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

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Engineering in
Computer Science and Engineering

Submitted by:

Amrutha Muralidhar

1BM21CS257

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

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(22CS6PCSEO) laboratory has been carried out by **Amrutha Muralidhar (1BM21CS257)** during 6th Semester Mar-June-2024.

Signature of the Faculty Incharge:

Dr Pallavi G B

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

Table of Contents

1. HOTEL MANAGEMENT SYSTEM

1.1 Problem Statement

The Hotel Management System is a software solution designed to streamline hotel operations. It maintains information on guests, staff, and rooms, managing reservations, check-ins, and checkouts. The system tracks services provided to guests, handles billing and payments, and generates invoices. Additionally, it manages staff schedules, payroll, and performance evaluations. The goal is to enhance operational efficiency, improve guest satisfaction, and provide insights into hotel performance.

1.2 Software Requirement Specification

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. This document serves as a guide for the development team to ensure all necessary functionalities are included and provides a clear understanding of the system's objectives.

1.2 Scope of this Document

This document describes the overall functioning and main objectives of the Hotel Management System. It details the value the system will provide to customers, including efficient hotel operations management, improved guest satisfaction, and comprehensive performance insights. It also outlines the estimated development cost and time required for the project.

1.3 Overview

The Hotel Management System is a software application designed to manage hotel operations, including guest reservations, check-ins, check-outs, billing, and service management. The system

also handles staff scheduling, payroll, and performance evaluation, providing a streamlined approach to hotel management.

2. General Description

The Hotel Management System is designed to streamline hotel operations by providing a centralized platform for managing guest information, room inventory, reservations, check-ins, check-outs, billing, and service tracking. It enhances operational efficiency by allowing easy access to guest details and room availability, facilitating accurate billing and payment processing, and managing staff schedules and payroll. The system is tailored for hotel staff, including receptionists, managers, housekeeping staff, and administrative personnel, ensuring that they can efficiently handle day-to-day tasks. Its features and benefits include improved guest satisfaction, reduced manual errors, and comprehensive performance reporting, making it an essential tool for modern hotel management.

3. Functional Requirements

3.1 Guest Information Management

- Store and manage guest details
- Retrieve guest information quickly

3.2 Reservation and Room Management

- Manage room inventory and availability
- Handle reservations, check-ins, and check-outs

3.3 Service Tracking

- Track services requested by guests
- Manage housekeeping and room service tasks

3.4 Billing and Payments

- Generate invoices for guests
- Process various payment methods

3.5 Staff Management

- Manage staff schedules and shifts
- Track staff performance and payroll

4. Interface Requirements

4.1 User Interfaces

- Intuitive graphical user interface for easy navigation
- Responsive design for use on various devices

4.2 Software Interfaces

- Integration with existing hotel management software (if any)
- Support for data import/export functionalities

4.3 Communication Interfaces

- Real-time notifications for staff and guests
- Email and SMS integration for alerts and confirmations

5. Performance Requirements

5.1 System Performance

- Quick response times for user actions
- Efficient data processing and retrieval

5.2 Resource Utilization

- Optimal memory usage
- Scalability to handle increasing data and user load

5.3 Error Handling

- Low error rate in data processing
- Robust error logging and notification mechanisms

6. Design Constraints

6.1 Technical Constraints

- Use of a specific programming language or framework
- Compatibility with existing hotel hardware and software

6.2 Regulatory Constraints

- Compliance with data protection and privacy regulations
- Adherence to industry standards for security and operations

7. Non-Functional Attributes

7.1 Security

- Secure data storage and access control
- Protection against unauthorized access and data breaches

7.2 Portability

- Support for multiple operating systems and devices

7.3 Reliability

- High system uptime and availability
- Redundant systems to ensure continuous operation

7.4 Scalability

- Ability to scale with growing user and data demands
- Flexible architecture to accommodate future enhancements

7.5 Usability

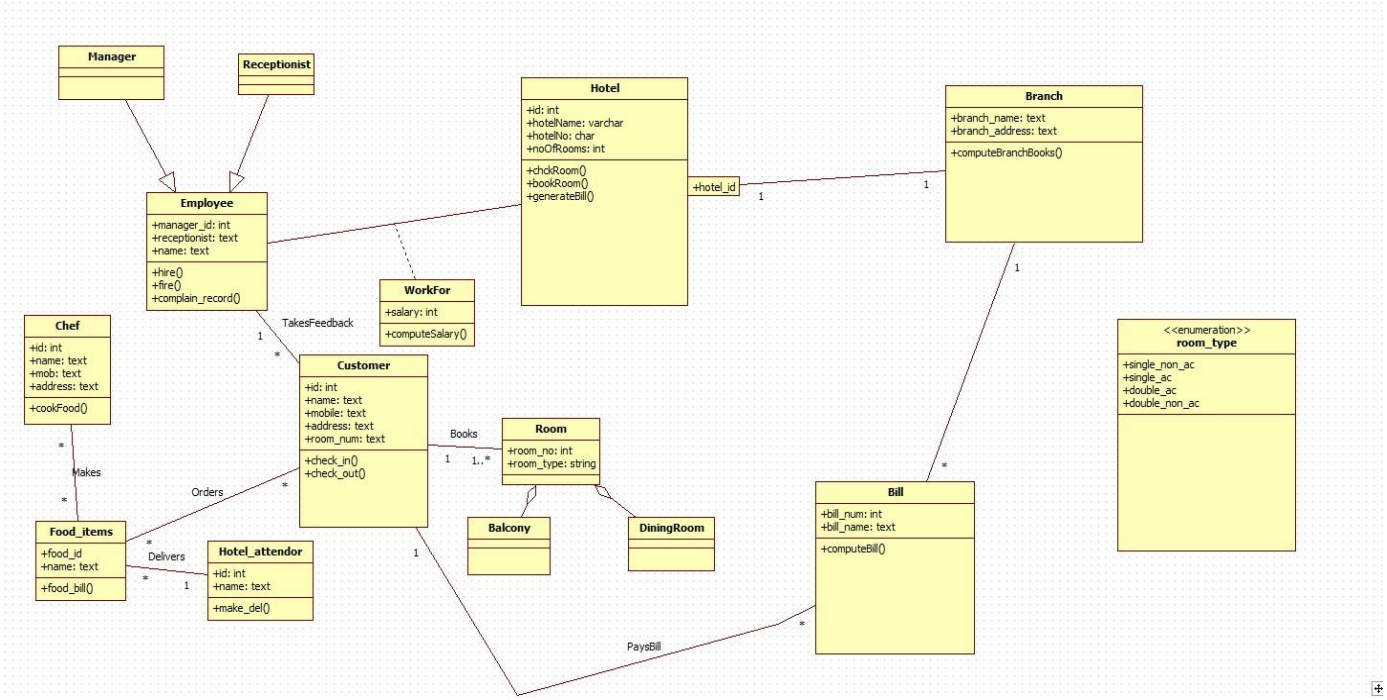
- User-friendly interface and easy navigation
- Comprehensive user documentation and support

8. Preliminary Schedule and Budget

8.1 Schedule

- Initial planning and requirement gathering: 1 month
- Design and architecture development: 2 months
- Implementation and coding: 4 months
- Testing and quality assurance: 2 months
- Deployment and training: 1 month

8.2 Budget



- Estimated development cost: $200,000
- Additional costs for hardware, training, and maintenance: $50,000

1.3 Class Diagram

Fig. 1.1

The Hotel Management System class diagram includes key entities such as Hotel, Branch, Customer, Room, Employee, Chef, Manager, and Receptionist, with specific functionalities and relationships. The **Hotel** class oversees multiple **Branch** entities, each managing its **Room**, **Employee**, and customer interactions. The **Customer** class handles room reservations, check-ins, check-outs, and service requests, interacting directly with the **Receptionist** for these tasks. **Rooms** have attributes like roomNumber, type (enumerated as different room types), status, and price, and are booked by customers. The **Employee** class, inherited by **Manager** and **Staff**, manages overall hotel operations. The **Chef** class, a specialized **Staff**, prepares orders and room services requested by customers. Additionally, the system generates bills for customers based on their room bookings and services utilized. This structure ensures efficient management of hotel operations, from customer service to staff coordination and billing.

1.4 State Diagram

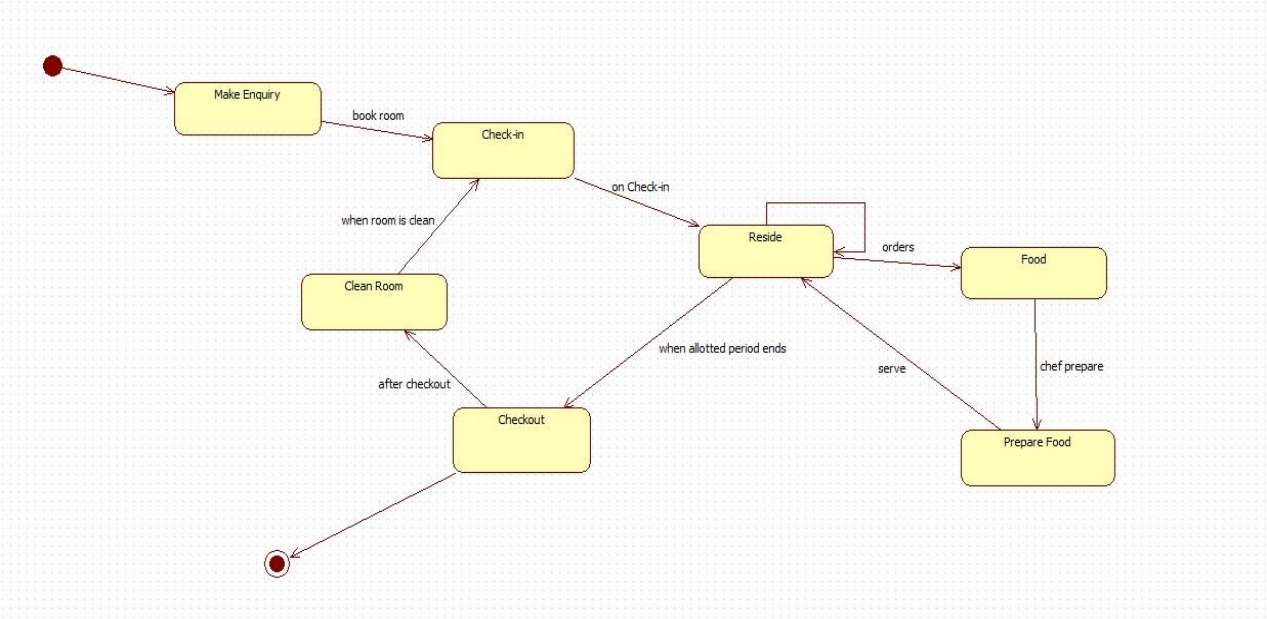


Fig. 1.2

The state diagram for the Hotel Management System illustrates the guest's journey, beginning with making an enquiry about room availability. If suitable, the guest proceeds to check in, where room allocation and key issuance occur. While residing, the guest can use various hotel services, including ordering food from the hotel restaurant, which is prepared by the chef and delivered to

the room. During the stay, housekeeping staff clean the room either on a schedule or upon request. Finally, the guest checks out, settling the bill for the room and any additional services used, completing the stay cycle as the guest leaves the hotel, and the room is prepared for the next guest.

1.5 Use Case Diagram

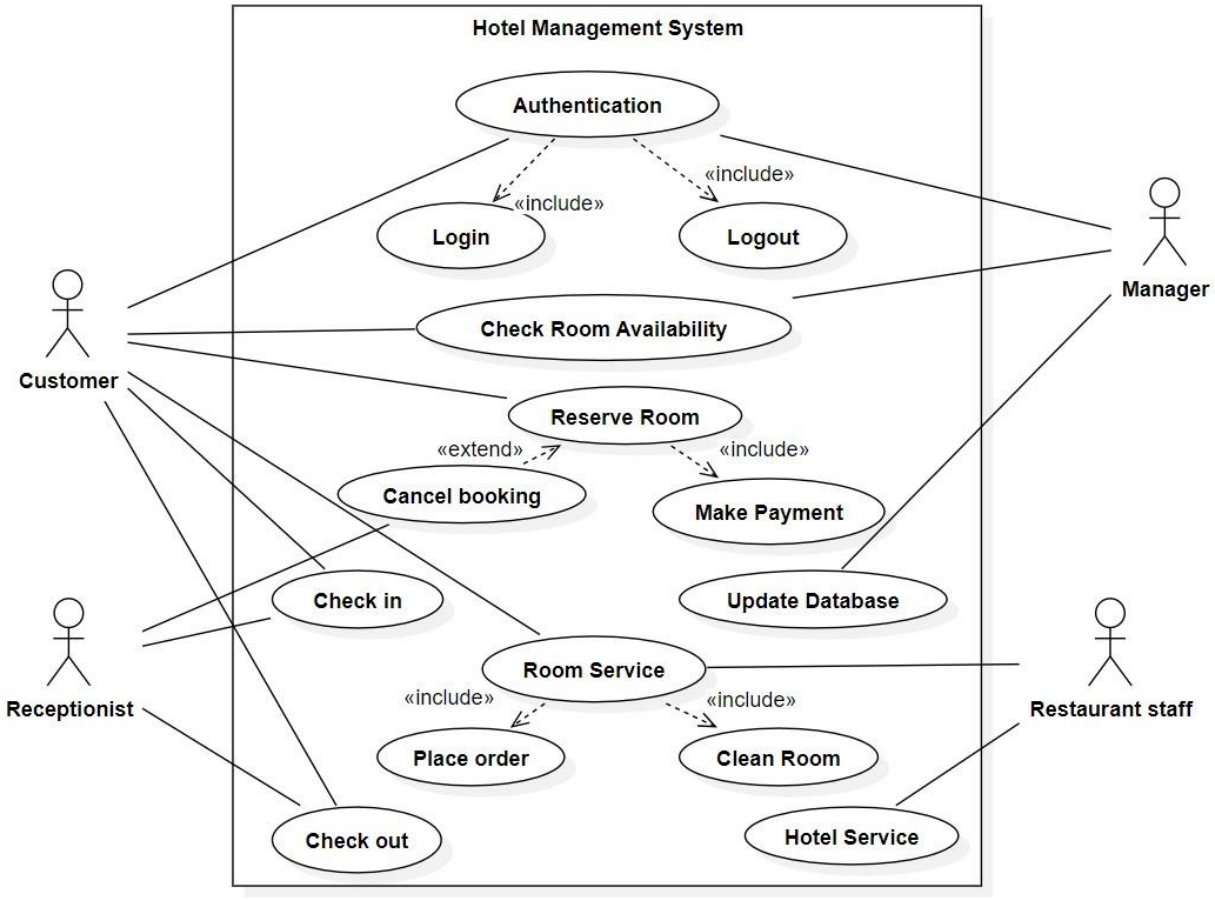


Fig. 1.3

The use case diagram for the Hotel Management System outlines the interactions between the system's actors, including customers, receptionists, managers, and staff. Customers can perform actions such as making reservations, checking in, ordering food, and checking out. Receptionists handle customer interactions, including managing reservations, checking guests in and out, and handling enquiries. Managers oversee the entire hotel operation, including monitoring

reservations, generating reports, and managing staff. Staff members, including housekeeping and kitchen staff, perform tasks such as cleaning rooms, preparing and delivering food orders, and assisting guests as needed. This use case diagram provides a comprehensive overview of the system's functionality and user interactions, ensuring efficient hotel operations and guest satisfaction.

1.6 Sequence Diagram

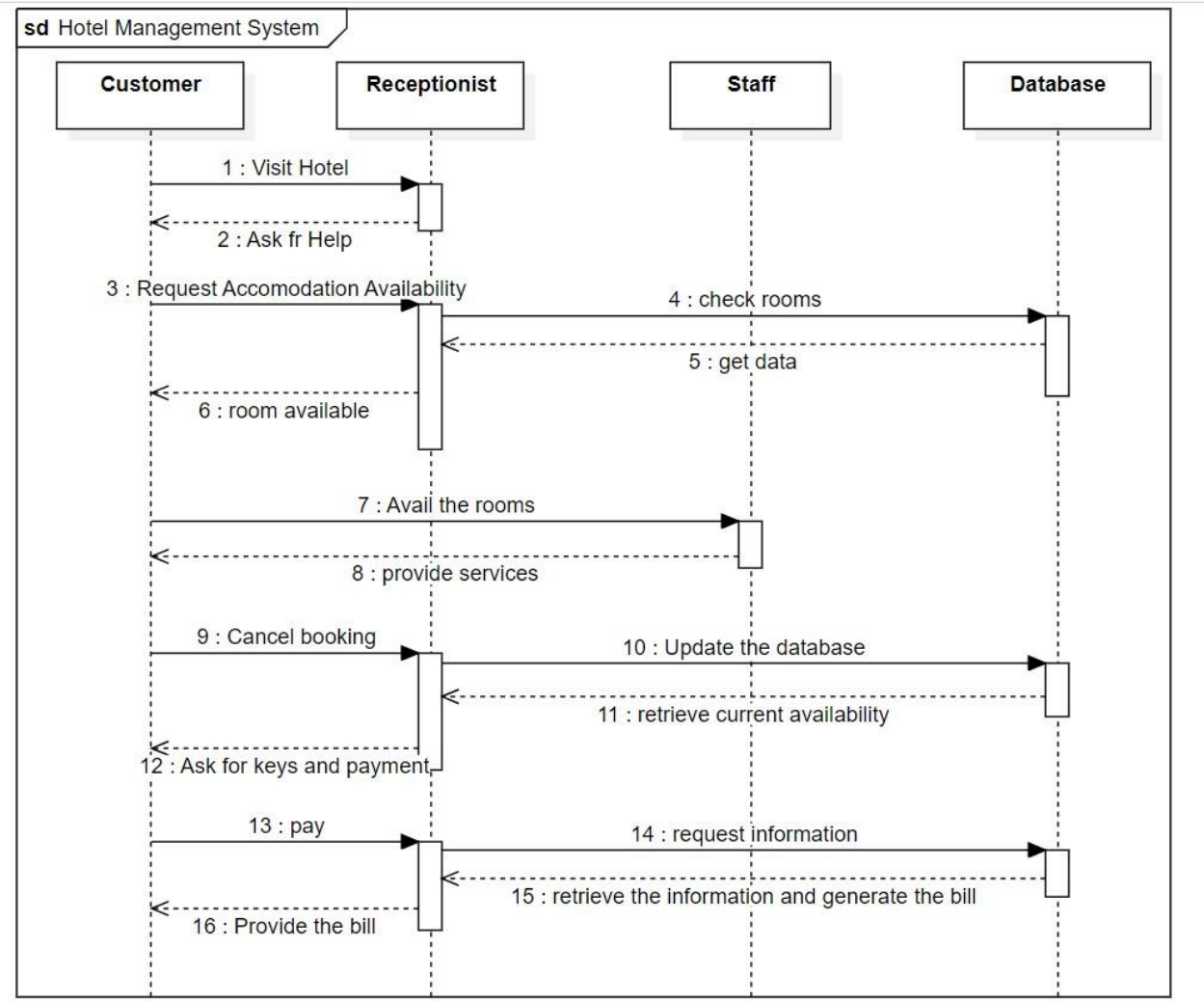


Fig. 1.4

The sequence diagram for the Hotel Management System illustrates the sequence of interactions between the system's actors, including customers, receptionists, staff, and the database. It begins with the customer making a reservation, which the receptionist verifies and records in the database.

Upon arrival, the customer checks in with the receptionist, who updates the database accordingly. The staff receives room cleaning requests from the receptionist, which are logged in the database. When the customer requests room service, the receptionist forwards the request to the staff, who fulfill it and update the database with the completed task. Finally, when the customer checks out, the receptionist updates the database, marking the room as vacant. This sequence diagram demonstrates the coordinated flow of actions between the actors and the database, ensuring smooth hotel operations and customer satisfaction.

1. 7 Activity Diagram

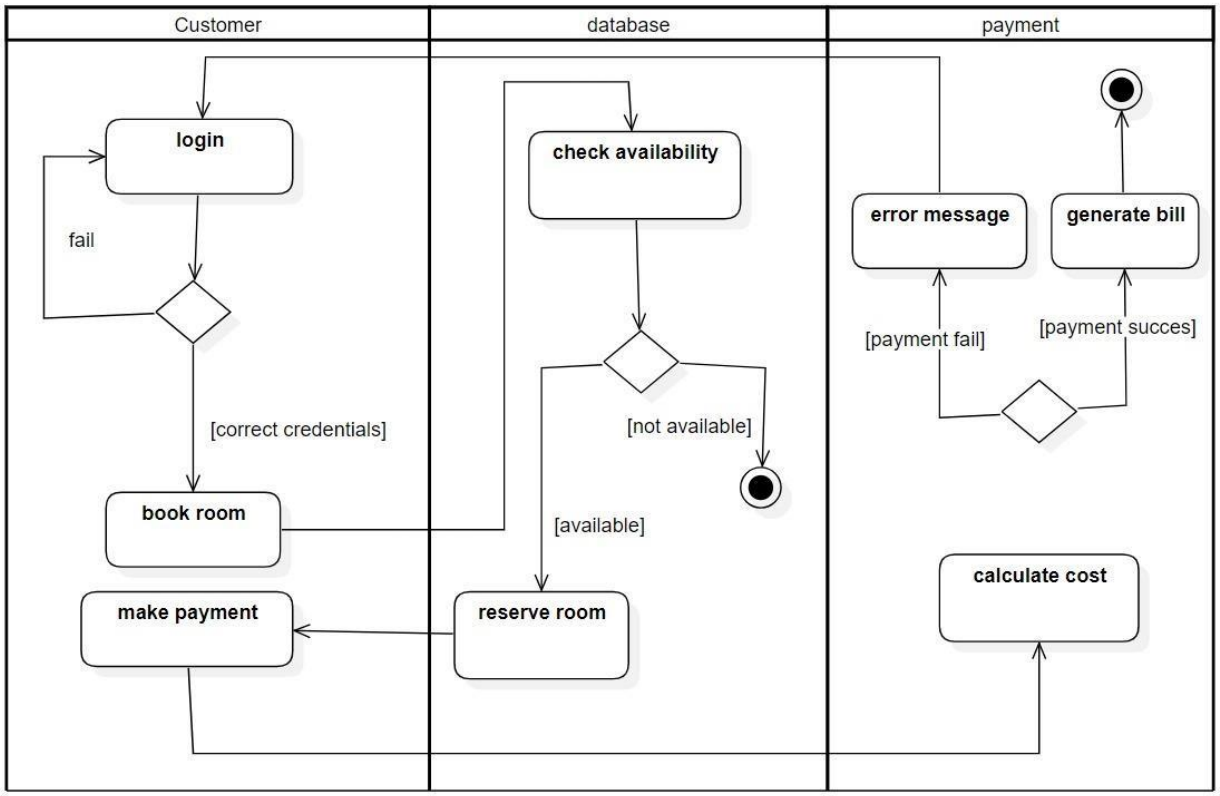


Fig. 1.5

The activity diagram for the Hotel Management System provides a visual representation of the sequential flow of activities within the system. It outlines the steps involved in various processes, such as room reservation, check-in, room cleaning, and check-out. Each activity is represented by a node, with arrows indicating the sequence of actions and decision points. The activity diagram

helps to illustrate the order of operations and the dependencies between different tasks, ensuring efficient workflow management and clear understanding of the system's processes.

2.CREDIT CARD PROCESSING

2.1 Problem Statement

The Credit Card Processing System aims to address the challenges faced by businesses and financial institutions in securely managing credit card transactions. The system will focus on enhancing security measures, minimizing transaction processing times, simplifying user management, ensuring scalability, and maintaining compliance with industry standards. The primary goal is to provide a secure, efficient, and user-friendly platform for processing credit card payments, benefiting businesses, financial institutions, and users alike.

2.2 Software Requirement Specification

1. Introduction

1.1 Purpose of this Document

The purpose of this SRS document is to define the requirements for developing a Credit Card Processing System. It serves as a comprehensive guide for the development team, stakeholders, and other involved parties to understand the system's functionalities, interfaces, performance expectations, design constraints, and non-functional attributes.

1.2 Scope of this Document

This document encompasses the entire scope of the Credit Card Processing System, including its objectives, features, user interactions, system performance, security measures, and development

timeline. It outlines the value proposition of the system to stakeholders, highlighting its importance in enhancing payment processing efficiency and security.

1.3 Overview

The Credit Card Processing System is designed to streamline credit card transactions for businesses, financial institutions, and users. It will provide a secure platform for authorizing, processing, and settling credit card payments, with a focus on user experience, data security, compliance with industry standards, and scalability to handle increasing transaction volumes.

2. General Description

The Credit Card Processing System will consist of the following key components and functionalities:

2.1 User Management

- Users can register, login, and manage their accounts.
- Account management includes adding, updating, and deleting credit card details.
- User authentication and access control mechanisms ensure secure access to accounts.

2.2 Transaction Processing

- Real-time authorization, processing, and settlement of credit card transactions.
- Support for multiple payment gateways and card networks.
- Transaction history tracking and reporting for users and administrators.

2.3 Security Measures

- Encryption and tokenization of sensitive credit card information.
- Compliance with Payment Card Industry Data Security Standard (PCI DSS) requirements.
- Regular security audits, vulnerability assessments, and intrusion detection mechanisms.

2.4 Interfaces

- Client Interface: Web and mobile interfaces for users to initiate and manage transactions.
- Employee Interface: Dashboard for employees to monitor transactions, handle disputes, and provide support.

- Admin Interface: Centralized dashboard for system administrators to manage settings, users, and reports.

3. Functional Requirements

3.1 User Account Management

- Users should be able to register with valid personal information and login securely.
- Users can add, update, and delete credit card details in their accounts.
- Account management actions should be logged for auditing purposes.

3.2 Transaction Processing

- The system must support real-time authorization and processing of credit card transactions.
- Transaction status updates should be visible to users and administrators.
- Automated settlement processes should be implemented based on transaction outcomes.

4. Interface Requirements

4.1 Client Interface

- Intuitive web and mobile interfaces for users to manage their accounts and initiate transactions.
- Responsive design for seamless user experience across devices.

4.2 Employee Interface

- Role-based access control for employees with different permissions (e.g., customer support, transaction monitoring).
- Dashboard with transaction tracking, dispute resolution tools, and customer communication features.

4.3 Admin Interface

- Centralized dashboard for administrators to manage system settings, user accounts, and transaction reports.

- Reporting tools for monitoring system performance, transaction volumes, and compliance metrics.

5. Performance Requirements

5.1 Response Time

- System response time for user actions should be less than 2 seconds.
- Transaction processing time should not exceed 5 seconds.

5.2 Scalability

- The system should handle a minimum of 1000 concurrent transactions.
- Scalable architecture to accommodate future growth and increased transaction volumes.

6. Design Constraints

6.1 Security Constraints

- Encryption standards (e.g., AES-256) for securing credit card data.
- Use of secure communication protocols (e.g., HTTPS) for data transmission.
- Regular security patches and updates to mitigate vulnerabilities.

6.2 Technology Stack

- Backend development using Java Spring Boot framework for robustness and scalability.
- Frontend development with React.js for responsive and dynamic user interfaces.
- Database management using PostgreSQL for data storage and retrieval.

7. Non-Functional Attributes

7.1 Security

- Data encryption at rest and in transit.
- Access control mechanisms to prevent unauthorized access to sensitive data.
- Security incident response and mitigation procedures.

7.2 Reliability

- System uptime of at least 99.9% to ensure uninterrupted service availability.
- Automated backup and recovery processes for data protection and disaster recovery.

7.3 Scalability

- Horizontal scalability to handle increased transaction volumes during peak periods.
- Load balancing and caching mechanisms for optimal performance.

8. Preliminary Schedule and Budget

8.1 Development Timeline

- Estimated development duration: 6 months.
- Iterative development approach with regular milestones and reviews.

8.2 Budget Estimate

- Development budget: $200,000.
- Includes costs for development, testing, deployment, and initial maintenance.

2.3 Class Diagram

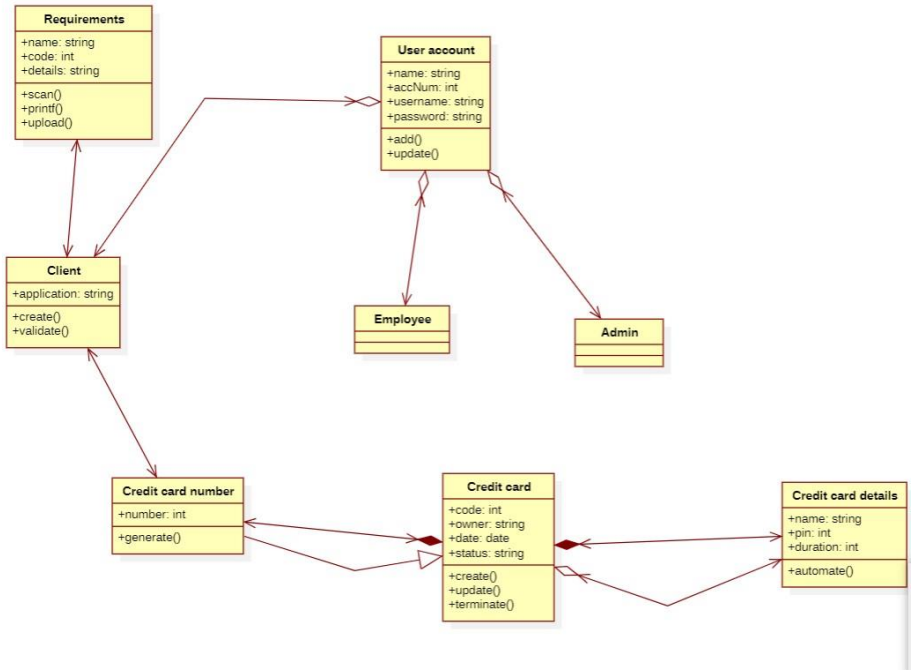


Fig. 2.1

The class diagram for the Credit Card Processing System represents the various classes and their relationships that are fundamental to the system's functionality. At the core of the diagram are classes such as User Account, Credit Card, Transaction, Employee, and Admin, each encapsulating specific attributes and behaviors. The User Account class manages user-related functionalities like account creation, login, and credit card management. It is associated with the Credit Card class, which holds details such as card number, expiration date, and security code. The Transaction class handles the processing, authorization, and settlement of credit card transactions, linking back to both User Account and Credit Card for transaction history and payment details. The system's administrative functionalities are represented by the Employee and Admin classes, with roles defined for customer support, transaction monitoring, and system management. These classes interact through various relationships such as aggregation, inheritance, and associations, depicting the flow of data and operations within the Credit Card Processing System.

2.4 State diagram

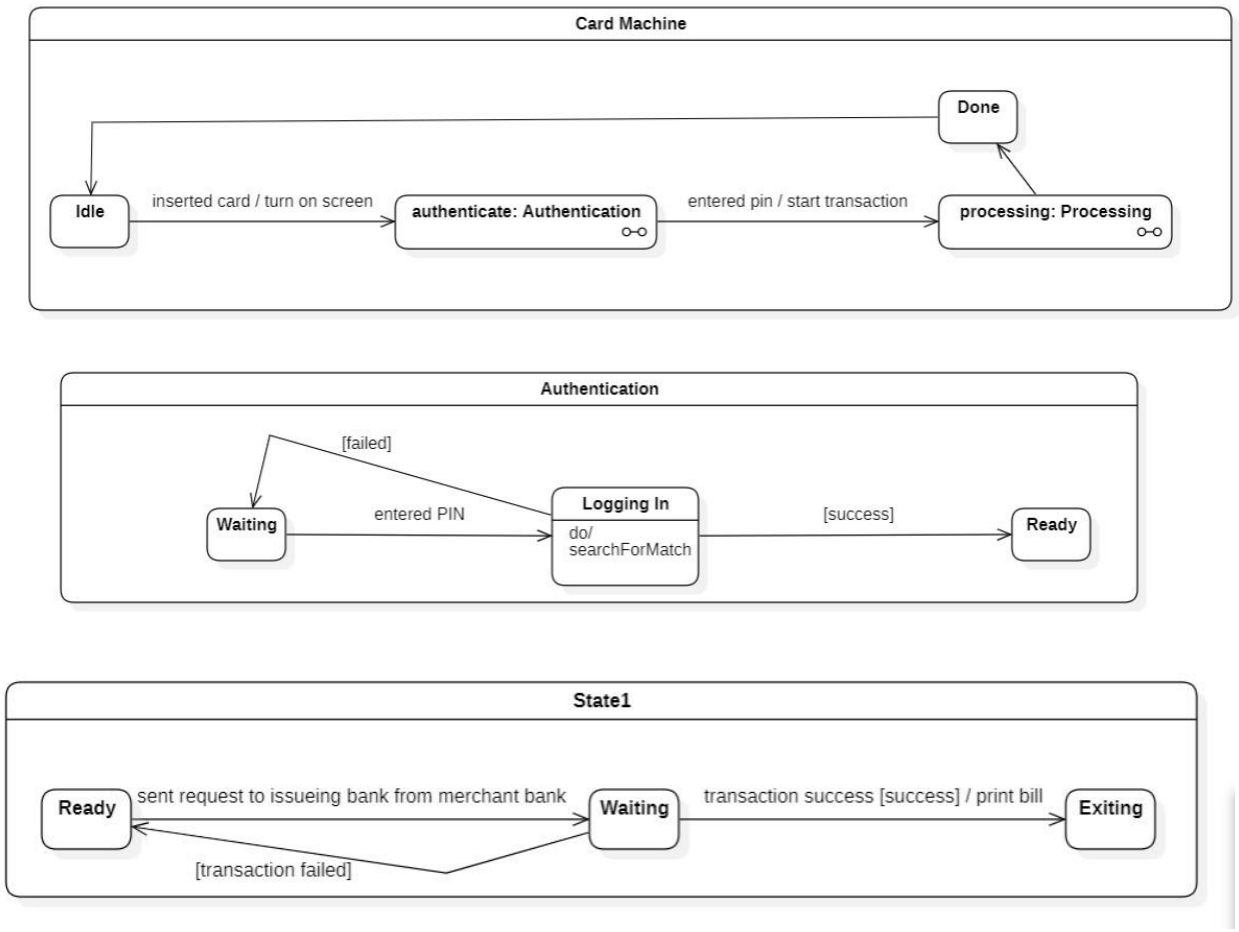


Fig 2.2

The state diagram for the Credit Card Processing System illustrates the different states that an object or system component can transition through in response to events or actions. For example, a state diagram may represent the states of a credit card transaction, including "Initiated," "Authorization Pending," "Approved," "Declined," and "Completed." Transitions between these states are triggered by events such as user request, authorization response from the bank, and transaction confirmation. The diagram visually represents these states as nodes, with arrows indicating transitions and the events/actions triggering those transitions. Each state may have associated actions or behaviors, such as updating transaction status, sending notifications, or

logging events. State diagrams help in understanding the lifecycle of objects or processes within the system, identifying possible states and transitions, and defining the behaviors associated with each state change, providing a clear overview of system dynamics and state-dependent functionalities.

2.5 Use Case Diagram

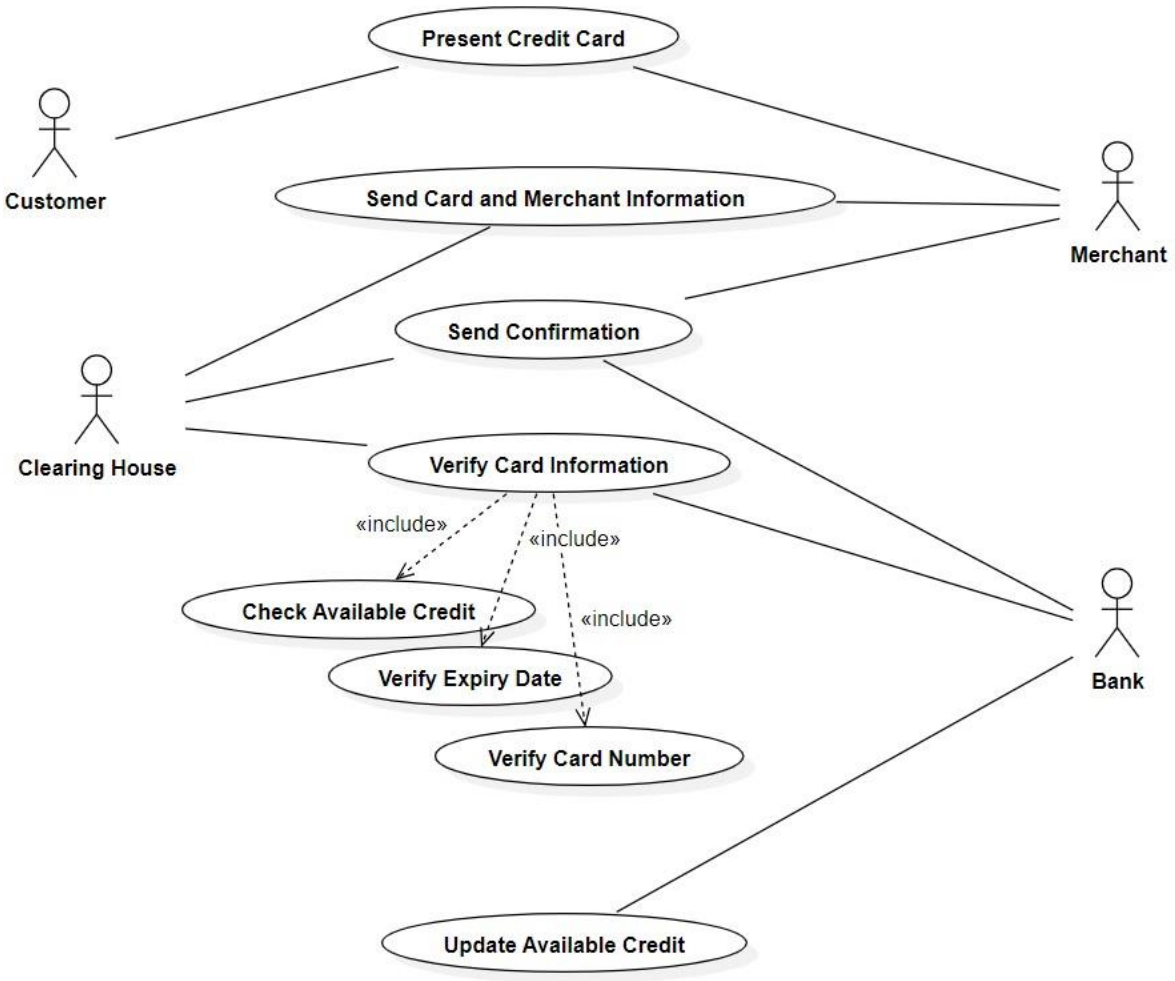


Fig 2.3

The use case diagram for the Credit Card Processing System illustrates the interactions between actors (users and system roles) and the system itself, showcasing the system's functionalities from a user's perspective. Actors include Client/User, Employee, and Admin, each with specific roles and interactions with the system. For instance, the Client/User actor can perform actions such as

registering an account, adding credit cards, initiating transactions, and viewing transaction history. The Employee actor, on the other hand, can manage transactions, handle disputes, and provide customer support. The Admin actor has administrative privileges, allowing them to manage system settings, user accounts, and generate reports. Use cases like "Register Account," "Add Credit Card," "Initiate Transaction," "Manage Transactions," "Handle Disputes," "Generate Reports," and others are depicted with their corresponding actors, showcasing the various functionalities and interactions within the Credit Card Processing System. This diagram serves as a visual representation of the system's use cases and helps stakeholders understand the system's functionality from a user's perspective.

2.6 Sequence diagram

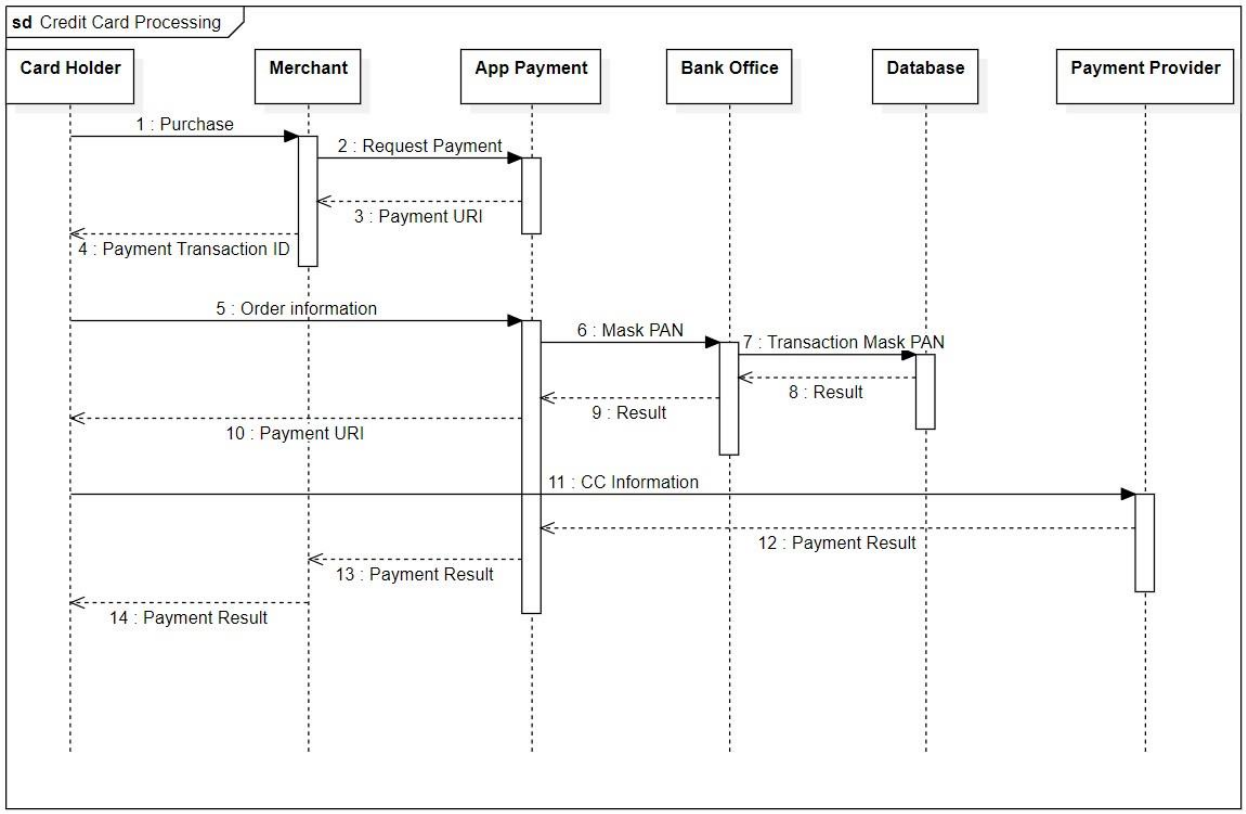


Fig. 2.4

The sequence diagram for the Credit Card Processing System illustrates the interactions and message exchanges between various objects or components in a chronological order, showcasing how different parts of the system collaborate to accomplish specific functionalities. For example, a typical sequence might involve a Client/User initiating a transaction, where the system first authenticates the user, then verifies the credit card details, checks for available funds, and finally processes the transaction. This sequence involves interactions between objects such as the Client/User, User Account, Credit Card, Transaction Processor, and Bank Interface. Each step in the sequence represents a message exchange or method call between these objects, detailing the flow of control and data within the system. The sequence diagram provides a clear visualization of the system's behavior during specific scenarios, helping developers understand the system's internal workings and facilitating communication among stakeholders regarding system functionalities and interactions.

2.7 Activity Diagram

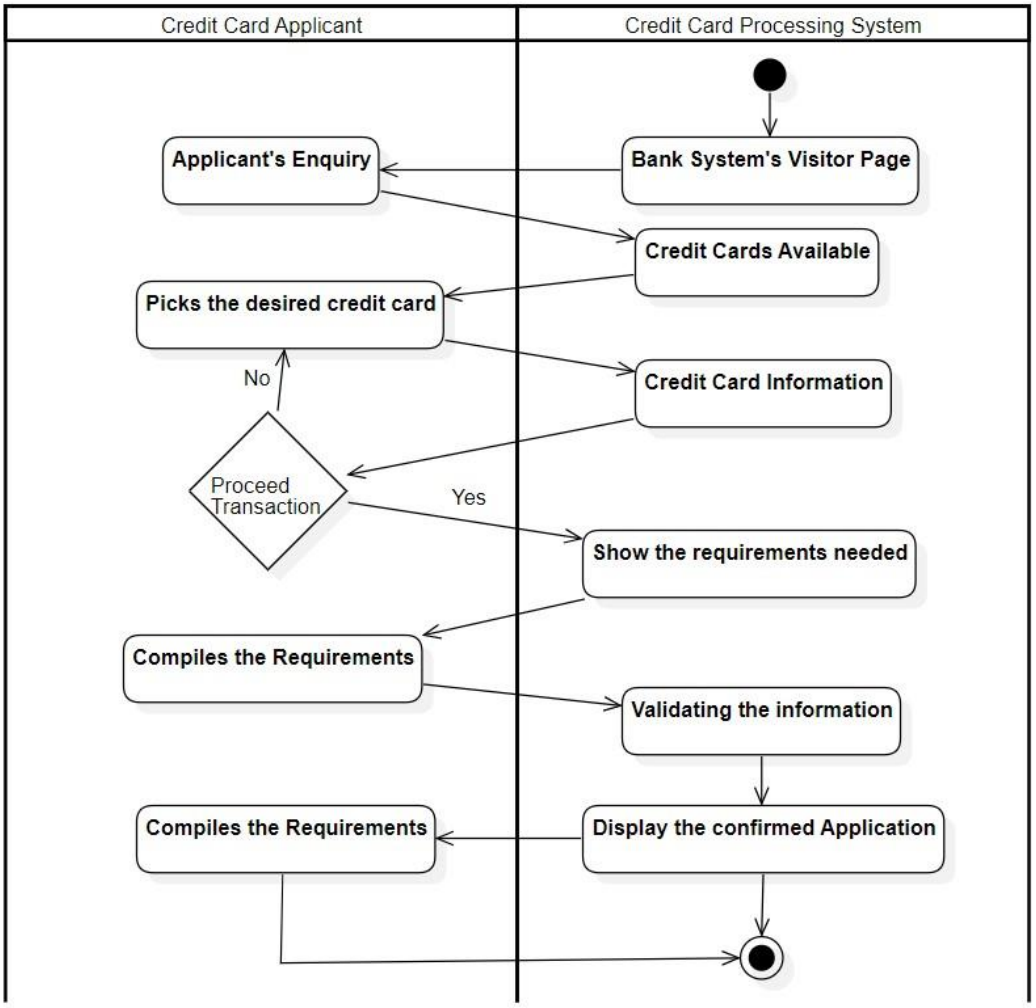


Fig. 2.5

The activity diagram for the Credit Card Processing System depicts the flow of activities or actions within the system, focusing on the steps involved in performing a specific task or process. For instance, the diagram may show the activities involved in processing a credit card transaction, starting from the user initiating the transaction to the final confirmation of the transaction's success. Activities such as user authentication, credit card verification, fund availability check, transaction processing, and transaction confirmation are represented as nodes in the diagram, connected by arrows to indicate the sequence of activities. Decision points, represented by diamonds, may be included to show conditional branching based on certain criteria (e.g., insufficient funds leading to a declined transaction). The activity diagram helps stakeholders visualize the workflow of the

Credit Card Processing System, understand the sequence of actions required to complete a task, and identify potential decision points or alternative paths within the system's processes.

3. LIBRARY MANAGEMENT SYSTEM

3.1 Problem Statement

Libraries face significant challenges in managing their operations efficiently due to the reliance on manual processes and outdated systems. These issues include difficulties in cataloging books, managing member records, tracking borrowing and returning transactions, and generating insightful reports. The manual handling of these tasks often leads to errors, delays, and a poor user experience for both library staff and members. There is a need for an automated, user-friendly, and reliable Library Management System (LMS) that streamlines these operations, enhances accuracy,

and provides real-time access to information, ultimately improving the overall efficiency and effectiveness of library services.

3.2 Software Requirement Specification

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to provide a detailed description of the Library Management System (LMS). It aims to outline the system's functionalities, interfaces, performance requirements, and constraints to ensure a clear understanding of the system's objectives and development process. This document serves as a guide for stakeholders, developers, and users, ensuring that all requirements are met for an efficient and effective library management solution.

1.2 Scope of this Document

This document encompasses the overall functionality and objectives of the LMS, detailing how it will manage library operations, including book cataloging, member management, borrowing and returning processes, and report generation. It will provide value to library staff and members by automating and streamlining library operations. Additionally, it includes an overview of the estimated development cost and the time required to implement the system.

1.3 Overview

The Library Management System is a software application designed to manage the operations of a library. It will facilitate the management of books, members, and transactions (borrowing and returning books). The system aims to enhance the efficiency of library operations, reduce manual errors, and provide real-time data on library activities.

2 General Description

The Library Management System will serve the primary function of managing library resources and operations. It will cater to the needs of librarians and library members, providing features such as book cataloging, member management, transaction handling, and reporting. The system is crucial for maintaining organized records and offering a seamless library experience.

2.1 User Characteristics

- **Librarians**: Manage the entire system, including book cataloging, member registration, and overseeing transactions.
- **Members**: Can search for books, borrow and return books, and check their transaction history.

2.2 Features and Benefits

- **Book Cataloging:** Easy entry and management of book details.
- **Member Management:** Efficient handling of member registrations and updates.
- **Transaction Management:** Automated tracking of book borrow and return activities.
- **Reporting:** Generation of various reports for better insights into library operations

2.3 Importance

The LMS is essential for the smooth operation of a library. It minimizes manual work, reduces errors, and provides quick access to information, thus enhancing the overall user experience.

3 Functional Requirements

The functional requirements detail the specific operations the LMS will perform, ranked by priority.

3.1 Book Cataloging

- Adding, updating, and deleting book records.
- Searching and filtering books by various criteria (title, author, genre).

3.2 Member Management

- Registering new members.
- Updating member information.
- Deleting member records.

3.3 Transaction Handling

- Recording book borrow and return transactions.
- Managing due dates and overdue notifications.
- Tracking transaction history for each member.

3.4 Reporting

- Generating reports on borrowed books, overdue books, and member activity.
- Providing summary statistics on library operations.

4 Interface Requirements

The LMS will have several interfaces for effective communication and operation.

4.1 User Interface

- A user-friendly graphical interface for librarians and members.
- Accessible via web browsers and potentially mobile applications.

4.2 Database Interface

Secure and efficient interaction with the backend database for storing and retrieving data.

4.3 External Interfaces

- Integration with email services for sending notifications.
- Potential integration with external library networks for inter-library loan services.

5 Performance Requirements

5.1 Response Time

- The system should respond to user actions (e.g., searching for a book) within 2 seconds.

5.2 Memory Usage

- Efficient memory usage to handle a large number of records without performance degradation.

5.3 Error Rate

- The system should have a minimal error rate, aiming for less than 0.01% in transaction processing.

6 Design Constraints

6.1 Hardware Constraints

- Must be compatible with existing library hardware, such as computers and barcode scanners.

6.2 Software Constraints

- Use of a specific database management system (e.g., MySQL, PostgreSQL).
- Compatibility with standard operating systems (Windows, macOS, Linux).

6.3 Regulatory Constraints

- Compliance with data protection regulations, ensuring the security and privacy of member information.

7 Non-Functional Attributes

7.1 Security

- Implement robust authentication and authorization mechanisms.
- Ensure data encryption for sensitive information.

7.2 Portability

- The system should be portable across different operating systems and devices.

7.3 Reliability

- High reliability with minimal downtime, aiming for 99.9% availability.

7.4 Reusability

- Modular design to allow reuse of components in other library-related applications.

7.5 Application Compatibility

- Compatible with other applications used in library operations, such as cataloging software.

7.6 Data Integrity

- Ensure data integrity with regular backups and validation checks.

7.7 Scalability

- The system should be scalable to accommodate the growing number of books and members.

8 Preliminary Schedule and Budget

8.1 Schedule

- **Phase 1: Requirements Gathering and Analysis** (2 months)
- **Phase 2: Design** (1 month)
- **Phase 3: Development** (3 months)
- **Phase 4: Testing** (2 months)
- **Phase 5: Deployment** (1 month)
- **Total Duration**: 9 months

8.2 Budget

- **Personnel Costs**: $50,000
- **Software and Tools**: $10,000
- **Hardware Costs**: $5,000
- **Miscellaneous Expenses**: $5,000
- **Total Estimated Cost**: $70,000

This document outlines the comprehensive requirements and plans for the Library Management System, ensuring clarity and alignment among all stakeholders.

3.3 Class Diagram

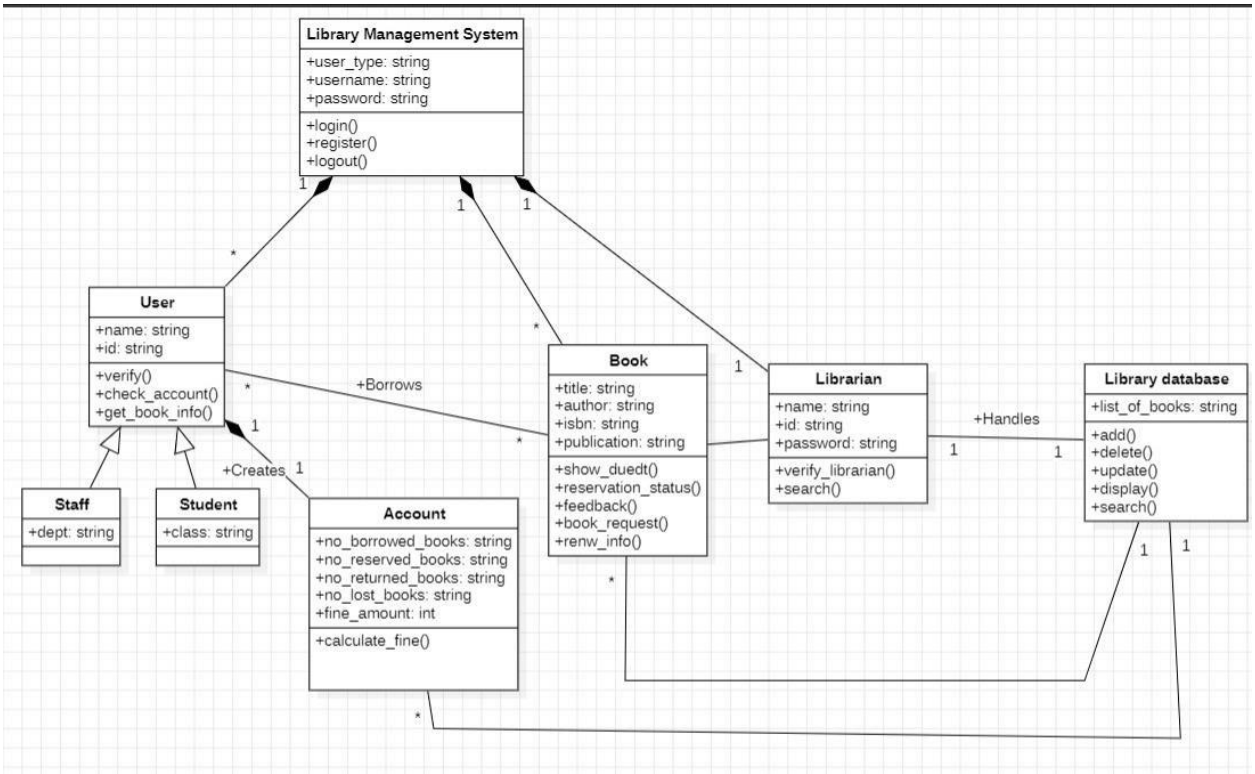


Fig. 3.1

The Library Management System class diagram includes key classes such as **User**, **Staff**, **Student**, **Librarian**, **Book**, and **LibraryAccount**. Users have attributes like **userID**, **name**, and methods for login/logout. Staff (inheriting from User) manage books and accounts, with Librarians (inheriting from Staff) overseeing library operations and cataloging books. Students (inheriting from User) can borrow and return books. Books have attributes like **bookID**, **title**, and availability status. LibraryAccount handles user accounts with attributes like **accountID**, **balance**, and methods to manage funds. Relationships include inheritance hierarchies and associations between users, accounts, and books.

3.4 State Diagram

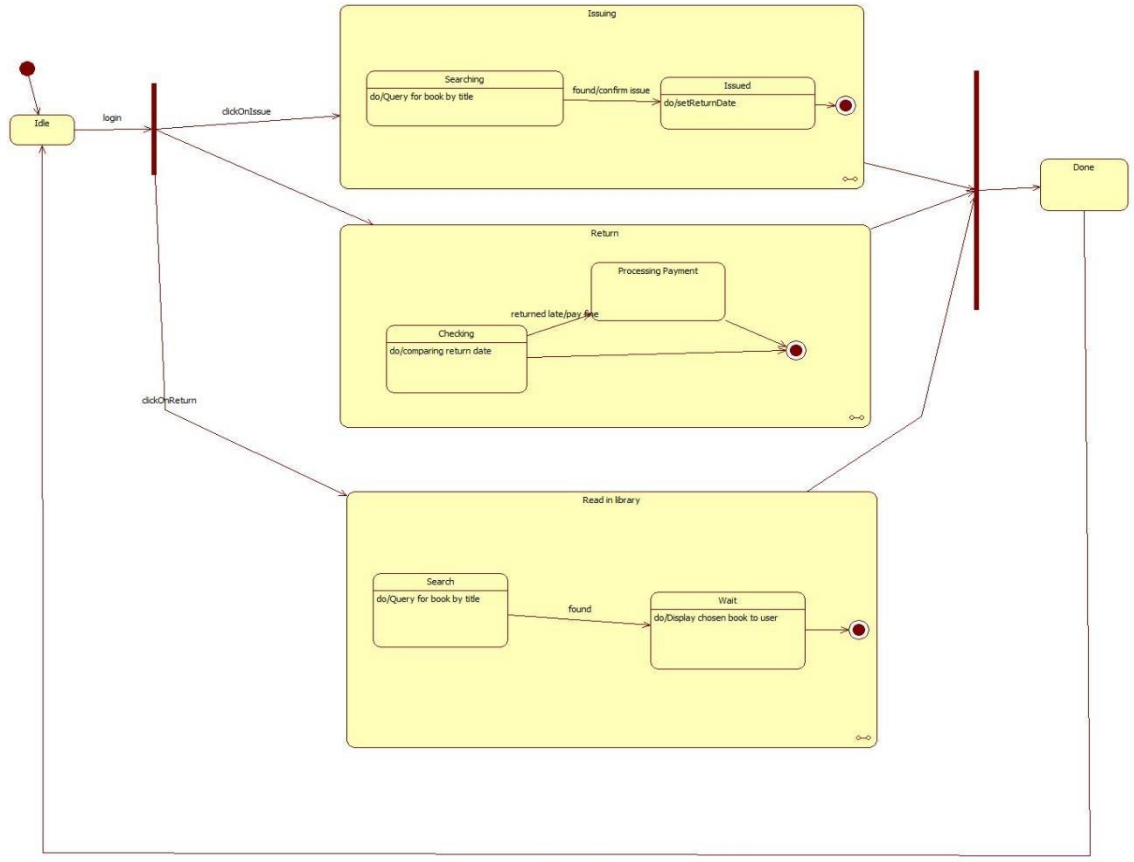


Fig. 3.2

The state diagram for the Library Management System outlines the various states a book can be in and the transitions between these states. The primary states include Available, Borrowed, Reserved, and Overdue. The transitions occur as follows: a book moves from Available to Borrowed when a student checks it out, from Borrowed to Available when it is returned, to Reserved if a user places a hold, and to Overdue if not returned by the due date. The diagram provides a clear, concise view of the lifecycle of a book within the library system.

3.5 Use case diagram

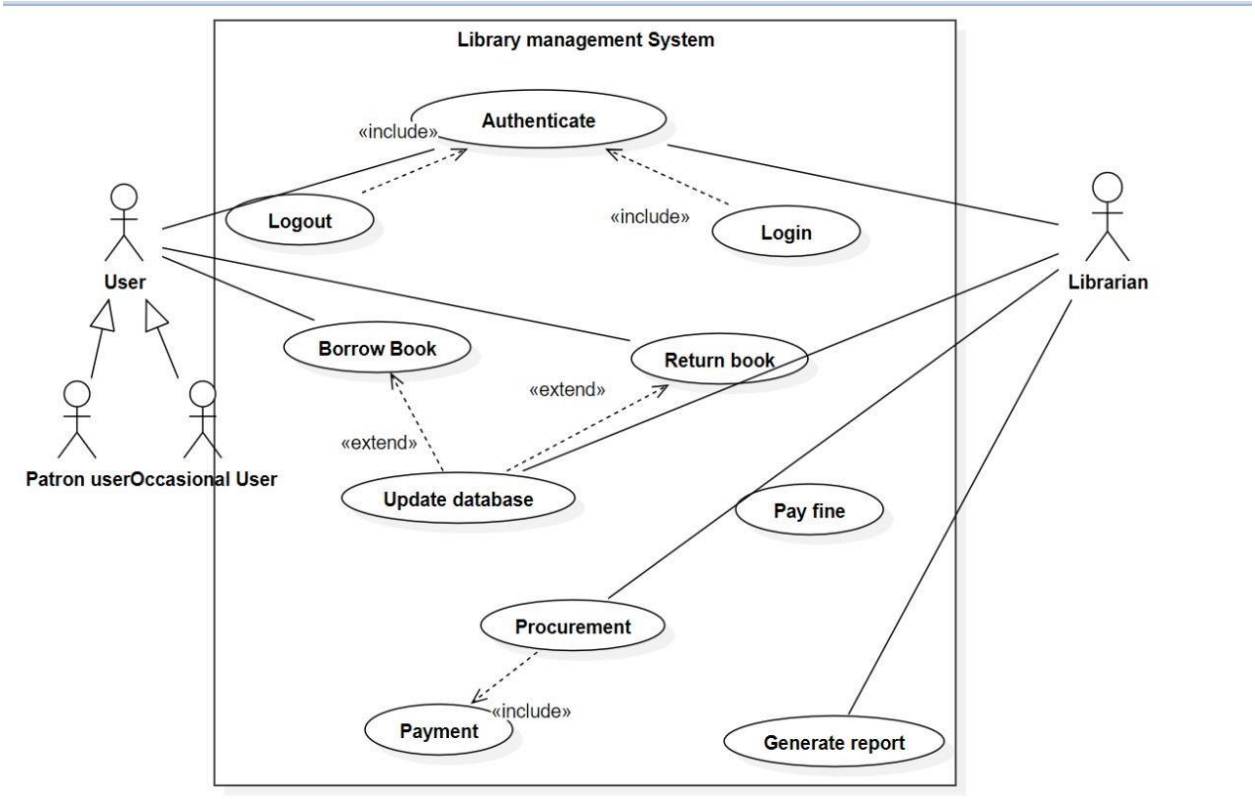


Fig. 3.3

The use case diagram for the Library Management System includes actors such as **Librarian**, **Student**, and **Staff**. The primary use cases are **Manage Books**, **Borrow Books**, **Return Books**, **Register Member**, **Generate Reports**, and **Manage Accounts**. **Librarians** handle book management and report generation, **Students** borrow and return books, and **Staff** manage member registrations and accounts. This diagram provides a concise overview of the interactions between users and the system's functionalities.

3.6 Sequence Diagram

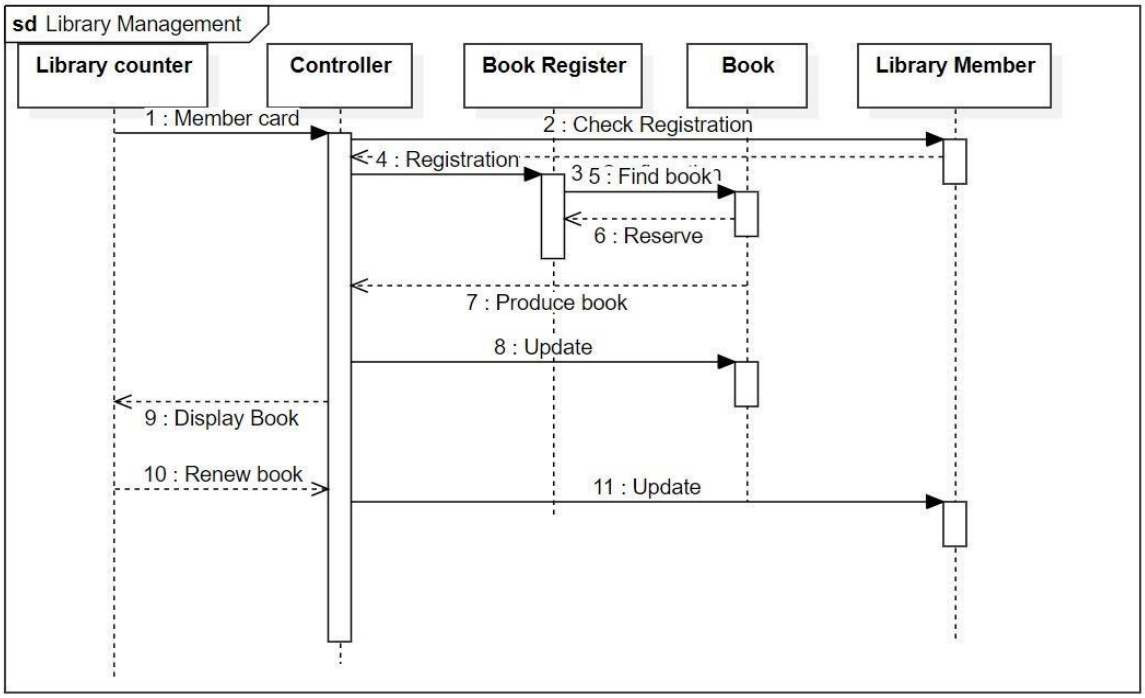


Fig. 3.4

The sequence diagram for the Library Management System shows the interactions between Librarian, Student, Staff, and the System. For borrowing a book, the Student requests to borrow, the System checks availability, and updates the status. For returning a book, the Student returns it, and the System updates the status. Librarians manage books by adding or updating details, and the System stores this information. Staff registers new members, with the System creating user accounts. Librarians request reports, and the System generates them. This diagram captures the flow of messages for these key processes.

3.7 Activity diagram

The activity diagram for the Library Management System depicts the workflow for key processes: borrowing a book, returning a book, managing books, registering members, and generating reports. For borrowing, the process includes checking availability and updating the

book status. Returning involves updating the status to available. Managing books involves adding or updating book details. Registering members involves creating user accounts. Generating reports involves retrieving data and creating the report. Each activity follows a sequential flow of actions to complete these tasks.

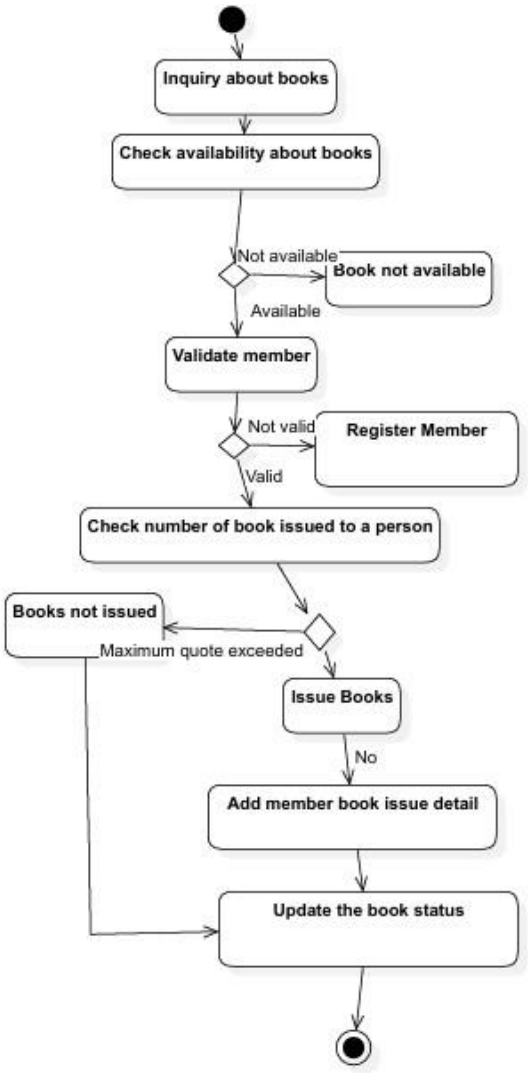


Fig. 3.5

4. STOCK MAINTENENCE SYSTEM

4.1 Problem Statement

Organizations face significant challenges in managing their stock efficiently due to the reliance on manual processes and outdated systems. These challenges include difficulties in accurately tracking stock levels, managing inventory movement, and generating insightful reports. Manual handling of these tasks often leads to errors, delays, and inefficiencies in stock management. There is a need for an automated, user-friendly, and reliable Stock Management System (SMS) that streamlines these operations, enhances accuracy, and provides real-time access to stock information, ultimately improving organizational performance.

4.2 Software Requirement Specification

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the requirements for the development of a Stock Management System (SMS). It provides a detailed description of the system's functionalities, interfaces, performance requirements, and constraints to guide stakeholders, developers, and users throughout the development process.

1.2 Problem Statement

Organizations face significant challenges in managing their stock efficiently due to manual processes and outdated systems. These challenges include difficulties in accurately tracking stock levels, managing inventory movement, and generating insightful reports. Manual handling of these tasks often leads to errors, delays, and inefficiencies in stock management. There is a need for an automated, user-friendly, and reliable Stock Management System that streamlines these operations, enhances accuracy, and provides real-time access to stock information, ultimately improving organizational performance.

1.3 Scope of this Document

This document encompasses the overall functionality and objectives of the Stock Management System, detailing how it will manage stock items, track inventory movement, generate reports, and facilitate efficient stock management. It includes an overview of the estimated development cost and time required for system implementation.

1.4 Overview

The Stock Management System is a software application designed to streamline the management of stock items, track inventory movement, and provide insights into stock operations. It aims to enhance the efficiency and accuracy of stock management processes, ultimately improving organizational performance. **2. General Description**

2.1 User Characteristics

Stock Manager: Responsible for managing stock items, updating inventory, and generating reports.

2.2 Features and Benefits

Stock Item Management: Easy addition, update, and deletion of stock items. Inventory Tracking: Real-time monitoring of stock movement (inward and outward). Reporting: Generation of stock reports for analysis and decision-making.

2.3 Importance

The SMS is essential for organizations to maintain optimal stock levels, reduce stockouts, and improve inventory management efficiency.

3. Functional Requirements

3.1 Stock Item Management

- Adding new stock items with details such as name, quantity, unit price, etc.
- Updating existing stock items including quantity and price.

- Deleting stock items from the inventory.

3.2 Inventory Tracking

- Recording stock movements (inward and outward) with timestamps.
- Tracking stock levels and generating alerts for low stock.

3.3 Reporting

- Generating reports on stock levels, stock movements, and inventory turnover.
- Providing summary statistics for analysis and decision-making.

4. Interface Requirements

4.1 User Interface

- Intuitive graphical interface for stock managers.
- Accessibility via web browsers and potentially mobile applications.

4.2 Database Interface

- Secure interaction with the backend database for storing and retrieving stock data.

4.3 External Interfaces

- Integration with email services for sending notifications.
- Potential integration with suppliers' systems for automated stock replenishment.

5. Performance Requirements

5.1 Response Time

- The system should respond to user actions (e.g., adding a stock item) within 2 seconds.

5.2 Memory Usage • Efficient memory usage to handle a large number of stock items without performance degradation. **5.3 Error Rate**

- The system should have a minimal error rate, aiming for less than 0.01% in stock recording.

6. Design Constraints

6.1 Hardware Constraints

- Compatibility with existing hardware infrastructure.

6.2 Software Constraints

- Use of a specific database management system (e.g., MySQL, PostgreSQL).
- Compatibility with standard operating systems (Windows, macOS, Linux).

7. Non-Functional Attributes

7.1 Security

- Implement robust authentication and authorization mechanisms.
- Ensure data encryption for sensitive stock information.

7.2 Portability

- The system should be portable across different operating systems and devices.

7.3 Reliability

- High reliability with minimal downtime, aiming for 99.9% availability.

7.4 Reusability

- Modular design to allow reuse of components in other inventory-related applications.

7.5 Data Integrity

- Ensure data integrity with regular backups and validation checks.

7.6 Scalability

- The system should be scalable to accommodate the growing number of stock items and transactions.

8. Preliminary Schedule and Budget

8.1 Schedule

- Phase 1: Requirements Gathering and Analysis (1 month)
- Phase 2: Design (1 month)
- Phase 3: Development (3 months)
- Phase 4: Testing (1 month)
- Phase 5: Deployment (1 month)
- Total Duration: 7 months

8.2 Budget

- Personnel Costs: $40,000
- Software and Tools: $15,000
- Hardware Costs: $5,000
- Miscellaneous Expenses: $5,000
- Total Estimated Cost: $65,000

This document outlines the comprehensive requirements and plans for the Stock Management System, ensuring clarity and alignment among all stakeholders.

4.3 Class Diagram

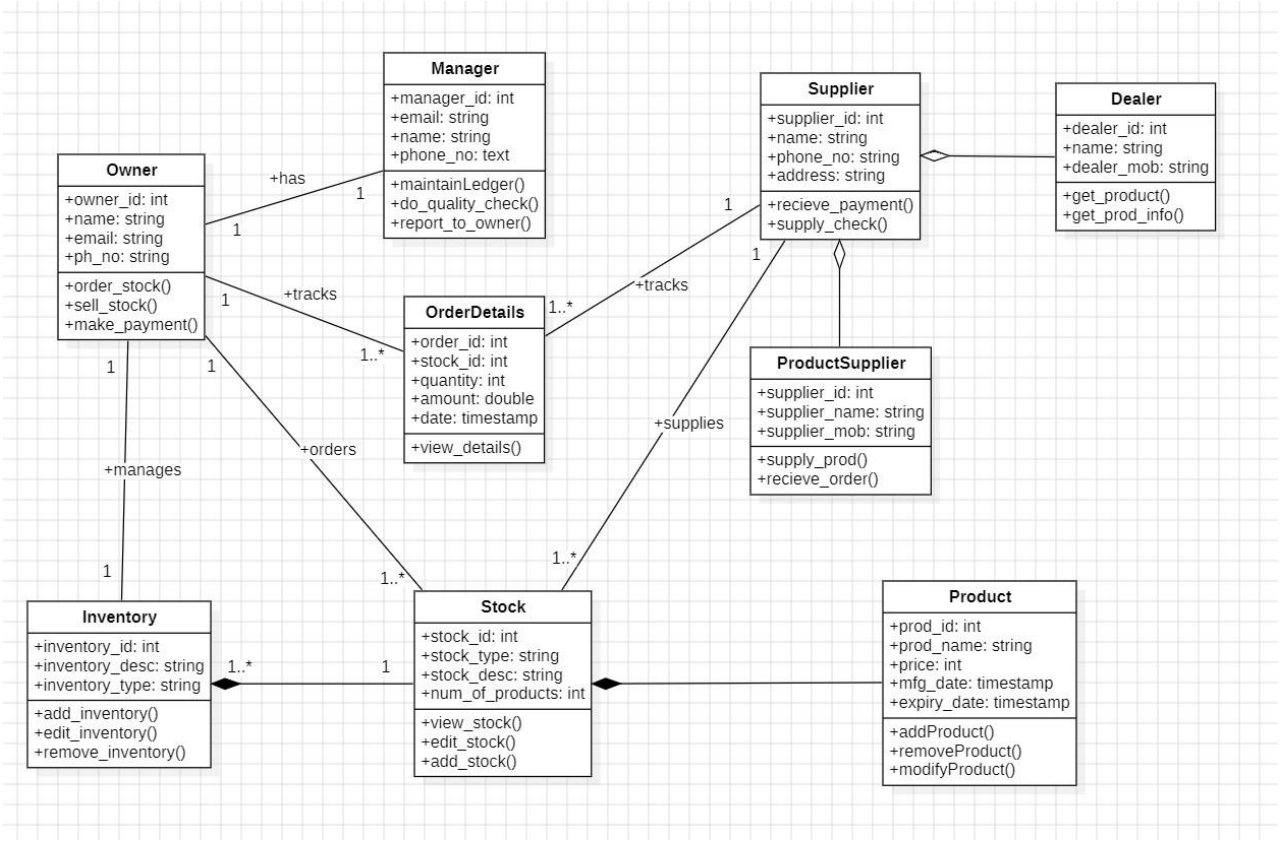


Fig. 4.1

The provided class diagram outlines the structure of a Stock Management System with a focus on the interactions and relationships between various entities involved in stock operations. At the core, the `Owner` class manages the overall system, overseeing `Inventory` and tracking `OrderDetails`. Each `Inventory` consists of multiple `Stock` items, which are detailed with attributes such as `stock_id`, `stock_type`, `stock_desc`, and `num_of_products`. The `Stock` class is central to the system, linking to `OrderDetails` to track stock movements and to `ProductSupplier` for supply chain interactions. The `ProductSupplier` class, in turn, ensures the supply and receipt of stock items. The `Manager` class assists the owner by maintaining ledgers, performing quality checks, and reporting to the owner. `Supplier` and `Dealer` classes handle the external interactions, with suppliers receiving payments and supplying stock, and dealers managing product information and availability. The `Product` class encapsulates individual product details, which are crucial for inventory tracking and management. Overall, the diagram illustrates a comprehensive system for managing stock through detailed record-keeping, efficient inventory tracking, and robust supplier-dealer interactions.

4.4 State Diagram

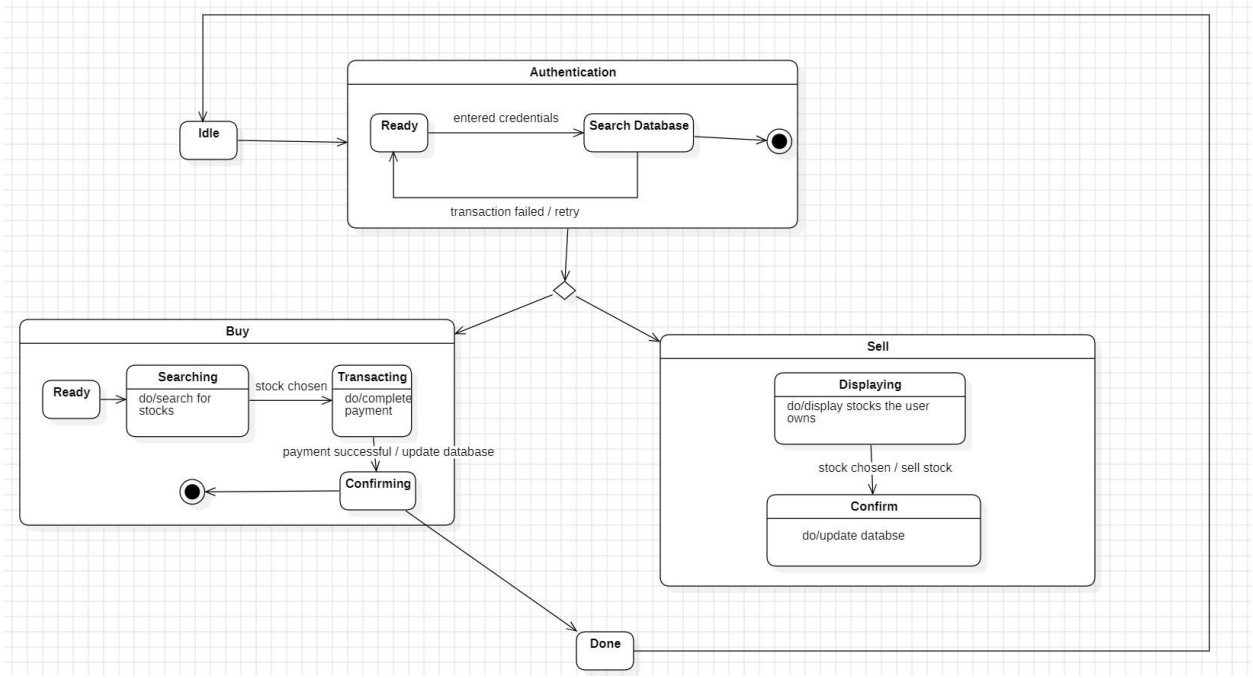


Fig. 4.2

The state diagram for the Stock Maintenance System captures the dynamic behavior of the system, highlighting its various composite states. The "Authentication" state encompasses sub-states "Ready" and "Search Database," illustrating the transition from initial readiness to actively searching the database for user credentials. The "Buy" composite state includes sub-states "Ready," "Searching," "Transacting," and "Confirming," depicting the sequential process from preparing to purchase, searching for items, processing the transaction, and finalizing the purchase. Similarly, the "Sell" composite state consists of "Displaying" and "Confirming," outlining the flow from displaying available stock to confirming the sale. These composite states and their sub-states provide a clear view of the system's operations, showing how it transitions through various stages to handle authentication, buying, and selling activities efficiently. This structured representation ensures a detailed understanding of the system's behavior and the steps involved in key processes.

4.5 Use Case Diagram

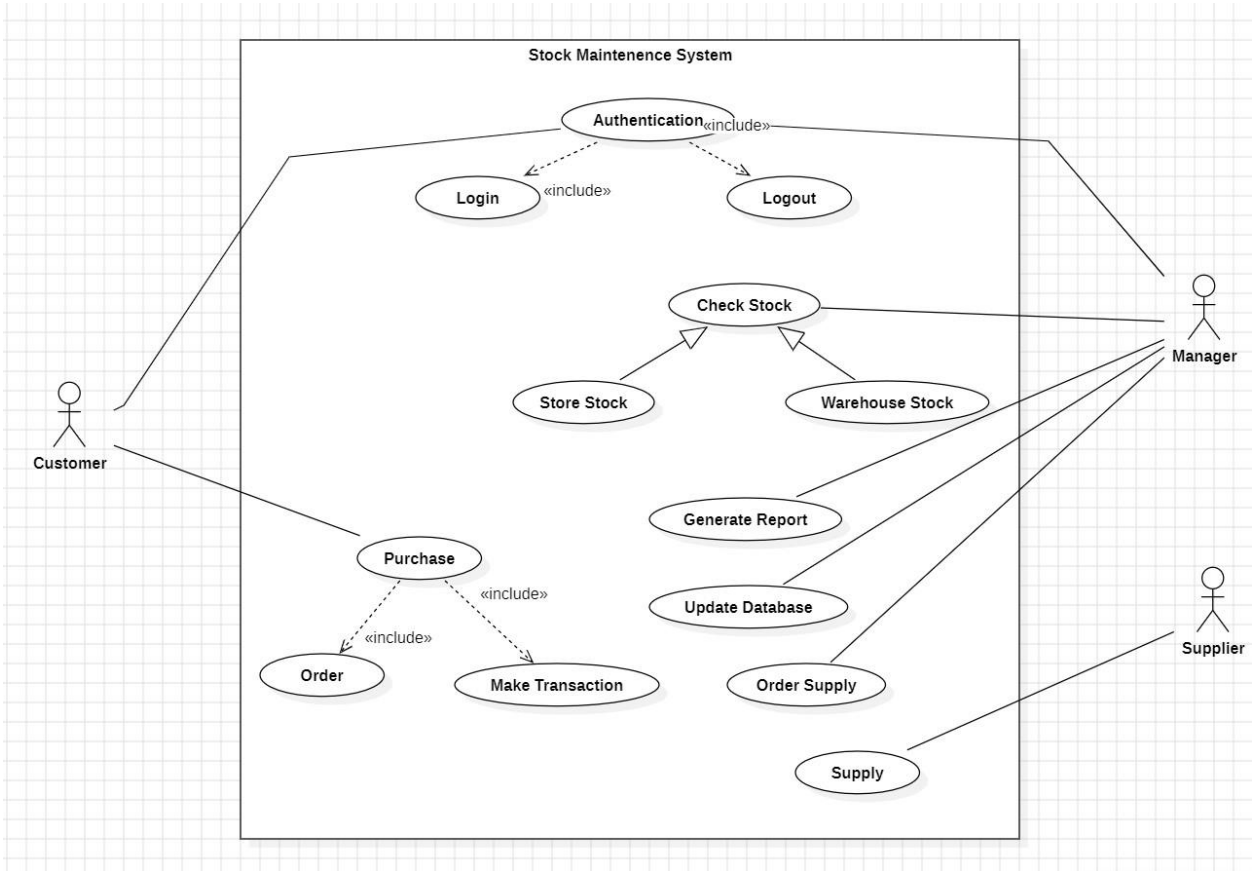


Fig. 4.3

The use case diagram for the Stock Maintenance System illustrates the key interactions between the system and its users, including Customers, Managers, and Suppliers. Customers engage with the system to log in, log out, and make purchases, which involve ordering products and completing transactions. Managers have a broader role, encompassing checking both store and warehouse stock levels, generating reports, updating the database, and ordering supplies to maintain optimal inventory. Suppliers are responsible for fulfilling these orders and supplying the necessary products. Central to the system are the functions of authentication (login/logout), stock checking, purchase processing, and supply management, all contributing to an efficient and accurate stock maintenance workflow. This diagram effectively captures the essential operations and responsibilities, ensuring seamless interaction and stock management within the system.

4.6 Sequence Diagram

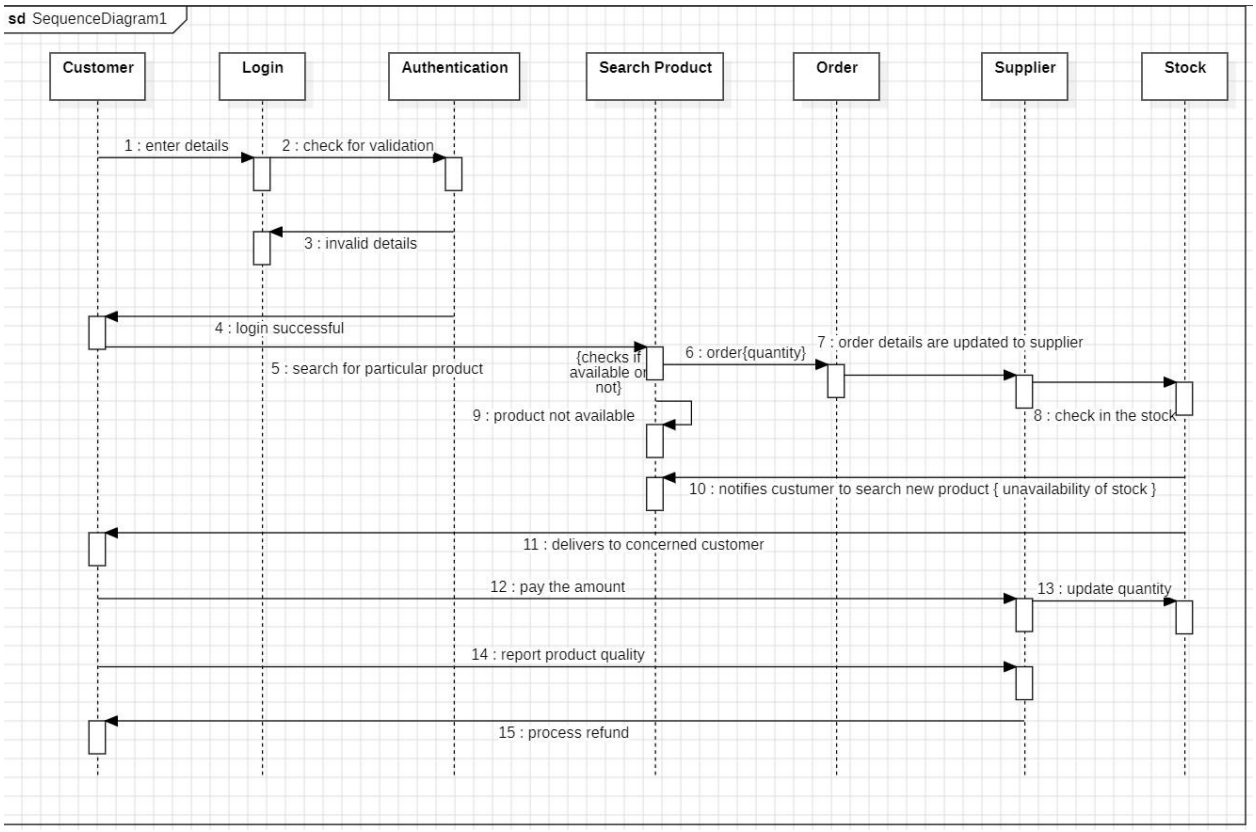


Fig. 4.4

The sequence diagram for the Stock Maintenance System demonstrates the interactions between various components involved in processing a customer's purchase. The diagram begins with the Customer initiating a request, leading to interactions with the Login Authentication component, which verifies the user's credentials. Upon successful authentication, the sequence moves to the Search Products component, where the customer searches for the desired products. Once the products are selected, the Order Supplier component is engaged to fulfill the order by ensuring the availability of stock. The Stock component then updates the inventory to reflect the new purchase. This sequential flow showcases the comprehensive process from customer login to product search, order placement, and stock update, ensuring a seamless and efficient transaction within the system. The diagram provides a clear view of the steps and interactions necessary to complete a purchase, emphasizing the coordination between different system components to maintain accurate stock levels and satisfy customer demands.

4.7 Activity Diagram

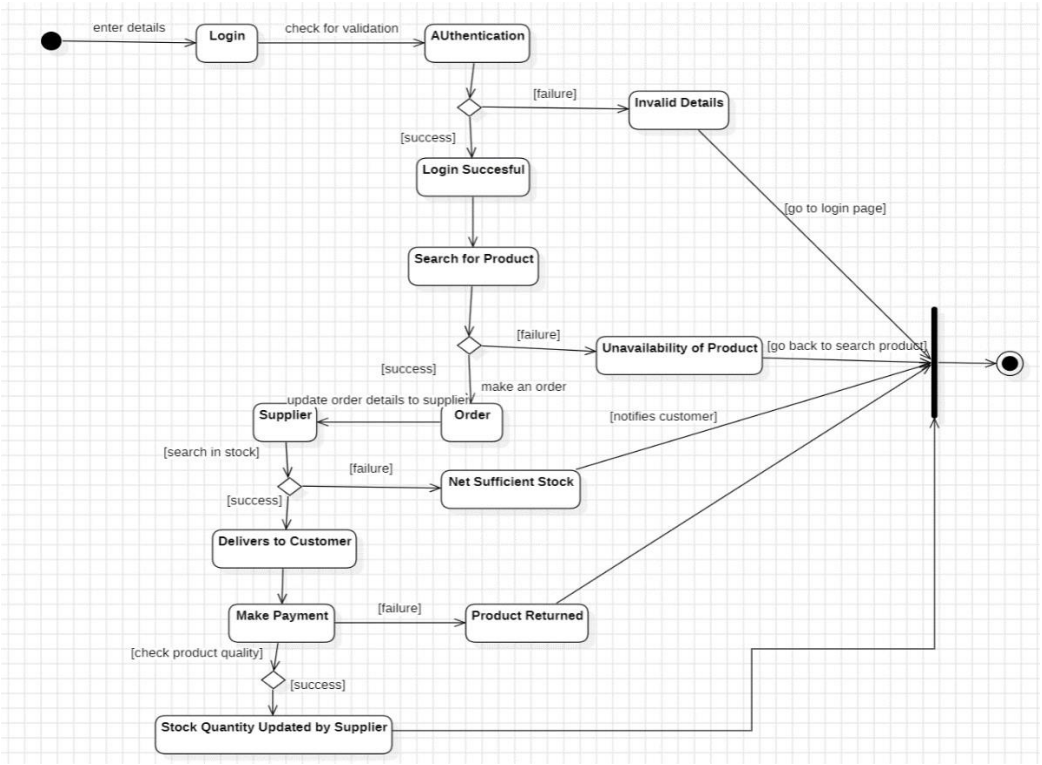


Fig. 4.5

The activity diagram for the Stock Maintenance System outlines the workflow from user login to the final update of stock quantity. The process starts with the "Login" activity, followed by "Authentication," where user details are verified. If the details are invalid, the flow moves to the "Invalid Details" state, prompting the user to re-enter credentials. Upon successful authentication, indicated by the "Login Successful" state, the user proceeds to "Search for Product." If the product is unavailable, the activity transitions to "Unavailability of Product." When the product is available, the next step is "Order," which involves placing an order and checking for sufficient stock. If the stock is insufficient, the order is processed through the "Supplier" to restock. The "Order" state is revisited, and upon confirmation, the product is delivered to the customer ("Delivers to Customer"). Following delivery, the customer "Makes Payment." If the product is returned, it enters the "Product Returned" state. The final activity involves the supplier updating the stock quantity ("Stock Quantity Updated by Supplier"), ensuring accurate inventory levels. This diagram effectively maps the detailed steps and decisions involved in managing customer orders, from login to inventory updates, ensuring a comprehensive and systematic approach to stock maintenance.

5. PASSPORT AUTOMATION SYSTEM

5.1 Problem Statement

Passport Automation System is used in the effective dispatch of passport to all of theapplicants. This system adopts a comprehensive approach to minimize the manual work andschedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name,address etc.,) filled by the applicant whose testament is verified for its genuineness by thePassport Automation System with respect to the already existing information in the database. This forms the first and foremost step in the processing of passport application. After the first round of verification done by the system, the information is in turn forwarded to theregional administrator's (Ministry of External Affairs) office. The application is then processed manually based on the report given by the system, and any forfeiting identified can make the applicant liable to penalty as per the law.The system forwards the necessary details to the police for its separate verification whose report is then presented to the administrator. After all the necessary criteria have been met, the original information is added to the database and the passport is sent to the applicant.

5.2 Software Requirement Specification

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the requirements for the development of a Passport Automation System (AMS). It provides a detailed description of the system's functionalities, interfaces, performance requirements, and constraints to guide stakeholders, developers, and users throughout the development process.

1.2 Problem Statement

Passport Automation System is used in the effective dispatch of passport to all of the applicants. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its

genuineness by the Passport Automation System with respect to the already existing information in the database.

1.3 Scope of this Document

The System provides an online interface to the user where they can fill in their personal The authority concerned with the issue of passport can use this system to reduce his workload and process the application in a speedy manner. Provide a communication platform between the applicant and the administrator.Transfer of data between the Passport Issuing Authority and the Local Police for verification of applicant's information.

1.4 Overview

SRS includes two sections overall description and specific requirements - Overall description will describe major role of the system components and inter-connections. Specific requirements will describe roles & functions of the actors

2. General Description

2.1 Product Perspective

The PAS acts as an interface between the 'applicant' and the 'administrator'. This system tries to make the interface as simple as possible and at the same time not risking the security of data stored in. This minimizes the time duration in which the user receives the passport.

2.2 Software Requirements

• Front End Client - The applicant and Administrator online interface is built using Microsoft Visual Basic 6.0.

• Back End – MS Access database.

2.3 Hardware Interface

The server is directly connected to the client systems. The client systems have access

to the database in th**e server.**

2.4 System Functions

• Secure Registration of information by the Applicants.

• Message box for Passport Application Status Display by the Administrator.

• Administrator can generate reports from the information and is the only authorized personnel to add the eligible application information to the database.

2.5 User Characteristics

• Applicant - They are the people who desires to obtain the passport and submit the information to the database.

• Administrator - He has the certain privileges to add the passport status and to approve the issue of passport. He may contain a group of persons under him to verify the documents and give suggestion whether or not to approve the dispatch of passport.

• Police - He is the person who upon receiving intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present. He has been vetoed with the power to decline an application by suggesting it to the Administrator if he finds any discrepancy with the applicant. He communicates via this PAS.

3. Functional Requirements

3.1 Applicant Functions • The system shall allow applicants to register for an account using a secure method.

- The system shall provide an online application form that captures all mandatory applicant information.
- The system shall allow applicants to upload required documents in various formats (e.g., PDF, JPG).
- The system shall integrate with a secure payment gateway for online fee collection.
- The system shall allow applicants to schedule appointments for biometric data capture (if applicable).
- The system shall provide a real-time dashboard for applicants to track application status.
- The system shall enable secure communication between applicants and administrators through an internal messaging system.

3.2 Administrator Functions

- The system shall provide administrators with a secure login mechanism.
- The system shall display a list of submitted applications for review.
- The system shall allow administrators to approve or reject applications with justification.
- The system shall integrate with government databases for verification of applicant information.
- The system shall provide secure storage and management of uploaded documents.
- The system shall facilitate passport printing and issuance management.
- The system shall allow administrators to create and manage user accounts (applicants & administrators).
- The system shall provide administrators with tools for system configuration and report generation.

6. Design Constraints

- The applicants require a computer to submit their information.
- Although the security is given high importance, there is always a chance of intrusion in the web world which requires constant monitoring.
- The user has to be careful while submitting the information. Much care is required.

• The Applicants and Administrator must have basic knowledge of computers and English Language.

• The applicants may be required to scan the documents and send.

7. Non-Functional Attributes

7.1 Performance Requirements

- The system response time for applicant actions (e.g., form submission, document upload) should be within 5 seconds on average.
- The system should be able to handle a peak load of X applications per hour (define based on expected volume).

7.2 Security Requirements

- The system shall employ industry-standard security practices to protect user data and prevent unauthorized access.
- All data transmission shall be encrypted using secure protocols (e.g., HTTPS).
- The system shall implement strong user authentication mechanisms.
- The system shall have provisions for user access control and role-based permissions.

7.3. Usability Requirements

- The user interface shall be intuitive and user-friendly for applicants with varying technical skills.
- The system shall provide clear instructions and informative messages to guide users through the application process.
- The system shall be accessible to users with disabilities in accordance with relevant accessibility standards.

8. Preliminary Schedule and Budget

8.1 Schedule

- Phase 1: Requirements Gathering and Analysis (1 month)
- Phase 2: Design (1 month)

- Phase 3: Development (3 months)
- Phase 4: Testing (1 month)
- Phase 5: Deployment (1 month)
- Total Duration: 7 months

8.2 Budget

- Personnel Costs: $40,000
- Software and Tools: $15,000
- Hardware Costs: $5,000
- Miscellaneous Expenses: $5,000
- Total Estimated Cost: $65,000

5.3 Class Diagram

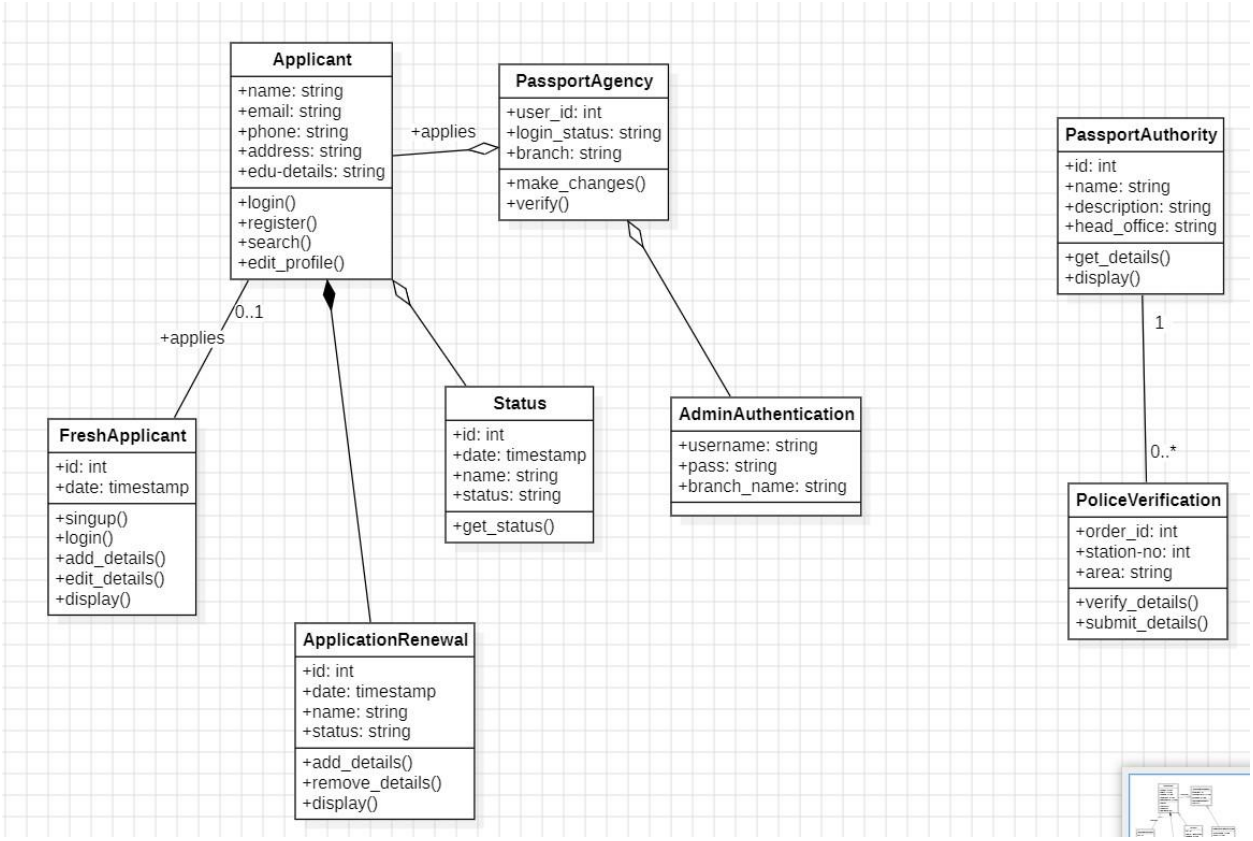


Fig 5.1

- APPLICANT-The applicant has attribute such as name and password and operations are login, givedetails and logout. The applicant login and fill the details that are required for applying the passport . After applying the person can view the status of the passport verification process.

- THE DATABASE-The database has attributed such as name and operation is store. The purpose is to store the data.

  REGIONAL ADMINISTRATOR- The regional administrator has attribute such as

  name and operation are get details, verify details and send. The regional administrator get the details form database and verify with their database
- PASSPORT ADMINISTRATOR-The passport administrator has attributed such as name and operation are get details, verify details and issue. The passport administrator get the details form database and verify with their database , update the verification and issue the passport
- THE POLICE-The police has attribute such as name and operation are get details, verify details and send. The police get the details form database and verify with their database , update the verification in the database

5.4 State Diagram

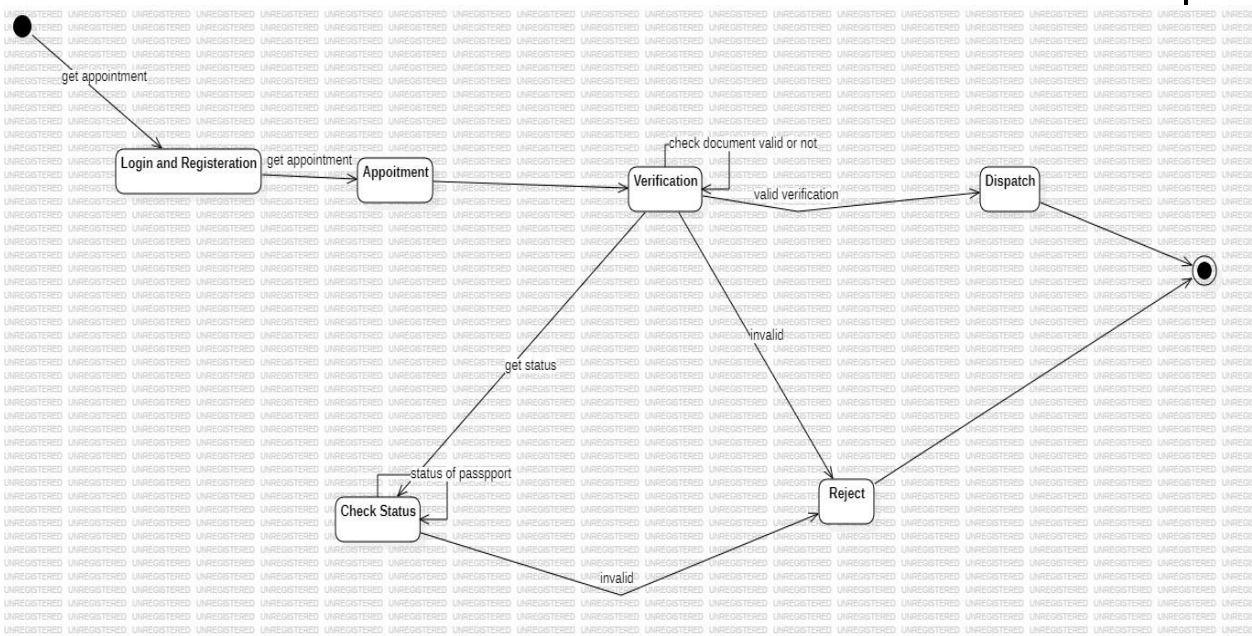


Fig 5.2

The state chart diagram contains the states in the rectangle boxes and starts in indicated by the dot and finish is indicated by dot encircled. The purpose of state chart diagram is to understand the algorithm in the performing method. The states of the passport automation system are denoted in the state chart diagram. Login state represent authentication for login the passport automation system. In this state, it checks whether the applicant has provided all the details that is required.

Police, regional administrator and passport administrator get necessary details and verification of the applicant are denoted from the Get detail state and verification state.

5.5 Usecase Diagram

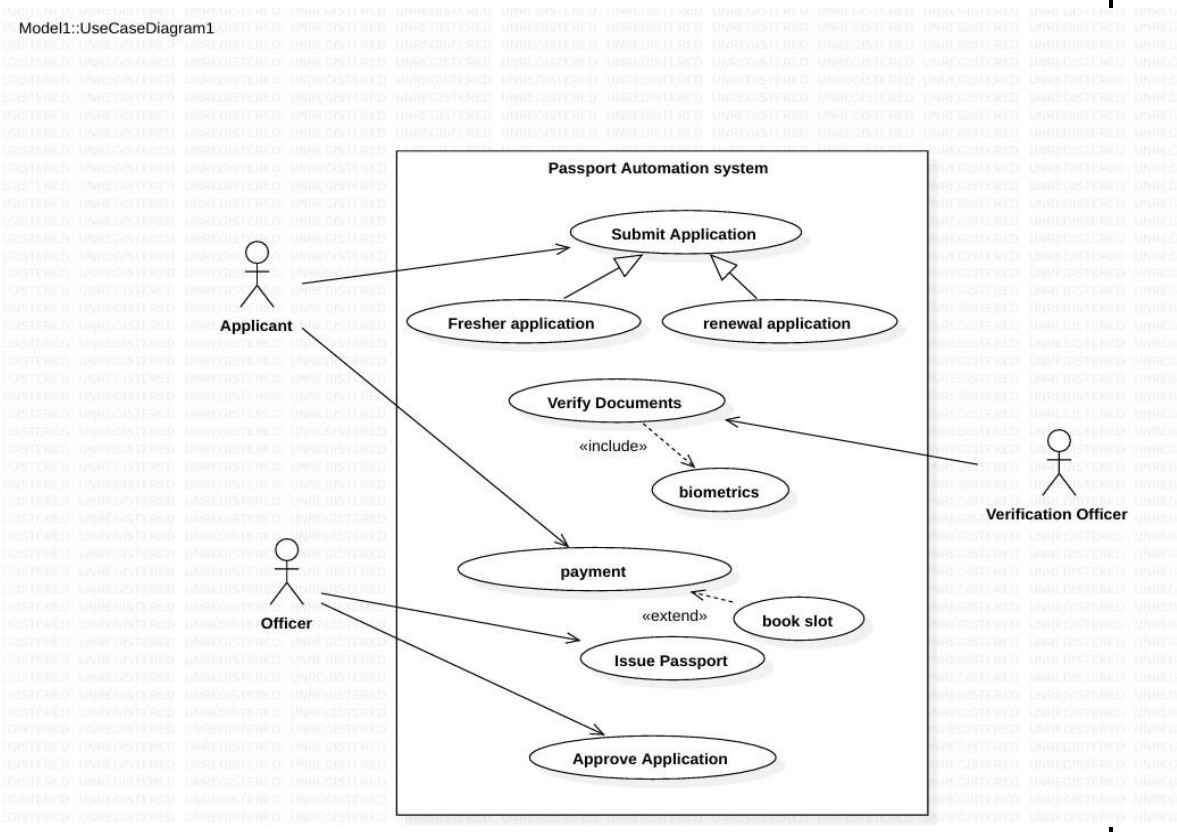


Fig 5.3

Use case is shown as an ellipse containing the name of use case. An actor is shown as a stick figure with the name below it. Use case diagram is a graph of actors. The actors in use case diagram are Applicant, regional administrator, database, passport Administrator, and Police. The use cases are Login, give details, logout, collect details, verification, issue. The actors use the use case are denoted by the arrow. The login use case checks the username and password for applicant, regional administrator, passport administrator and police. The submit details use case is used by the applicant for submitting his detailsThe check status use case is used by the applicant for checking the status of the application process.The get details, verify and store verification use case is used by passport administrator, regional administrator, and police. The details use case is used for getting the details form the database for verification. The verify use case is used for verifying the details by comparing the data in the database. The store verification use case is to update the data

in the database And finally the issue passport use case is used by the passport administrator for issuing passport who's application verified successfully by all the actor

5.6 Sequence Diagram

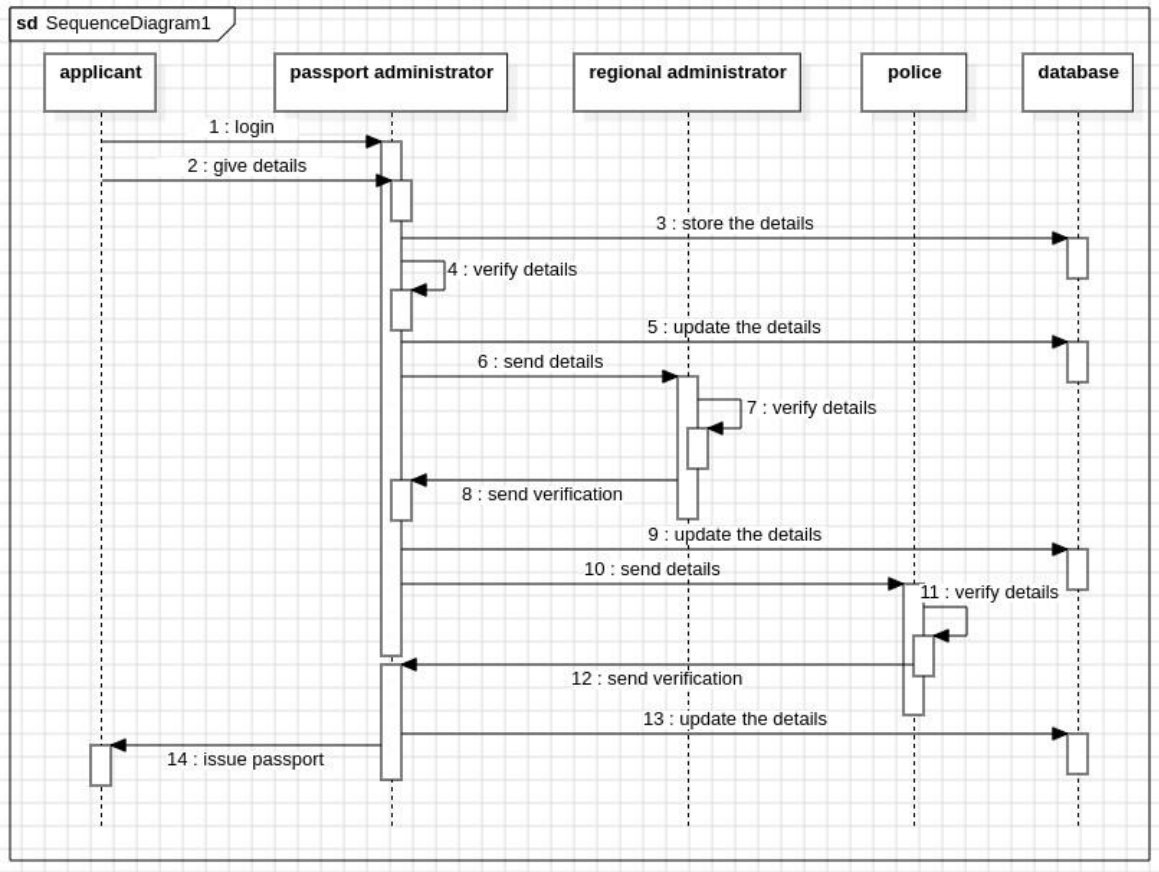


Fig 5.4

The sequence diagram shows the interactions between the applicant, passport administrator, regional administrator, police and the database. The applicant logs in and gives their details to the passport administrator, who then stores and verifies those details, and updates the details in the database. The same details are sent to the regional administrator, who then verifies the details and then sends the verification back. The details are updated in the database and those details is sent to the police, who then verifies the details and sends a verification back. The details are then updated in the database by the passport administrator, who then issues the passport to the applicant.

5.7 Activity Diagram

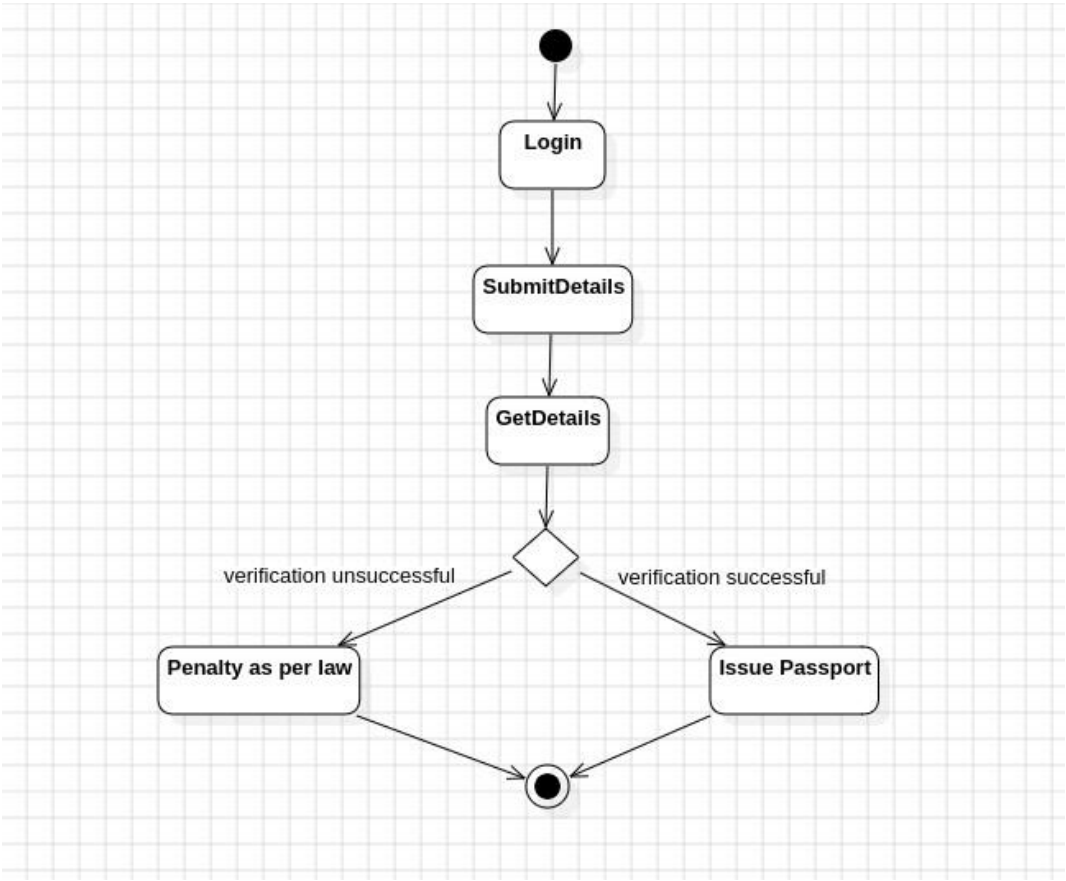


Fig 5.5

An activity diagram is a variation or special case of a state machine in which the states or activity representing the performance of operation and transitions are triggered by the completion of operation. The purpose is to provide view of close and what is going on inside a use case or among several classes. An activity is shown as rounded box containing the name of operation. The activities in the passport automation system are login, submit details, get details,issue passport and penalty and verification. In the login activity applicant give username and password and then login into the passport automation system after then fill the details that are required for application. After the verification procedure completed successfully the passport is issued to the applicant.