure frameworks compliant with government standards.

7. Non-Functional Attributes
7.1 Security
- End-to-end encryption of sensitive data.



- Implement multi-factor authentication for officials.

7.2 Reliability: Ensure 99.9% uptime with backup and recovery mechanisms.

7.3 Scalability: Designed to support increasing application volumes in future.

7.4 Portability: Accessible on web and mobile platforms.

7.5 Usability: Provide multilingual support for wider accessibility.

7.6 Reusability: Modular design for integration with other government services.

7.7 Compatibility: Compatible with common browsers and government IT infrastructure.

7.8 Data Integrity: Maintain accurate logs for all applicant and processing records.

8. Preliminary Schedule and Budget
The development of the Passport Automation System is estimated to take 9 months with a budget of $300,000. This includes project planning, development, security auditing, testing and deployment phases.

3) Class Diagram

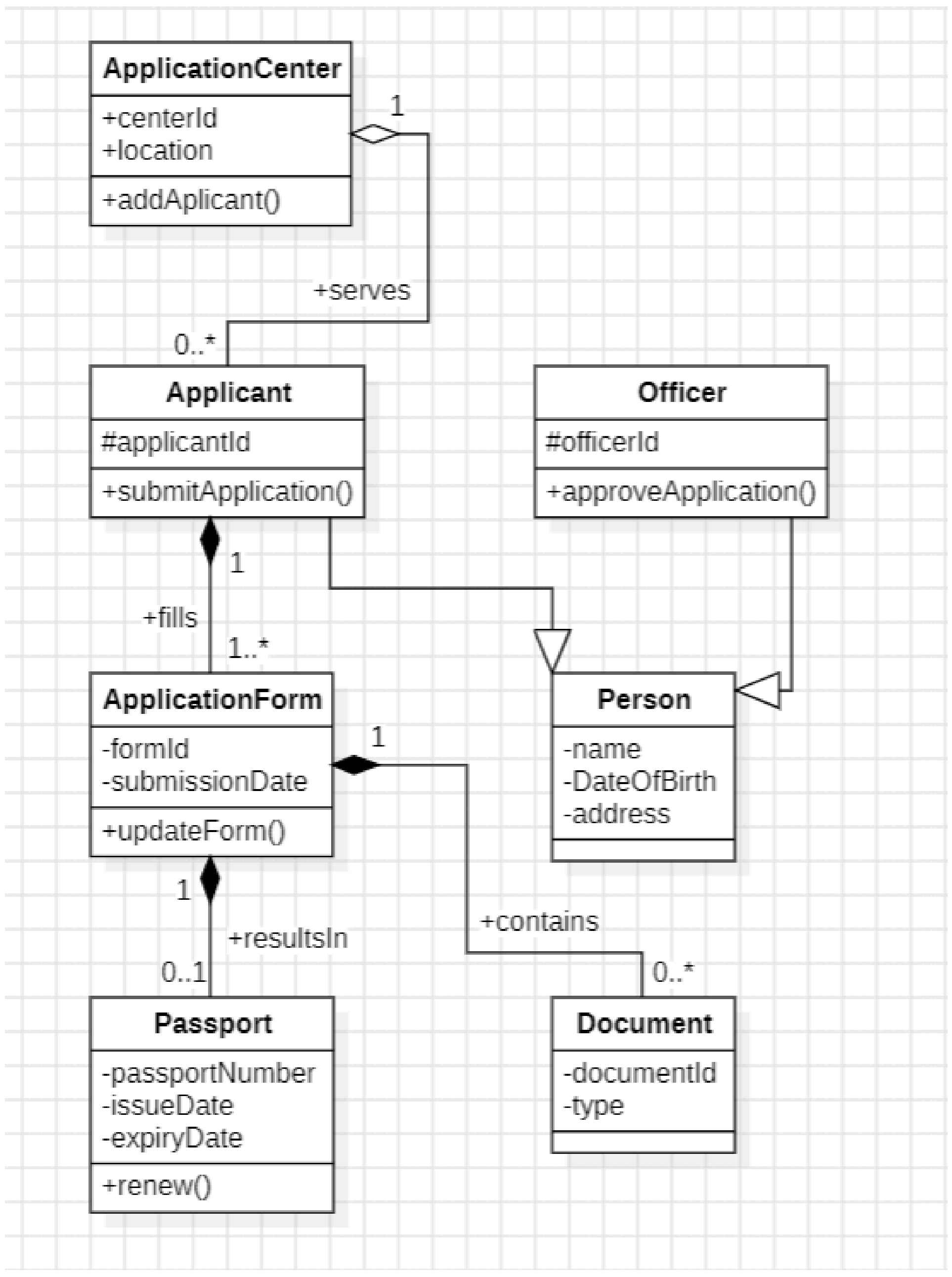


Fig 5.1: Class Diagram for Passport Automation System

The class diagram organizes applicants, applications, documents, officers, verification units, and appointments. It shows how these entities interconnect to support application submission, verification, and issuance. The structure ensures data consistency throughout the passport process. It forms the architectural core for all system operations.

4) State Diagram

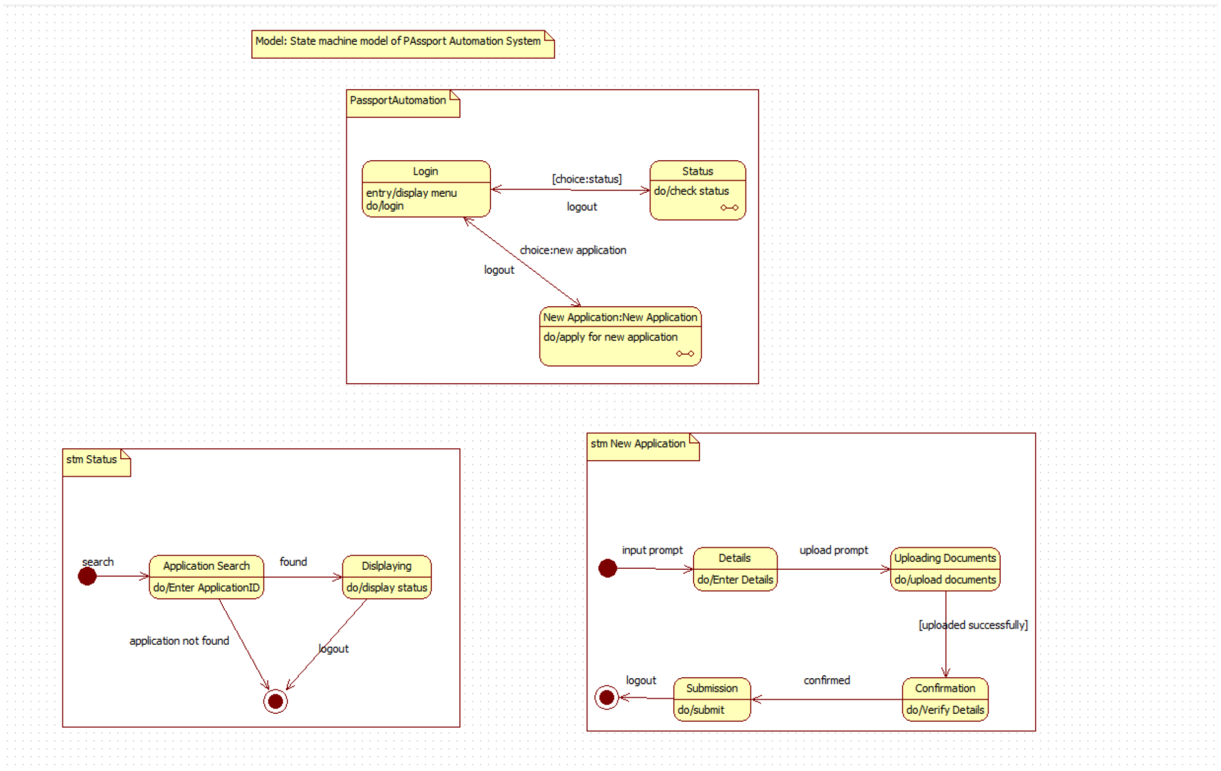


Fig 5.2: State Diagram for Stock Maintenance System

This diagram captures how a passport application moves from submission through verification, approval, printing, dispatch, and delivery. Each transition is triggered by events like document validation or police verification. It ensures no invalid state jumps occur and highlights exceptional cases like rejection. This clarifies the entire lifecycle of an application.

5) Use Case Diagram

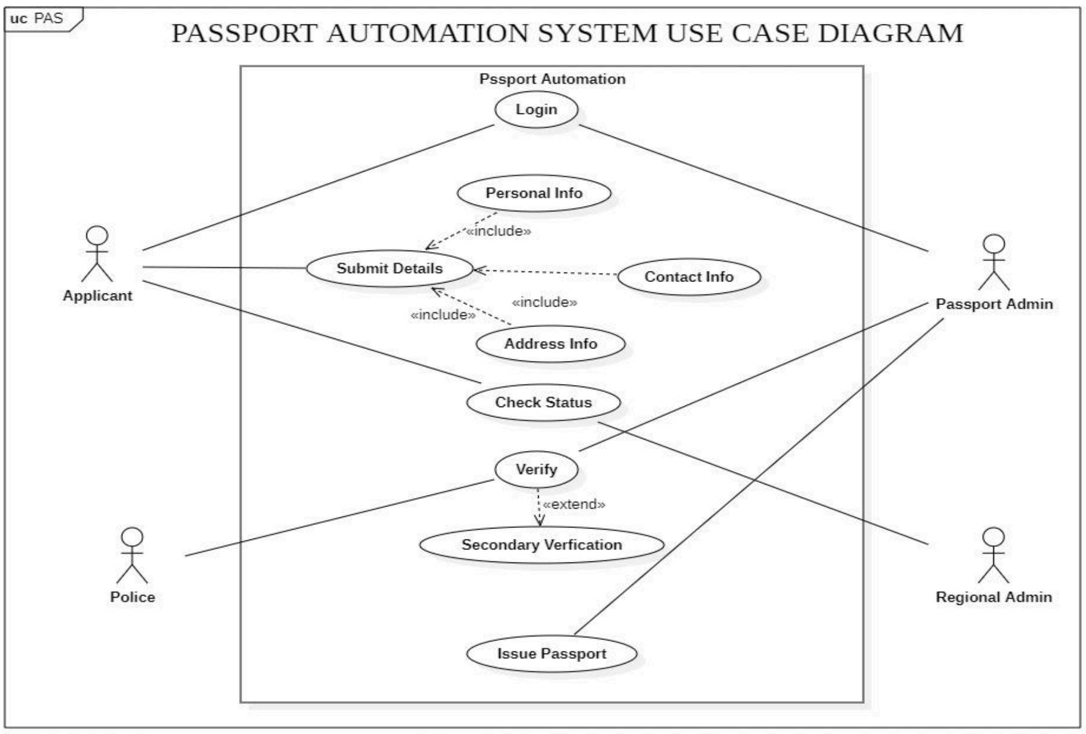


Fig 5.3: Use Case Diagram for Stock Maintenance System

The use-case diagram shows how applicants, officers, verification agencies, and payment gateways interact with the system. It maps essential actions like submitting forms, scheduling appointments, uploading documents, paying fees, and tracking status. The diagram uses include/extend to show dependencies and optional flows. It provides a clear summary of system functionality.

6) Sequence Diagram

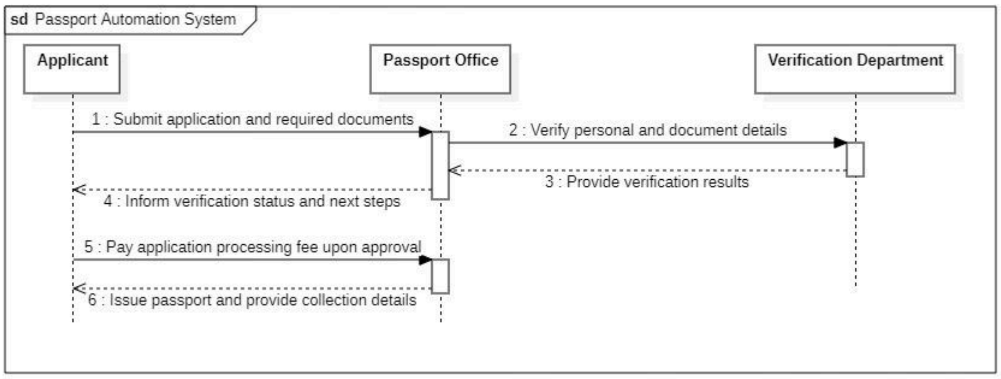

sd Passport Automation System
Applicant
Passport Office
Verification Department
1 : Submit application and required documents
2 : Verify personal and document details
3 : Provide verification results
4 : Inform verification status and next steps
5 : Pay application processing fee upon approval
6 : Issue passport and provide collection details

Fig 5.4: Sequence Diagram for Stock Maintenance System

This diagram visualizes the communication sequence when an applicant submits a passport application. It shows how the portal, payment service, verification system, printing unit, and courier interact step-by-step. This helps understand the real-time workflow of passport processing. It ensures no required operation is missed.

7) Activity Diagram

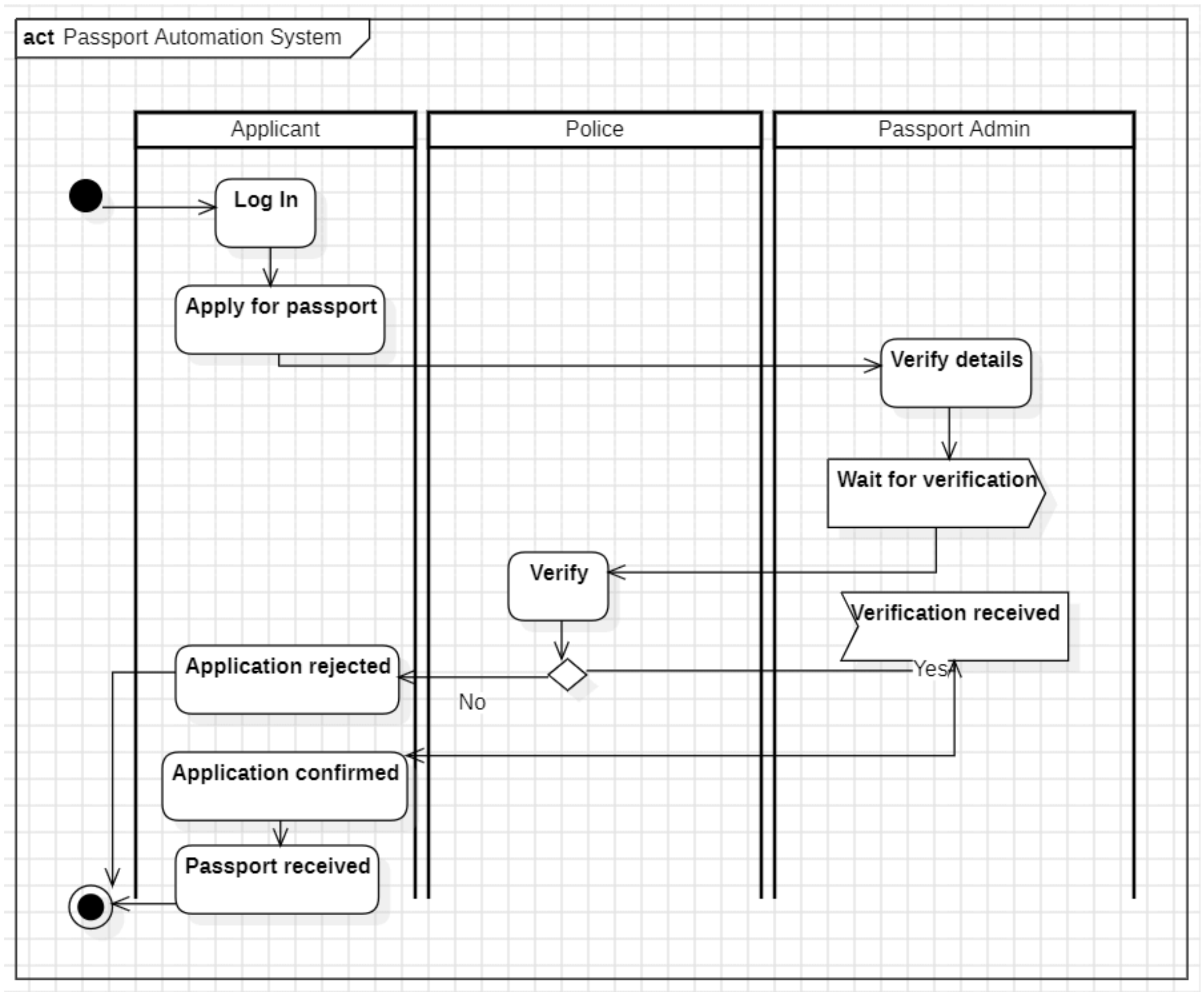


Fig 5.5: Activity Diagram for Stock Maintenance System

The activity diagram outlines the workflow from form submission to passport delivery. It shows verification tasks, parallel background checks, approval steps, and final dispatch. Decision points model exceptions like insufficient documents. It clarifies the overall processing pipeline.