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:

NANDANA KISHORE C NAIR

1BM23CS364

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 NANDANA KISHORE C NAIR (1BM23CS364) during the 5th Semester August 2025 - December 2025.

Signature of the Faculty Incharge:

Dr Adarsha Sagar HV
Assistant Professor,
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

Problem Statement

Managing reservations, customer data, billing, payment information, room scheduling, and housekeeping tasks manually is time-consuming and prone to errors. A digital Hotel Management System (HMS) is required to streamline operations, enhance the customer experience, and optimize hotel resources.

Software Requirements Specification

1. Introduction

1.1 Purpose

This document specifies the software requirements for the development of a Hotel Management System (HMS). It is intended for developers, testers, project managers, and other stakeholders to serve as a reference throughout the project lifecycle.

1.2 Scope

The HMS will provide a comprehensive, web-based solution to automate core hotel operations. It will cater to three primary user groups:

1. Customers: Can browse room availability, make online bookings, make payments, and submit feedback.
2. Hotel Staff: Can manage reservations, allocate rooms, track housekeeping tasks, and handle billing.
3. Administrators: Can generate detailed reports on occupancy, revenue, and staff performance to analyze business success.
   The system aims to reduce manual errors, lower operational costs, and improve overall efficiency.

1.3 Overview

The HMS is a modular system that will include features for Reservation Management, Billing and Payments, Customer Information Management, Room Allocation, Housekeeping, Feedback Collection, and Reporting. It will provide a dedicated booking portal for customers, an operations panel for hotel staff, and a dashboard for administrators.

2. General Description

The HMS is designed for use by administrators, hotel staff, and customers.

1. Customers can book rooms online or process walk-in reservations and complete payments.
2. Hotel Staff can manage reservations, assign rooms, and oversee housekeeping schedules.
3. Administrators have access to dashboards for monitoring revenue and key performance indicators.
   The system is designed to implement primary hotel operations while minimizing errors, improving efficiency, and enhancing the customer experience.

3. Functional Requirements

FR1: Book, modify, or cancel room reservations.

FR2: Store customer information with secure data handling.

FR3: Assign rooms automatically based on availability and customer requirements.

FR4: Generate accurate bills and process payments via cash, credit card, or online transfers.

FR5: Track room cleaning status and manage other housekeeping tasks.

FR6: Generate reports on occupancy, revenue, and staff performance for administrators.

4. Interface Requirements

a) User Interfaces:

Customer Portal: For online booking, payment, and feedback.

Staff Operations Panel: To manage reservations, room status, and billing.

Admin Dashboard: For viewing reports on revenue and performance.

b) Hardware Interfaces: The system must be accessible on desktops, tablets, and POS terminals.

c) Software Interfaces: The system shall integrate with external payment gateways.

d) Communication Interfaces: Requires internet connectivity and should support LAN for local hotel systems.

5. Performance Requirements

a) The system must process reservation requests within 3 seconds.

b) Must support up to 500 concurrent users during peak booking hours.

c) Must ensure 99.9% system uptime.

d) Must maintain a maximum error rate of less than 1% for online transactions.

6. Design Constraints

a) Must comply with data privacy laws to protect customer data.

b) Must use a relational database management system.

c) Must be accessible via standard web browsers.

7. Non-Functional Requirements

a) Security: Must protect sensitive customer and payment data.

b) Reliability: Must be fault-tolerant with automatic backup systems.

c) Usability: Must provide a single, intuitive interface for both staff and customers.

d) Portability: Must be accessible on web and mobile platforms.

e) Scalability & Maintainability: Must support multiple hotel branches and use a modular design for easy future updates.

8. Preliminary Schedule and Budget

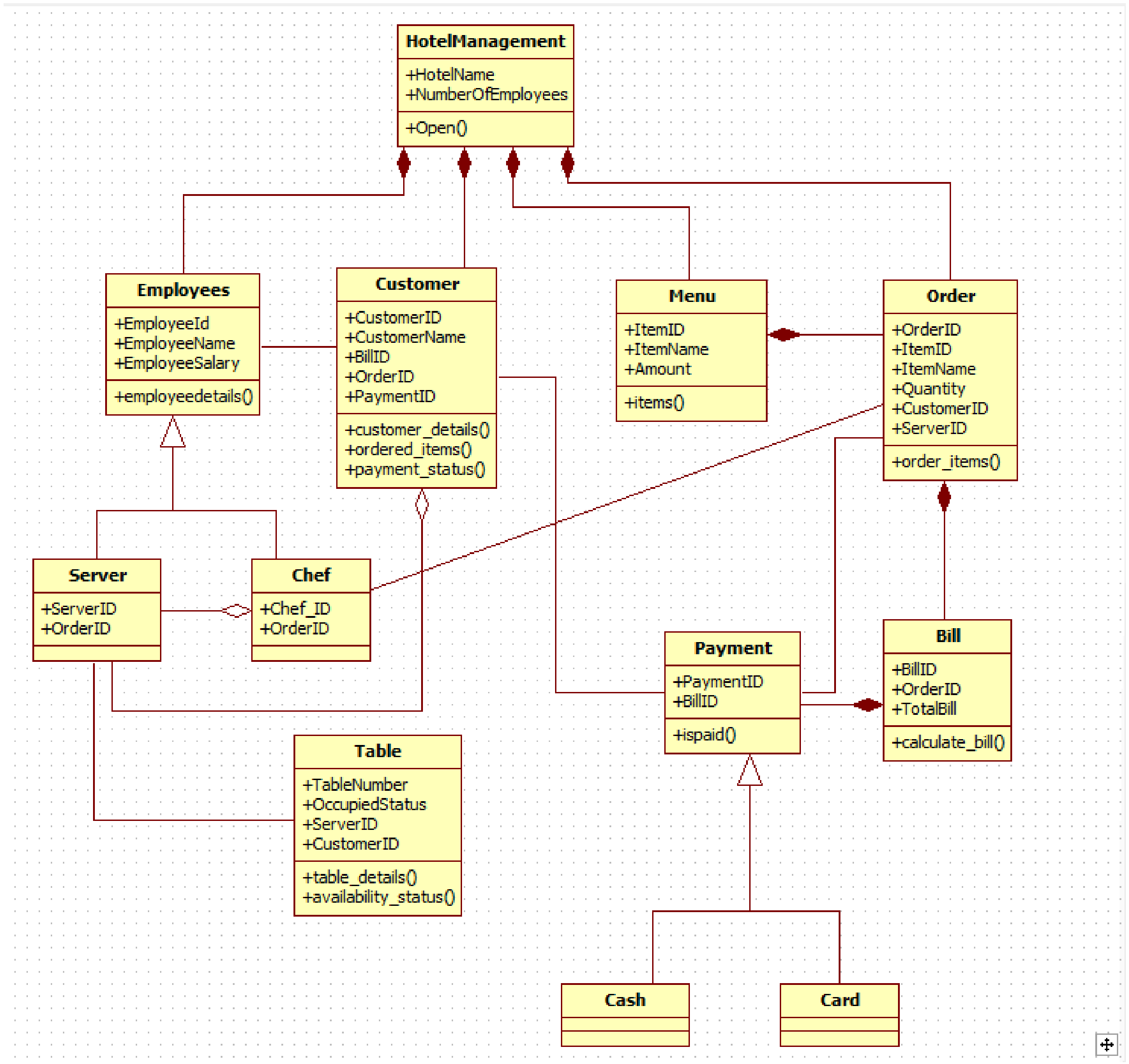
a) Development Timeline: 6 months

1. Analysis & Design: 2 months
2. Development: 3 months
3. Testing and Deployment: 1 month

b) Budget Estimate: Approximately $50,000 USD for design, development, testing, and deployment.

Class Diagram

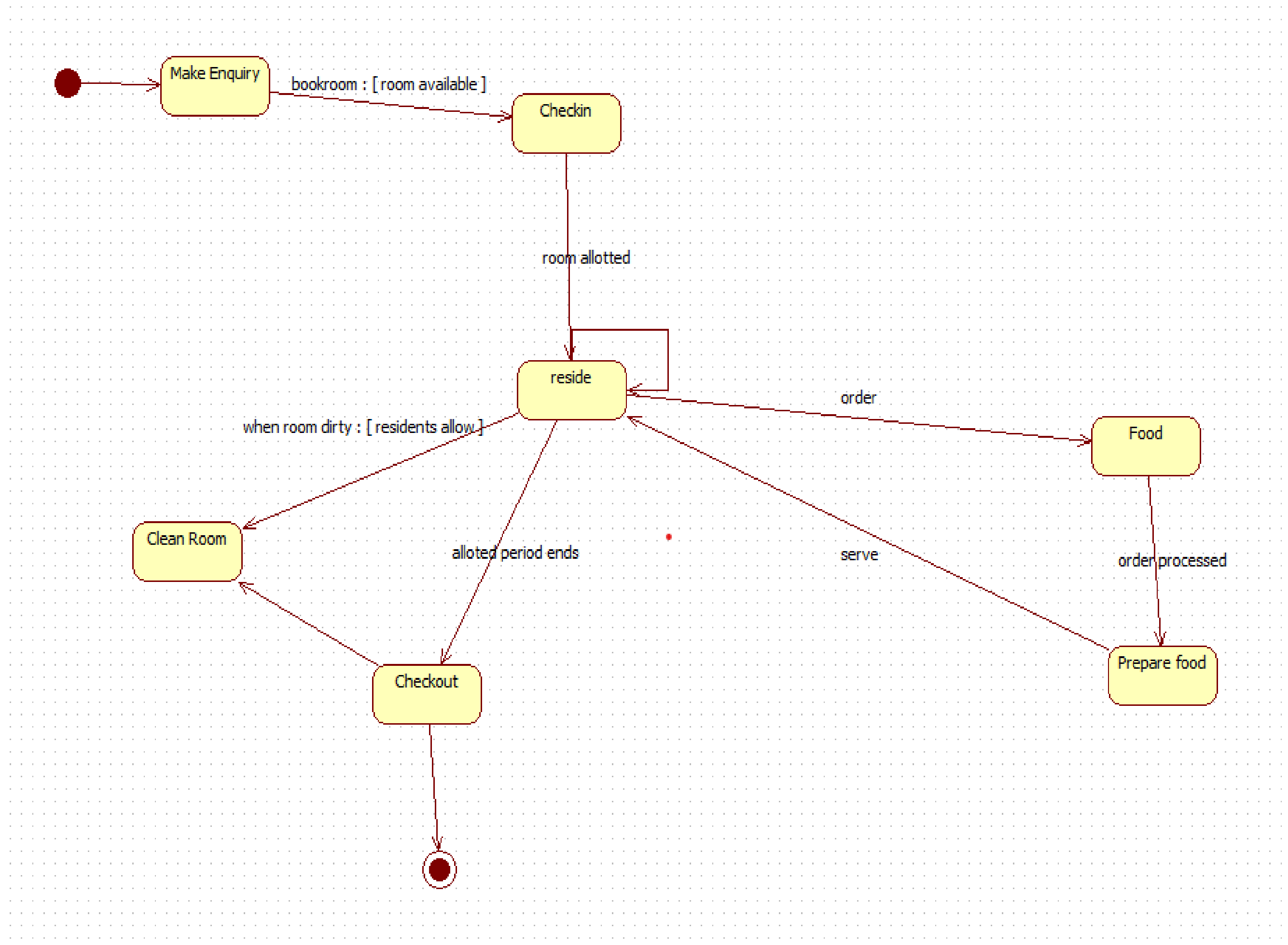
Models hotel entities including Hotel, Employee, Customer, Menu, Order, and Payment with their attributes and operations. Relationships include inheritance between Payment and its subclasses Cash/Card, composition between Hotel and Employees, and associations linking Customer to Orders and Bills.



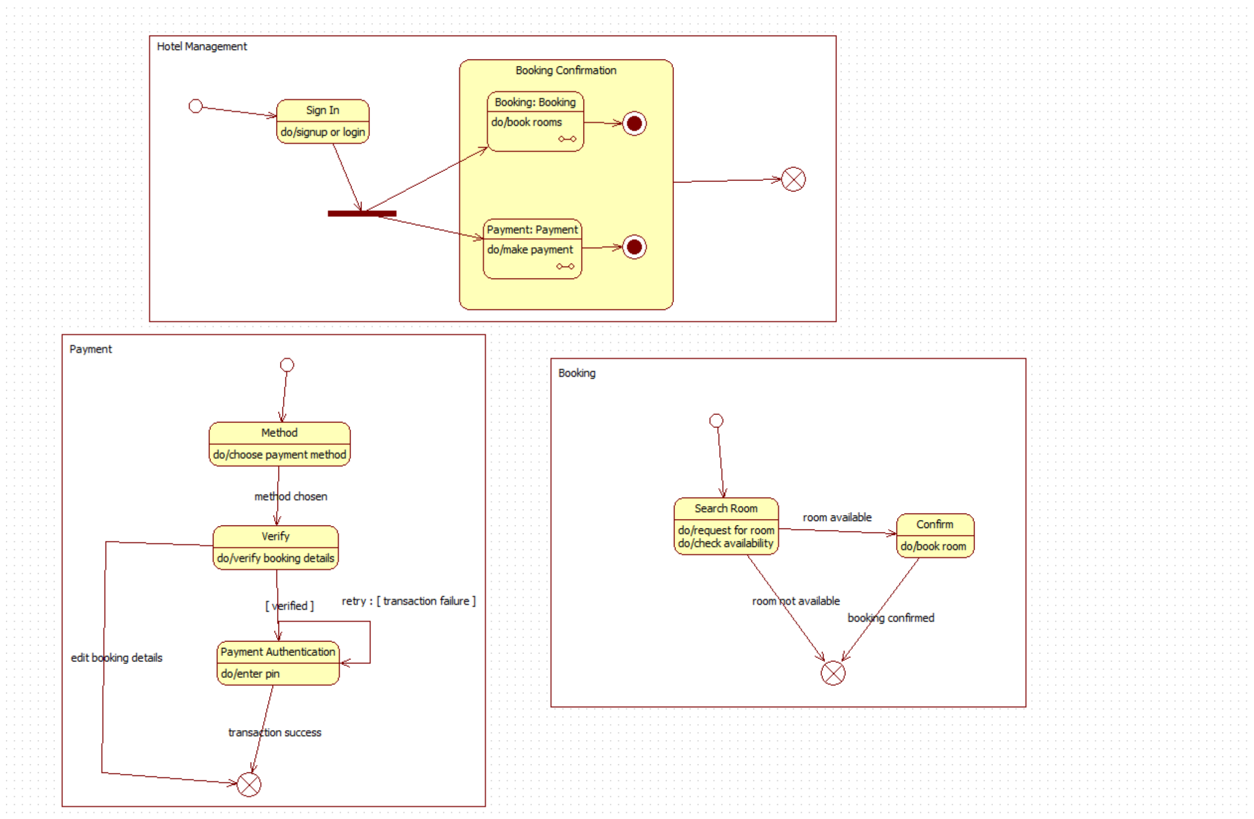
State Diagrams

Tracks the room booking lifecycle from initial enquiry through check-in, occupancy, and final checkout. Includes conditional transitions like room availability checks and events such as food ordering. Features concurrent substates managing simultaneous payment processing and room allocation workflows.

Simple State Diagram



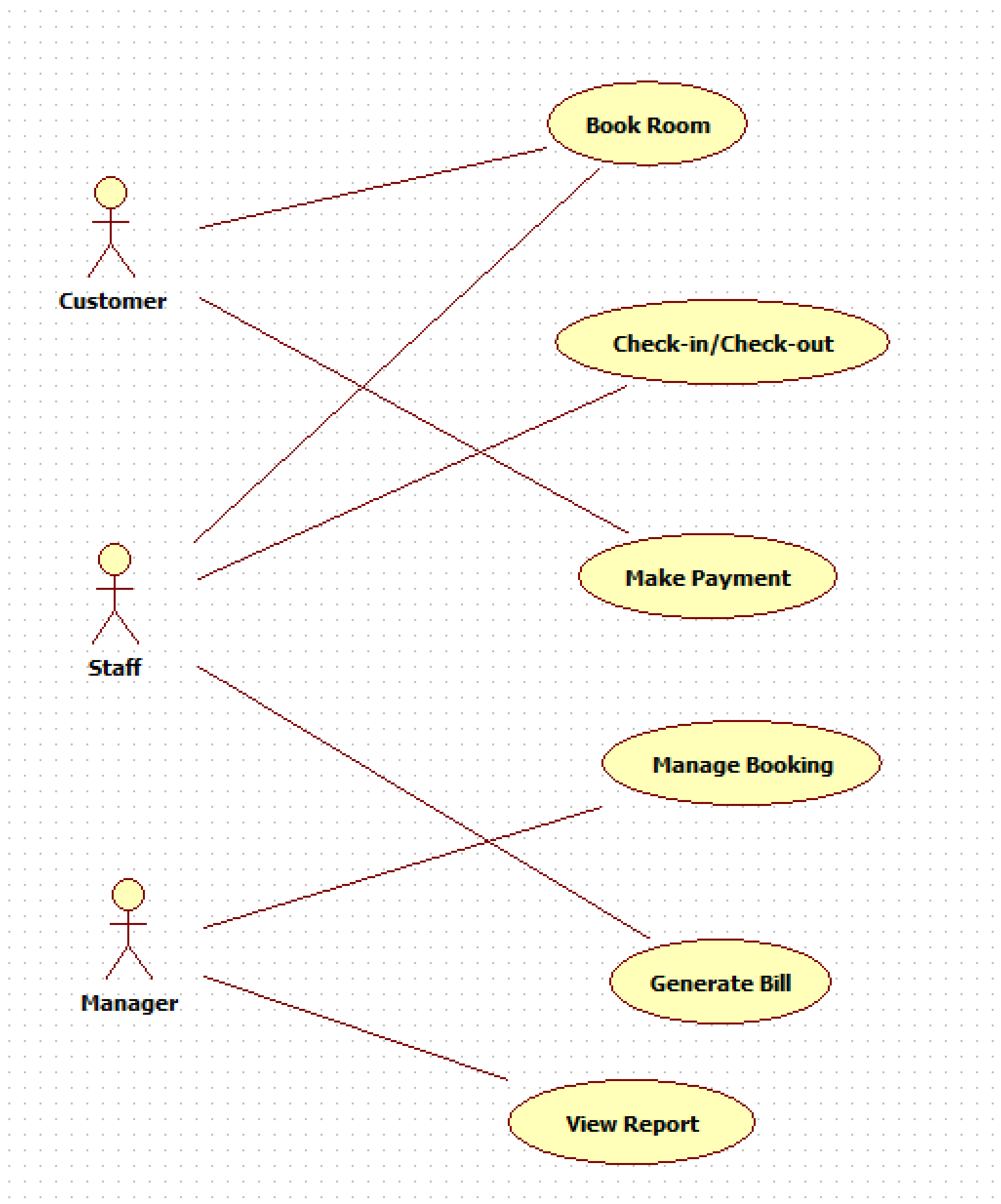
Advance State Diagram



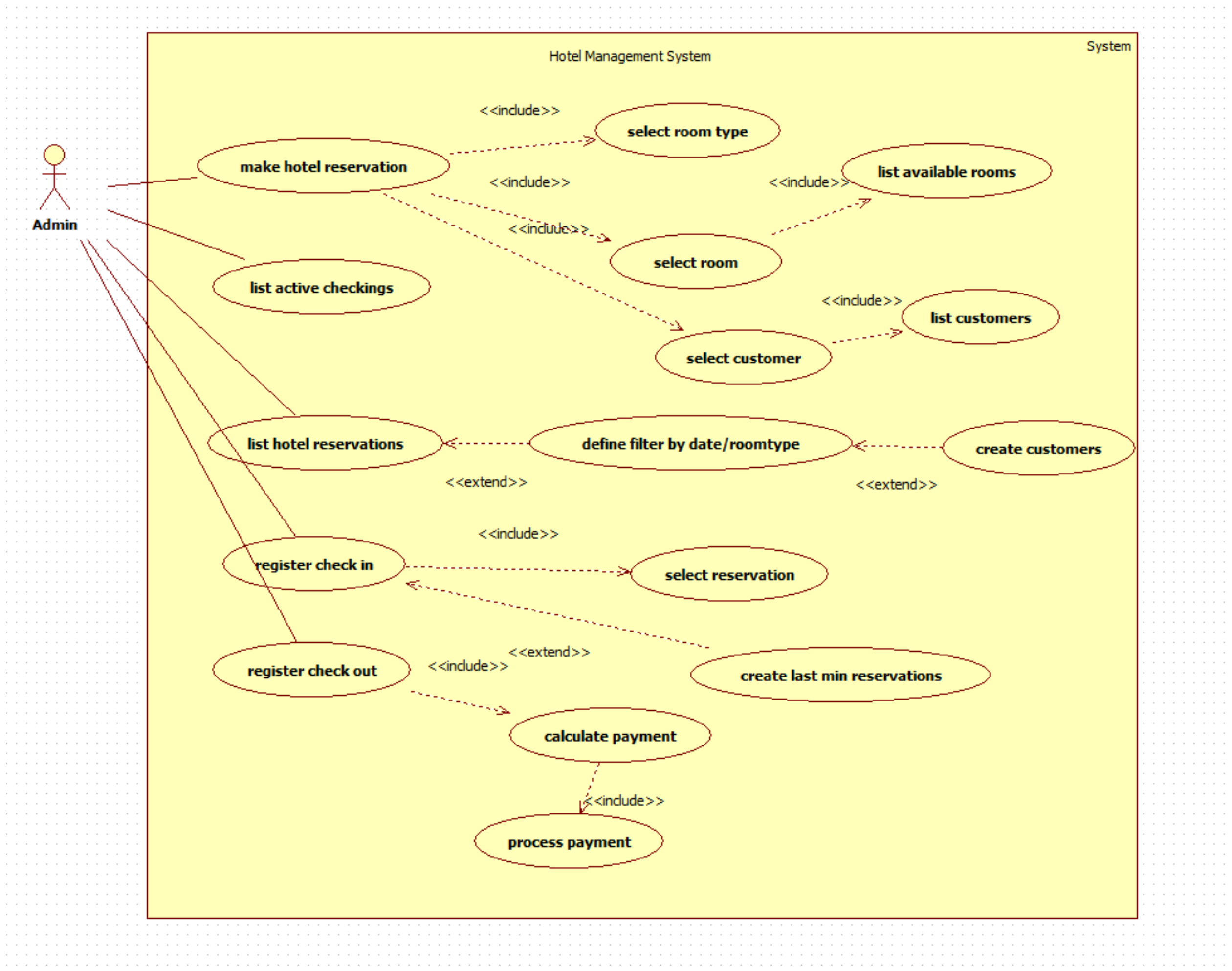
Use Case Diagrams

Illustrates system interactions for Customers making bookings and payments, Staff managing reservations, and Administrators generating reports. Demonstrates relationship dependencies where payment processing extends booking functionality and report generation includes revenue analysis components.

Simple Use Case Diagram



Advance Use Case Diagram



Sequence Diagram

Details the chronological interaction between Customer, Receptionist, and System during room reservation. Sequences include availability checks, booking confirmation, payment processing, and receipt generation. Captures synchronous communication patterns and object collaboration timing.

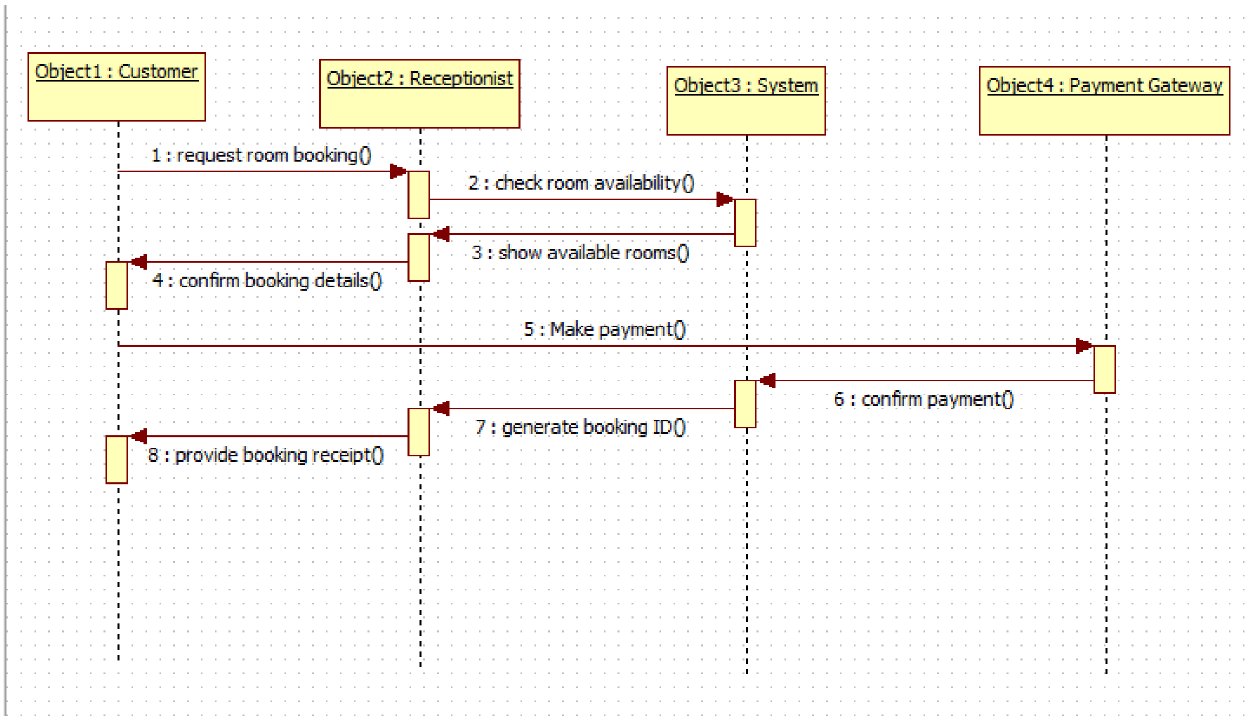
<u>Scenario 1: Successful Online Booking</u>

1. Pre-condition: User is logged into the system, available rooms exist for selected dates
2. Main Flow: User selects check-in and check-out dates → system displays available room types with prices → user selects deluxe room → system shows room details and amenities → user enters guest information → system calculates total cost including taxes → user provides payment details → system processes payment → system generates booking confirmation with reservation ID → system sends confirmation email to user
3. Post-condition: Room status updated to booked, inventory reduced, booking record created in database, confirmation notification sent

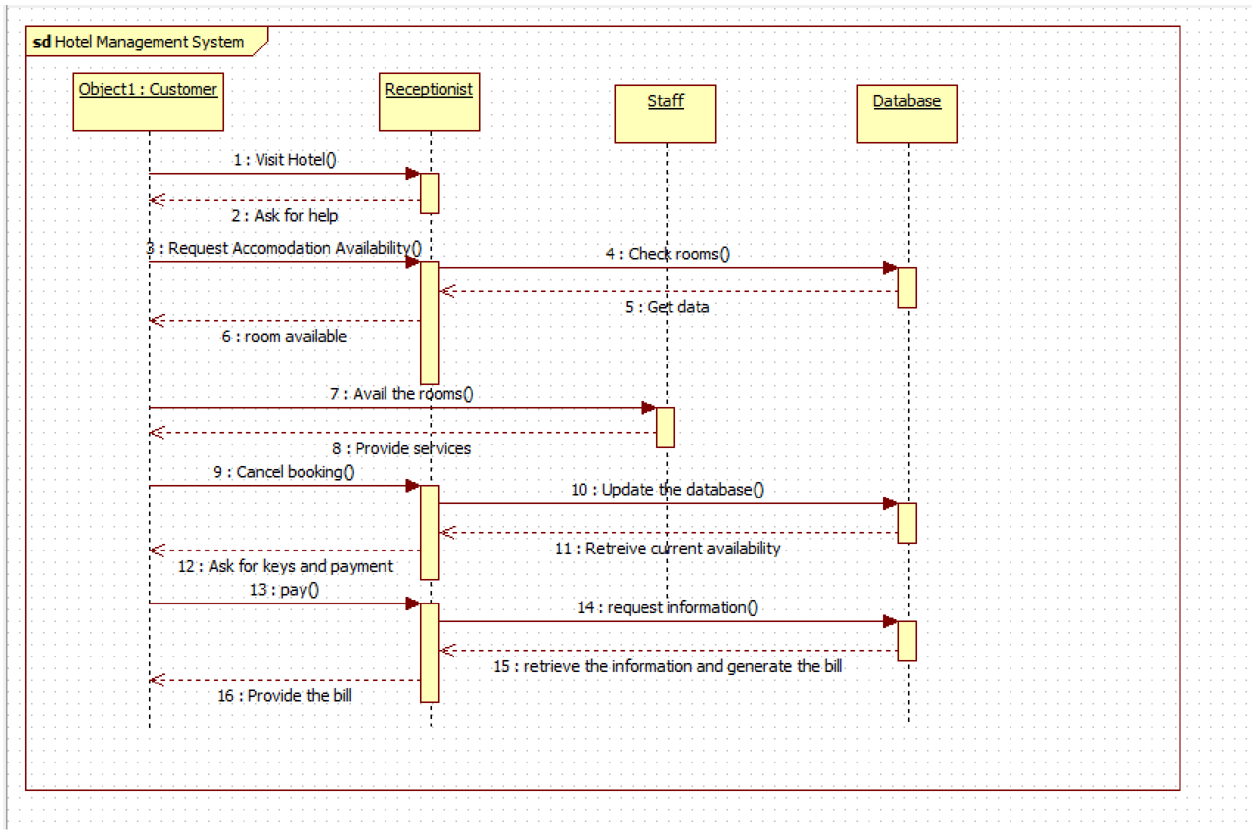
<u>Scenario 2: Booking Modification with Room Upgrade</u>

1. Pre-condition: User has existing confirmed booking, user is authenticated in system
2. Main Flow: User accesses booking management section → system displays current reservation details → user selects modify booking option → user changes dates to extend stay → system checks room availability for new dates → system shows upgrade options for extended period → user selects suite upgrade → system calculates price difference → user confirms changes and pays additional amount → system updates booking record → system sends modification confirmation email
3. Post-condition: Original booking modified, room type upgraded, additional payment recorded, new confirmation generated

Simple Sequence Diagram

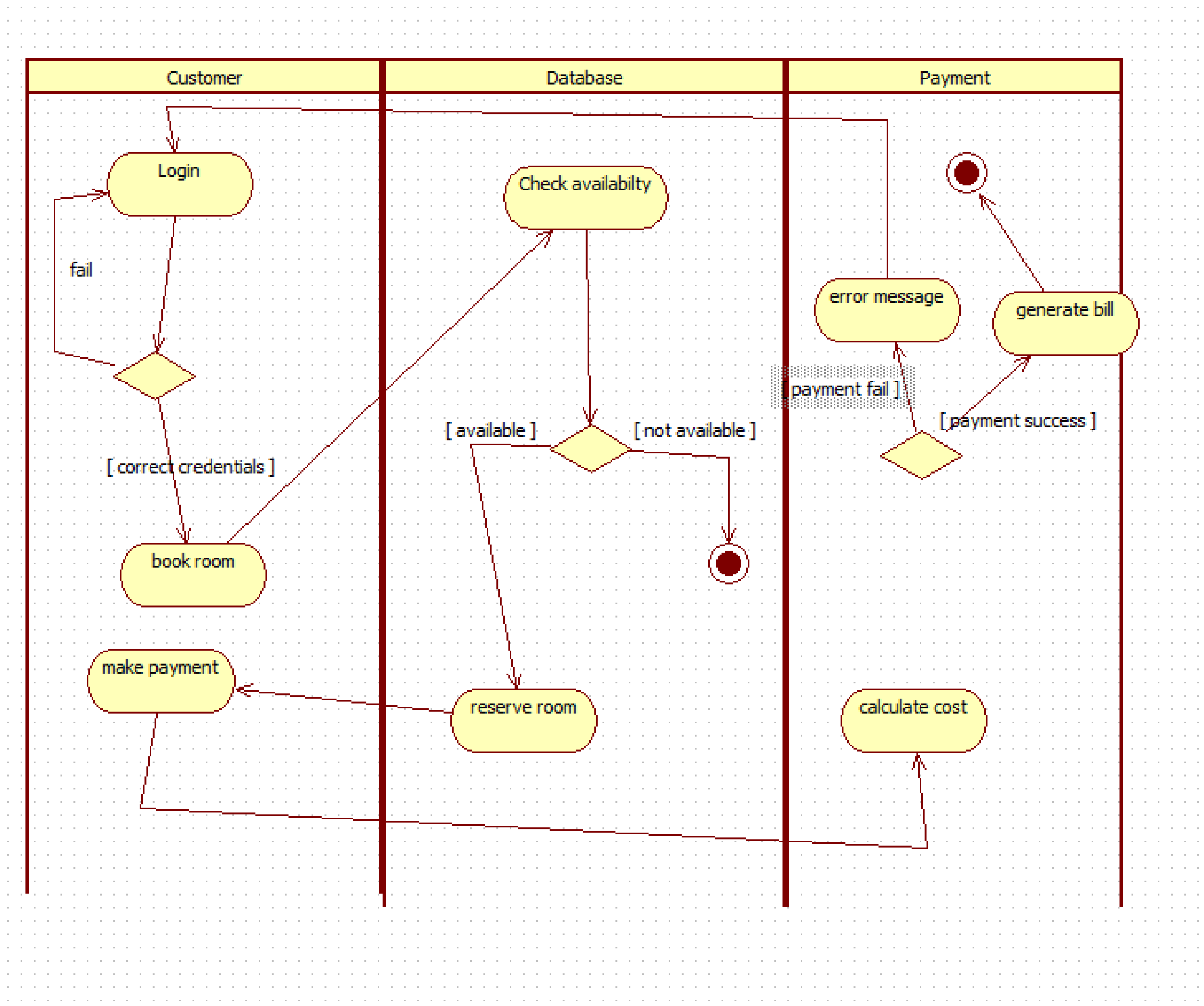

Object1 : Customer
Object2 : Receptionist
Object3 : System
Object4 : Payment Gateway
1 : request room booking()
2 : check room availability()
3 : show available rooms()
4 : confirm booking details()
5 : Make payment()
6 : confirm payment()
7 : generate booking ID()
8 : provide booking receipt()

Advance Sequence Diagram


sd Hotel Management System
Object1 : Customer
Receptionist
Staff
Database
1 : Visit Hotel()
2 : Ask for help
3 : Request Accomodation Availability()
4 : Check rooms()
5 : Get data
6 : room available
7 : Avail the rooms()
8 : Provide services
9 : Cancel booking()
10 : Update the database()
11 : Retreive current availability
12 : Ask for keys and payment
13 : pay()
14 : request information()
15 : retrieve the information and generate the bill
16 : Provide the bill

Activity Diagram

Maps the complete room booking workflow from user authentication through room selection, payment authorization, to final confirmation. Swimlanes distribute activities among Customer, Database, and Payment subsystems, showing parallel processing and error handling pathways.



2. Credit Card Processing System

Problem Statement

The growing volume of online and offline transactions requires a secure and efficient credit card processing system. Current methods face significant challenges including inadequate fraud detection mechanisms, transaction processing delays, and complex reconciliation processes. There is an urgent need for a comprehensive system that ensures fast, secure, and accurate authorization, settlement, and support of credit card transactions while maintaining regulatory compliance and providing robust fraud prevention capabilities.

Software Requirements Specification

1. Introduction

1.1 Purpose

This document defines the comprehensive requirements and specifications for the development of a Credit Card Processing System. It serves as a definitive reference for developers, testers, project managers, and stakeholders throughout the project lifecycle, ensuring all functional and non-functional requirements are clearly specified and implemented.

1.2 Scope

This document covers the complete credit card transaction lifecycle including authorization, fraud detection, billing, reporting, and settlement processes. It defines system architecture, performance benchmarks, security protocols, development timeline, and budget allocation for the entire project implementation.

1.3 Overview

The Credit Card Processing System is a secure, high-availability software solution that enables real-time authorization, processing, and settlement of credit card transactions across multiple payment networks. The system incorporates advanced fraud detection algorithms and ensures compliance with international payment card industry standards.

2. General Description

The Credit Card Processing System is designed to serve multiple stakeholders with specialized functionality:

a) Cardholders can make secure payments, view transaction history, monitor account activity, and receive real-time fraud alerts through multiple communication channels.

b) Merchants can process customer payments, manage transaction batches, generate sales reports, handle refunds and chargebacks, and reconcile daily settlements through integrated POS terminals and online gateways.

c) Acquiring Banks can authorize transactions, manage merchant accounts, process settlements, handle dispute resolution, and generate comprehensive financial reports for business intelligence.

d) Issuing Banks can issue credit cards, manage cardholder accounts, set credit limits, process payments, and implement risk management strategies through configurable business rules.

The system implements end-to-end payment processing while minimizing transaction failures, reducing fraudulent activities, improving operational efficiency, and ensuring regulatory compliance across all payment channels and geographic regions.

3. Functional Requirements

3.1 Transaction Authorization

Validate card details, check available credit limit, verify security codes, and authorize transactions in real-time with multiple fallback mechanisms for system redundancy.

3.2 Fraud Detection

Implement multi-layered fraud prevention using machine learning algorithms, behavioral analysis, velocity checks, and geographic pattern recognition to identify and block suspicious transactions.

3.3 Payment Settlement

Process batch settlements, calculate interchange fees, manage fund transfers between financial institutions, and generate settlement reports with detailed breakdowns.

3.4 Dispute Management

Handle chargeback requests, manage retrieval processes, facilitate arbitration cases, and maintain comprehensive audit trails for all dispute resolution activities.

3.5 Reporting and Analytics

Generate real-time transaction reports, compliance documentation, financial statements, and business intelligence dashboards with customizable parameters and export capabilities.

4. Interface Requirements

4.1 User Interfaces

Web-based admin portal for bank staff, merchant management dashboard, mobile application for cardholders, and kiosk interfaces for branch operations with role-based access control.

4.2 Hardware Interfaces

Integration with POS terminals, ATM networks, card readers, biometric scanners, and hardware security modules for cryptographic key management and secure data storage.

4.3 Software Interfaces

API integration with banking core systems, payment networks (Visa/Mastercard), credit bureaus, fraud detection services, and accounting software packages.

4.4 Communication Interfaces

Secure socket layer encryption for online transactions, ISO-8583 protocol for payment network communications, and SFTP for batch file processing with financial partners.

5. Performance Requirements

5.1 Response Time

Authorization response within 2 seconds for 95% of transactions, with maximum 5-second response time during peak processing periods.

5.2 Scalability

Support minimum 10,000 concurrent transactions with linear scalability to handle 50,000+ transactions during seasonal peaks and promotional events.

5.3 Availability

Ensure 99.99% system uptime with redundant data centers, automatic failover capabilities, and zero-downtime maintenance windows for critical updates.

5.4 Data Integrity

Maintain complete transaction consistency through ACID properties, with comprehensive audit trails and real-time replication across geographically distributed databases.

6. Design Constraints

6.1 Hardware Limitations

Must operate on standard banking-grade servers, support existing POS infrastructure, and maintain compatibility with EMV chip card readers and contactless payment terminals.

6.2 Software Dependencies

Implementation using Java with Spring Boot framework, Oracle database for transaction processing, and integration with existing core banking systems through enterprise service bus.

7. Non-Functional Attributes

7.1 Security

Implement end-to-end encryption, tokenization of sensitive data, PCI-DSS compliance, multi-factor authentication, and regular security penetration testing.

7.2 Reliability

Fault-tolerant architecture with automated disaster recovery, data replication, and business continuity planning to ensure uninterrupted payment processing services.

7.3 Scalability

Modular microservices architecture supporting horizontal scaling, load balancing, and elastic resource allocation to handle variable transaction volumes.

7.4 Maintainability

Comprehensive logging, monitoring capabilities, automated deployment pipelines, and modular design facilitating easy updates and feature enhancements.

7.5 Compliance

Adherence to PCI-DSS standards, local financial regulations, data protection laws, and international anti-money laundering requirements across all operational regions.

8. Preliminary Schedule and Budget

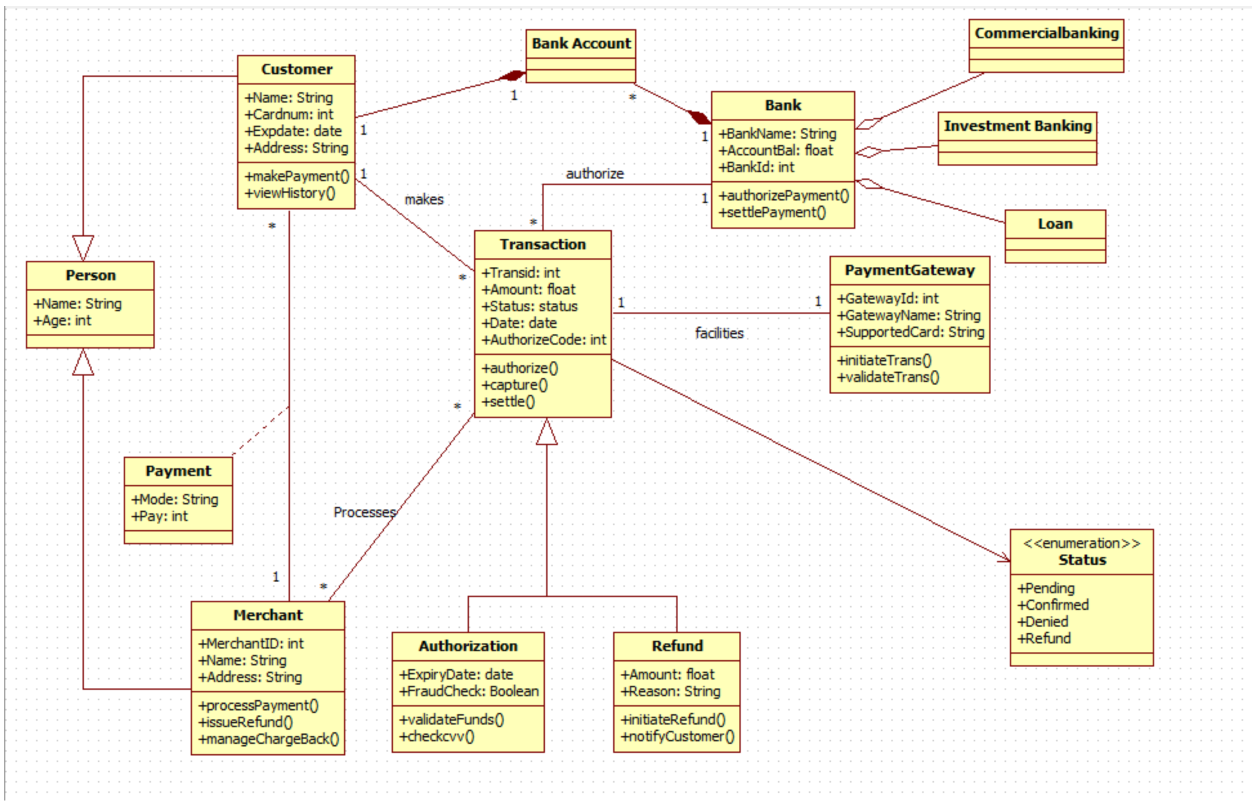
8.1 Development Timeline: 9 months

1. Requirements Analysis & Design: 2 months
2. Core Development: 4 months
3. Security Testing & Compliance: 2 months
4. Deployment & Transition: 1 month

8.2 Budget Estimate: $500,000 USD covering licensing, development, security certification, infrastructure, and implementation services.

Class Diagram

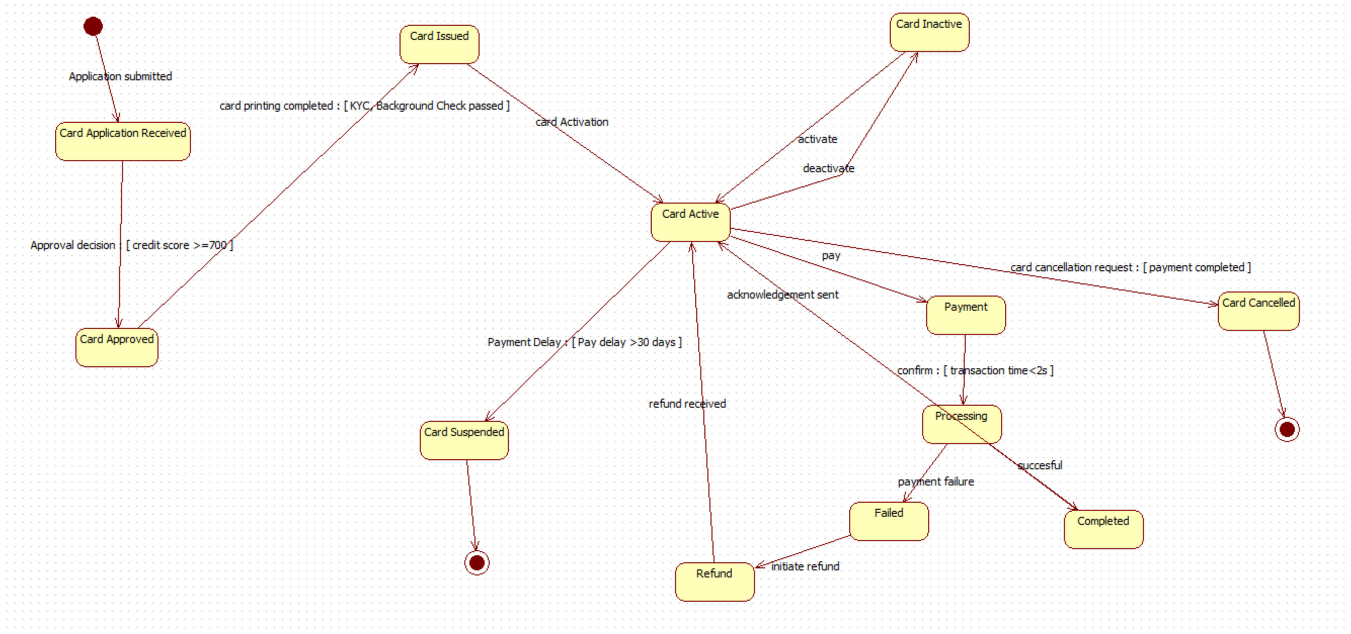
Models the core entities in credit card processing including Customer, Transaction, Merchant, PaymentGateway, and Bank. Shows inheritance relationships with CommercialBanking and Investment Banking specializing from Business class. Includes enumeration for Transaction Status and association classes for authentication. Demonstrates how transactions flow between customers, merchants, and banks through payment gateways with security validations.



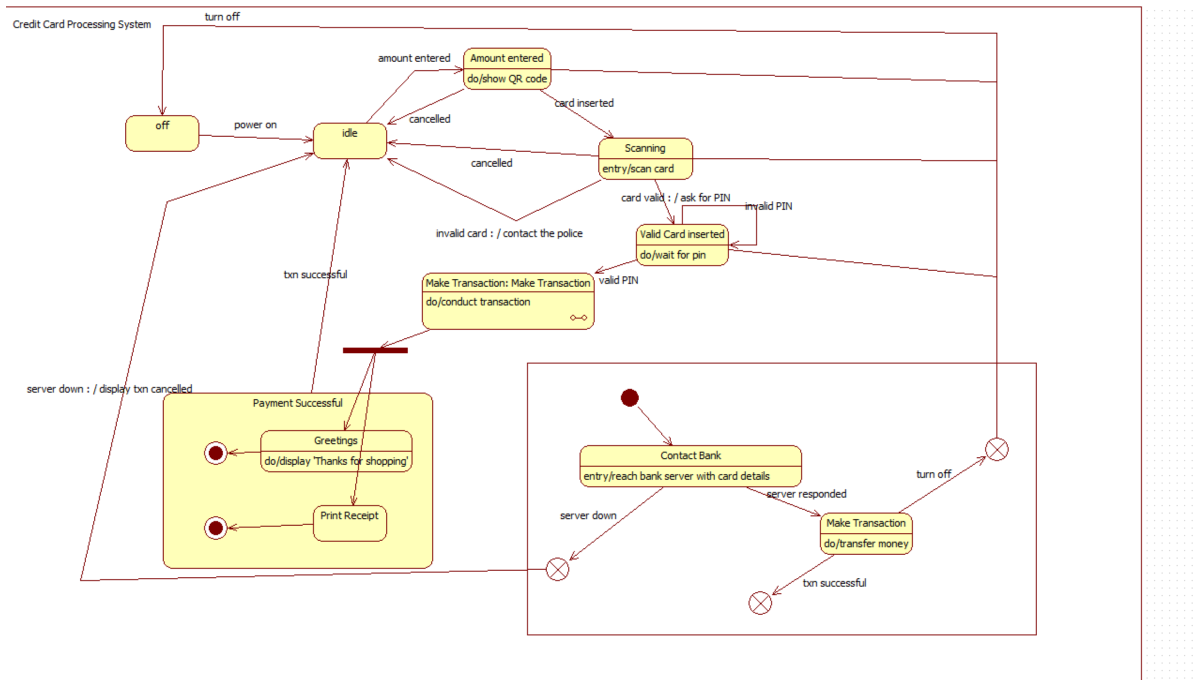
State Diagram

Tracks transaction lifecycle from initiation through authorization, processing, settlement, to completion. Includes states for fraud checking, fund verification, and exception handling. Shows transitions with guard conditions like [sufficient funds] and [fraud detected]. Manages failed transaction rollbacks and refund processing states with concurrent verification processes.

Simple State Diagram


Application submitted
Card Application Received
Approval decision : [credit score >=700]
Card Approved
card printing completed : [KYC, Background Check passed]
Card Issued
card Activation
Card Inactive
activate
deactivate
Card Active
pay
card cancellation request : [payment completed]
Card Cancelled
acknowledgement sent
Payment
Payment Delay : [Pay delay >30 days]
Card Suspended
refund received
confirm : [transaction time<2s]
Processing
succesful
payment failure
Failed
Completed
initiate refund
Refund

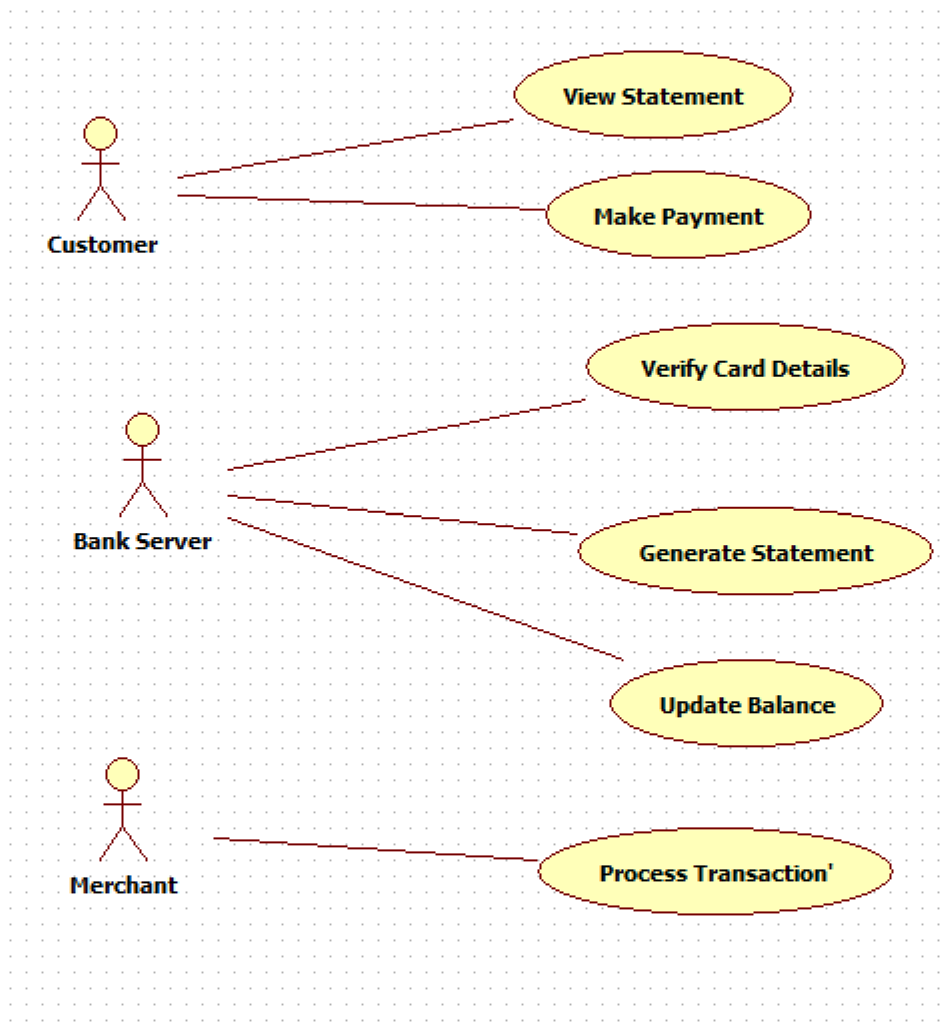
Advance State Diagram



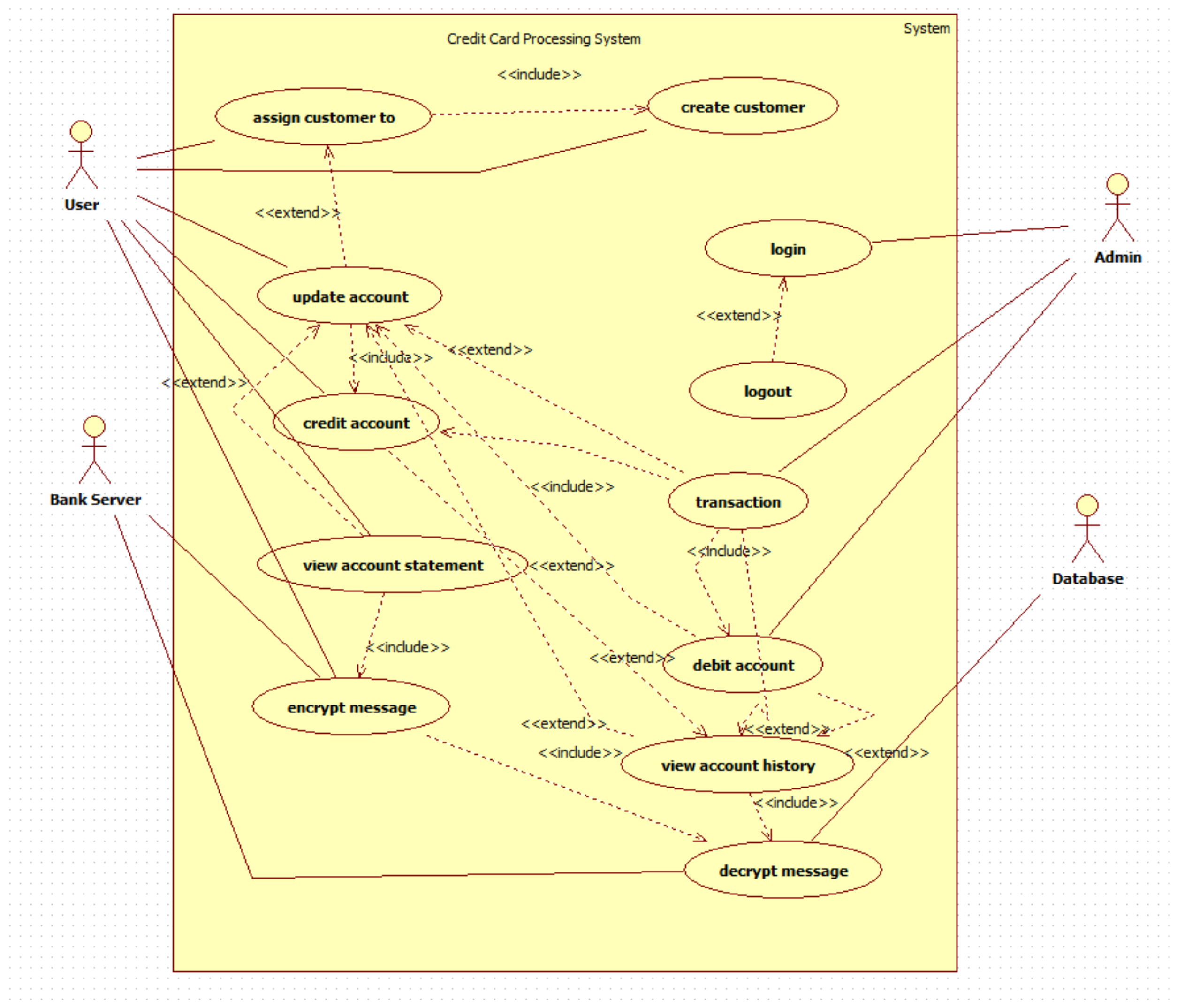
Use Case Diagram

Illustrates interactions between Cardholder, Merchant, Bank, and Administrator. Core use cases include Process Payment, View Statement, and Handle Chargeback. Extend relationships manage fraud detection and transaction reversal scenarios. Include relationships connect payment processing with OTP verification and receipt generation for complete transaction handling.

Simple Use Case Diagram



Advance Use Case Diagram



Sequence Diagram

Chronologically details payment processing from card swipe through authorization to confirmation. Shows message exchanges between Customer, POS Terminal, Payment Gateway, and Bank systems. Includes synchronous calls for real-time authorization and asynchronous messages for settlement processing. Demonstrates error handling for declined transactions and retry mechanisms.

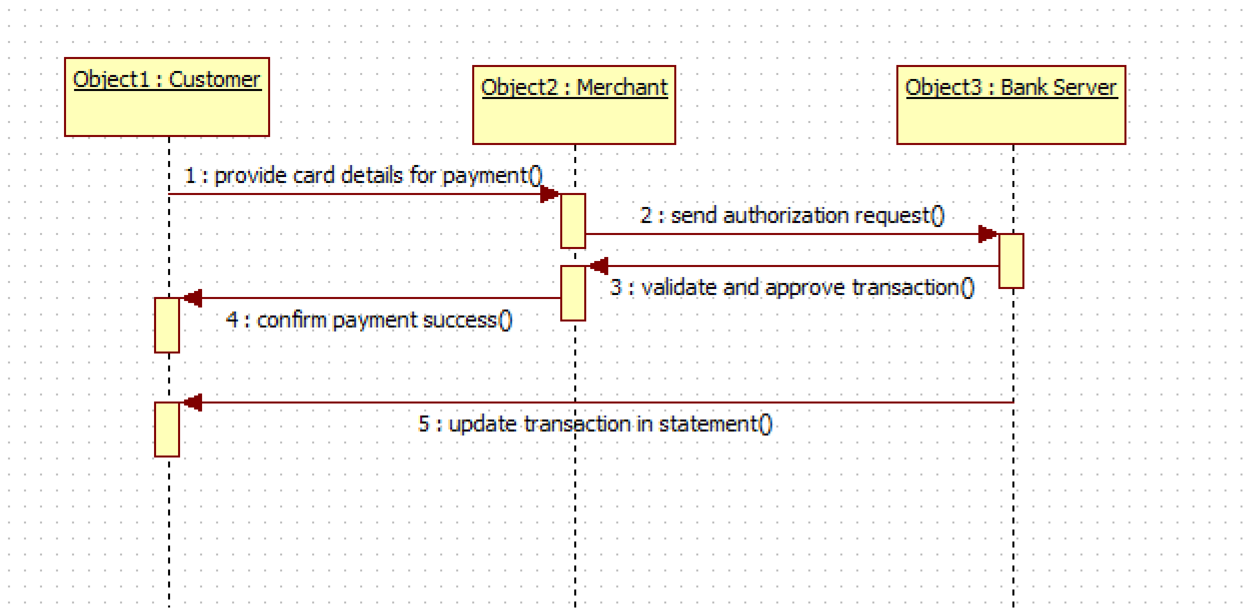
Scenario 1: Successful Credit Card Payment

1. Pre-condition: Card is valid and has sufficient credit balance.
2. Main Flow: User enters card details → system validates card → system processes payment → bank authorizes transaction → system shows “Payment Successful.”
3. Post-condition: Transaction recorded, credit balance reduced, confirmation sent.

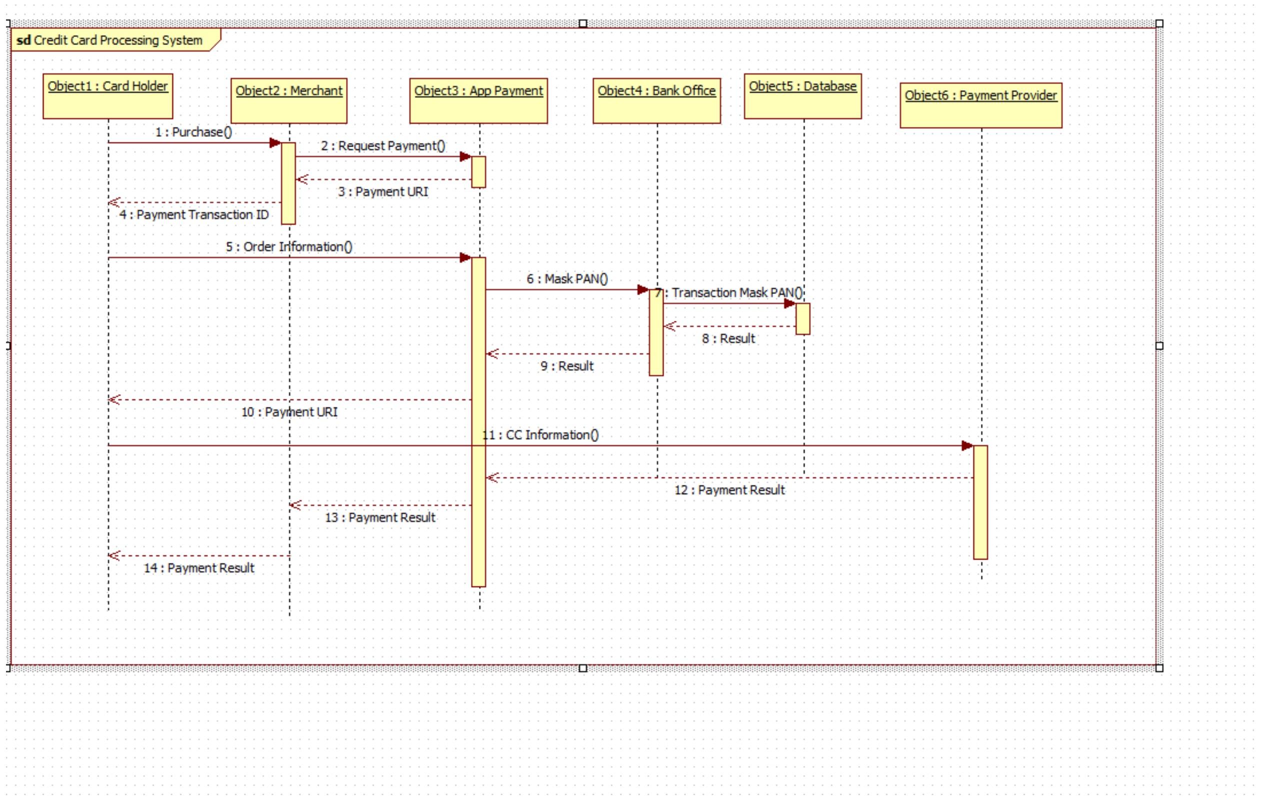
Scenario 2: Credit Card Bill Payment

1. Pre-condition: User has an outstanding bill and is logged in.
2. Main Flow: User selects “Pay Bill” → enters amount → system processes payment → bank confirms transfer → system updates outstanding balance.
3. Post-condition: Bill amount reduced, payment record stored.

Simple Sequence Diagram

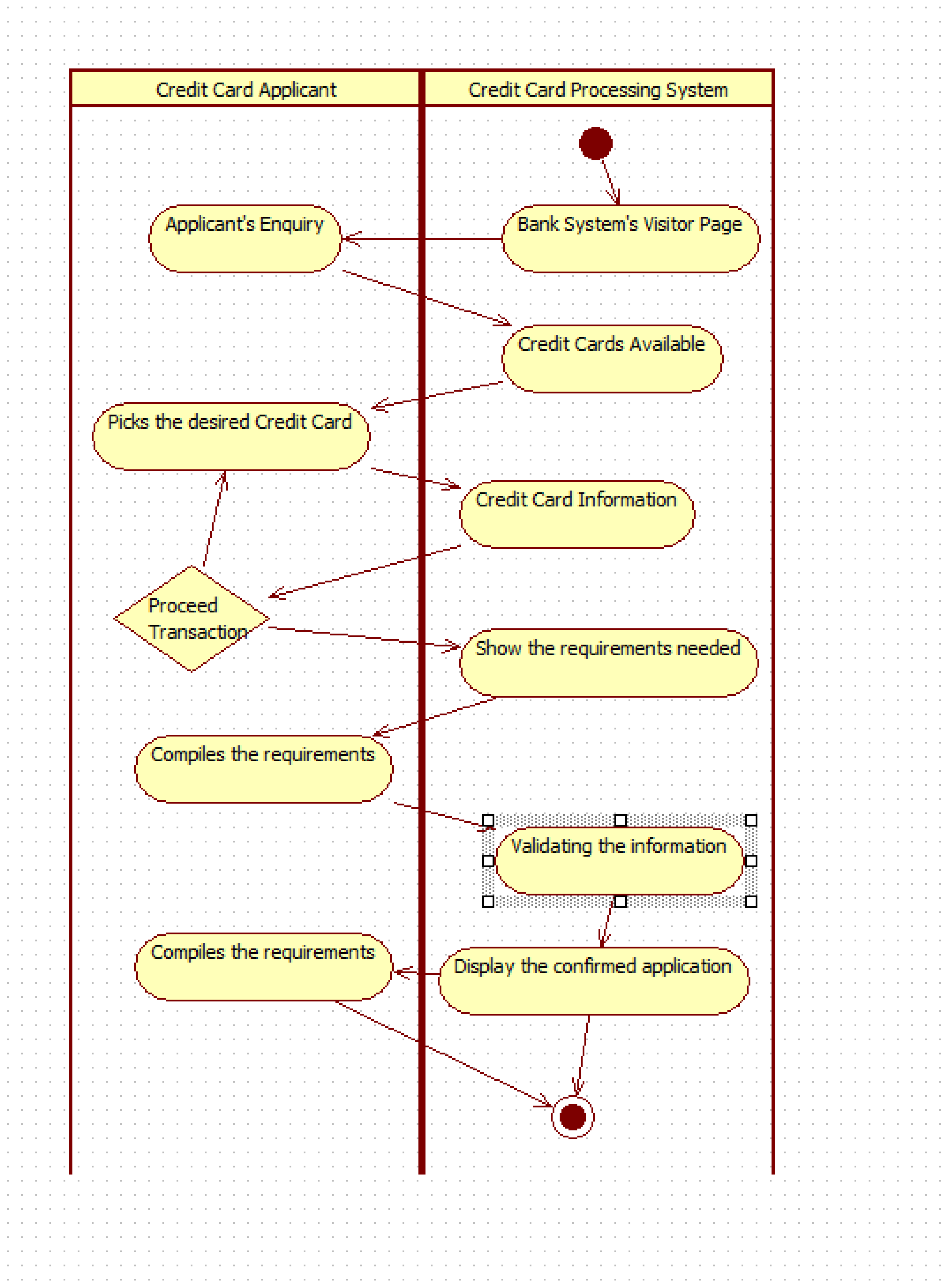

Object1 : Customer
Object2 : Merchant
Object3 : Bank Server
1 : provide card details for payment()
2 : send authorization request()
3 : validate and approve transaction()
4 : confirm payment success()
5 : update transaction in statement()

Advance Sequence Diagram



Activity Diagram

Maps complete payment workflow from transaction initiation through verification, processing, to final settlement. Swimlanes organize activities by Customer, Merchant, Payment Gateway, and Bank responsibilities. Shows parallel processing for fraud checking and fund verification with decision nodes for approval/decline paths. Includes exception handling for network failures and insufficient funds.


Credit Card Applicant
Credit Card Processing System
Applicant's Enquiry
Bank System's Visitor Page
Credit Cards Available
Picks the desired Credit Card
Credit Card Information
Proceed Transaction
Show the requirements needed
Compiles the requirements
Validating the information
Compiles the requirements
Display the confirmed application

3. Library Management System

Problem Statement

Managing book inventories, member records, and transactions manually is inefficient and error-prone. An automated Library Management System is needed to streamline cataloging, circulation processes, member services, and overall library operations.

Software Requirements Specification

1. Introduction

1.1 Purpose

This document comprehensively defines the requirements and specifications for the Library Management System. It will outline all project objectives, scope boundaries, functional requirements, and deliverables for all stakeholders.

1.2 Scope

This document defines the complete working methodology of the library management system, including all functional modules, technical specifications, project timeline, resource allocation, and cost estimation.

1.3 Overview

The Library Management System is a comprehensive software solution designed to handle cataloging processes, book check-ins and returns, member management, fine calculations, reporting, and typesetting for efficient library operations and enhanced user experience.

2. General description

The Library Management System is designed to serve multiple user groups with specialized functionality:

a) Librarians can manage complete book inventories, process acquisitions, handle cataloging, manage member registrations, and generate operational reports for library administration.

b) Library Members can search the catalog, check book availability, reserve items, renew loans online, view borrowing history, and pay fines electronically through self-service portals.

c) Administrative Staff can manage membership databases, generate statistical reports, analyze collection usage patterns, track budget expenditures, and monitor library performance metrics.

d) Library Assistants can process daily transactions, handle book returns, manage shelving operations, assist with inventory checks, and provide frontline support to library users.

The system addresses current limitations in manual library processes while providing librarians, members, and administrative staff with automated tools that minimize errors, improve operational efficiency, enhance user satisfaction, and optimize resource utilization across all library functions.

3. Functional requirements

3.1 Book Catalog Management

The system shall maintain a comprehensive and searchable database of all library materials including books, journals, multimedia, and digital resources. Track real-time book availability, physical location, and circulation status with detailed metadata management.

3.2 Member Management

Register new members using complete personal and demographic data. Maintain member profiles, track borrowing history, manage membership tiers, and handle membership renewals and cancellations systematically.

3.3 Circulation Management

The system shall efficiently manage the entire process of issuing and returning books, including loan period management, renewal processing, reservation handling, and automatic imposition of fines for late returns with multiple notification escalations.

3.4 Reporting and Analytics

The system shall generate comprehensive reports on book usage statistics, member activity patterns, circulation trends, inventory status, and provide detailed collection statistics for administrative decision-making and strategic planning.

4. Interface Requirements

4.1 User Interface

Simple, intuitive, and user-friendly web portal for librarians and members with responsive design. Ensure seamless web and desktop accessibility with accessibility compliance for users with disabilities.

4.2 Integration Interfaces

Comprehensive integration with barcode scanners and RFID systems for efficient book identification and circulation management. Support for online catalog services, digital resource platforms, and academic search engines for enhanced discovery services.

5. Performance Requirements

5.1 Response Time

Provide rapid search functionality and display transaction processing responses within 3 seconds under normal operational load to ensure smooth user experience.

5.2 Scalability

Robust architecture to support at least 2000 active members simultaneously while maintaining system performance during peak usage periods such as semester beginnings and examination seasons.

5.3 Data Integrity

Ensure complete accuracy and consistency in book records, member information, and transaction data through validation rules, referential integrity constraints, and comprehensive audit trails.

6. Design Constraints

6.1 Hardware Constraints

Compatibility with standard library computers, barcode scanners, RFID readers, and self-service kiosks commonly deployed in modern library environments.

6.2 Software Dependencies

The system should utilize enterprise-grade relational database management systems. Backend implementation to be developed using Java with Spring Boot framework ensuring robustness, security, and maintainability.

7. Non-Functional Attributes

7.1 Security

Protect sensitive member data and library records with robust authentication mechanisms, role-based access control, and data encryption for privacy compliance.

7.2 Reliability

Ensure high system availability during peak library hours with minimal downtime, automated backup systems, and disaster recovery capabilities for continuous service delivery.

7.3 Scalability

The system should be architecturally designed to easily expand and support larger book collections, additional branches, and increased user capacity without significant reengineering.

7.4 Accessibility

The software must be readily accessible via multiple platforms including desktop computers, self-service kiosks, tablets, and mobile applications with consistent functionality.

7.5 Usability

Provide easy navigation and intuitive interface design for users with limited technical skills, minimizing training requirements and supporting quick adoption across all user groups.

7.6 Maintainability

The system should ensure modular architecture with well-defined interfaces during its future upgrades, allowing easy feature additions and minimal disruption during maintenance.

7.7 Compatibility

The system shall maintain full compatibility with common web browsers, operating systems, and mobile platforms while adhering to web standards and accessibility guidelines.

7.8 Data Integrity

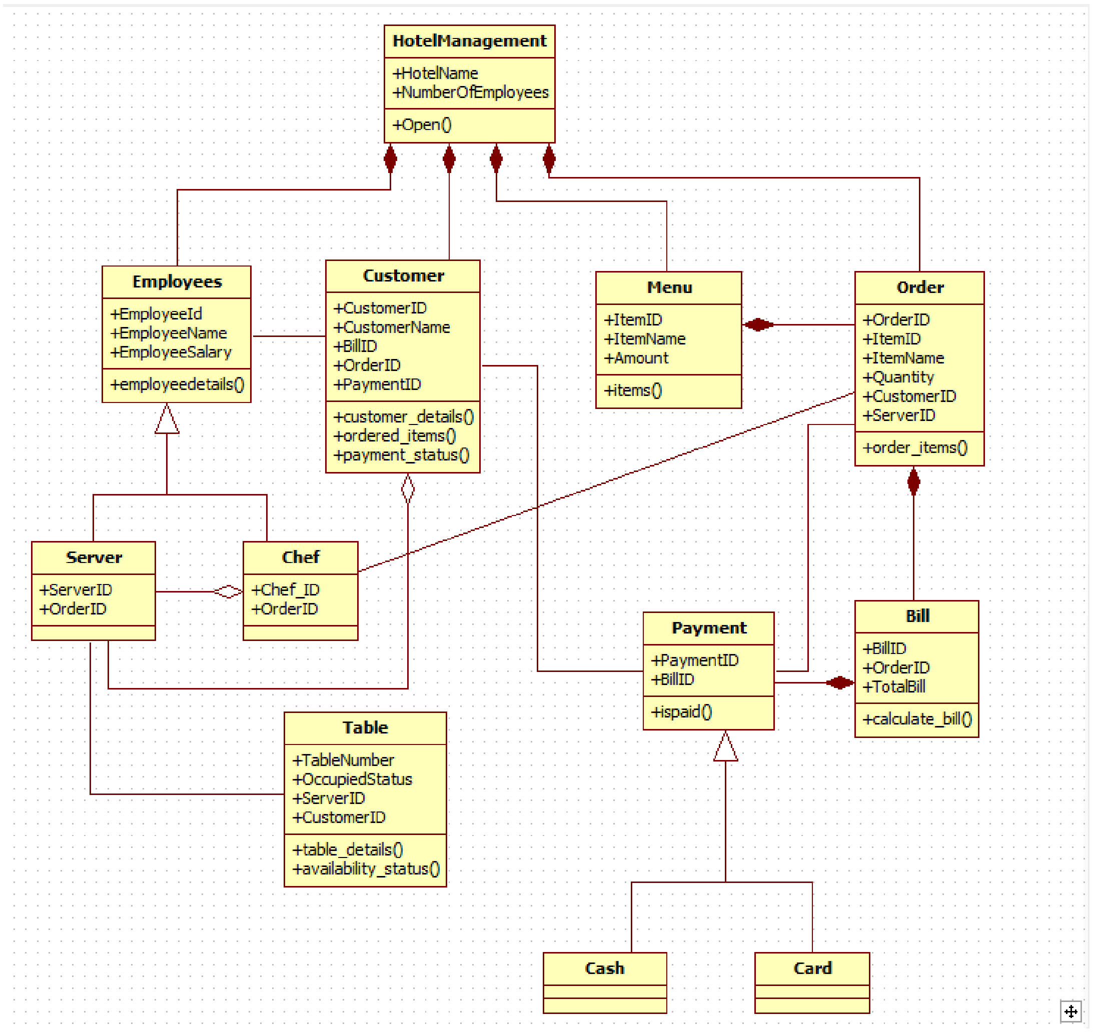
The system shall ensure consistent and accurate management of book inventory records, member information, circulation data, and all transactional records through comprehensive validation and reconciliation processes.

8. Preliminary schedule and budget

The Library Management System is estimated to require 4 months for complete implementation with an allocated budget of $20,000 including all phases of development, comprehensive testing, staff training, and production deployment. The project timeline includes 3 weeks for requirements analysis, 10 weeks for development, 3 weeks for testing, and 2 weeks for deployment and user training.

Class Diagram

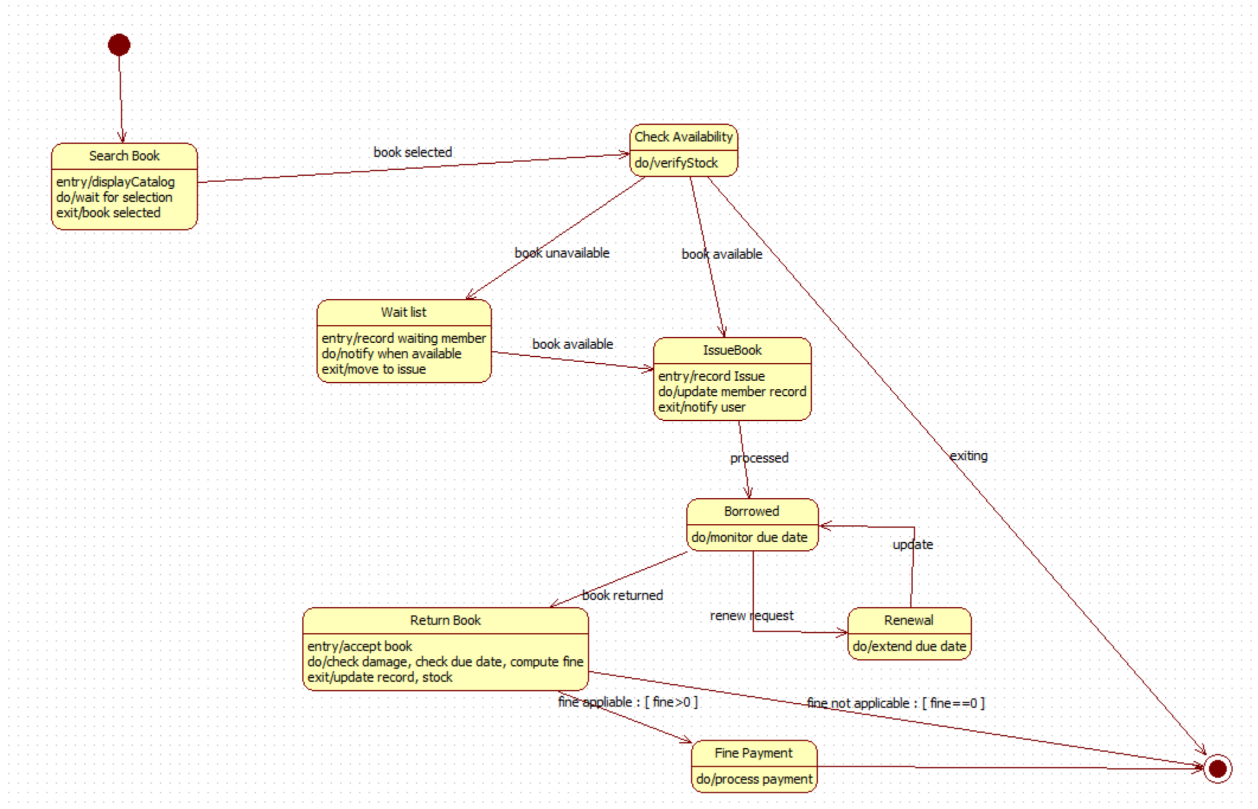
Models library entities including Book, Member, Librarian, Loan, and Fine with their attributes and operations. Relationships include composition between Library and Books, associations between Members and Loans, and inheritance for different Member types. Shows how fines are calculated and linked to overdue loans, demonstrating the complete library ecosystem structure and interactions.



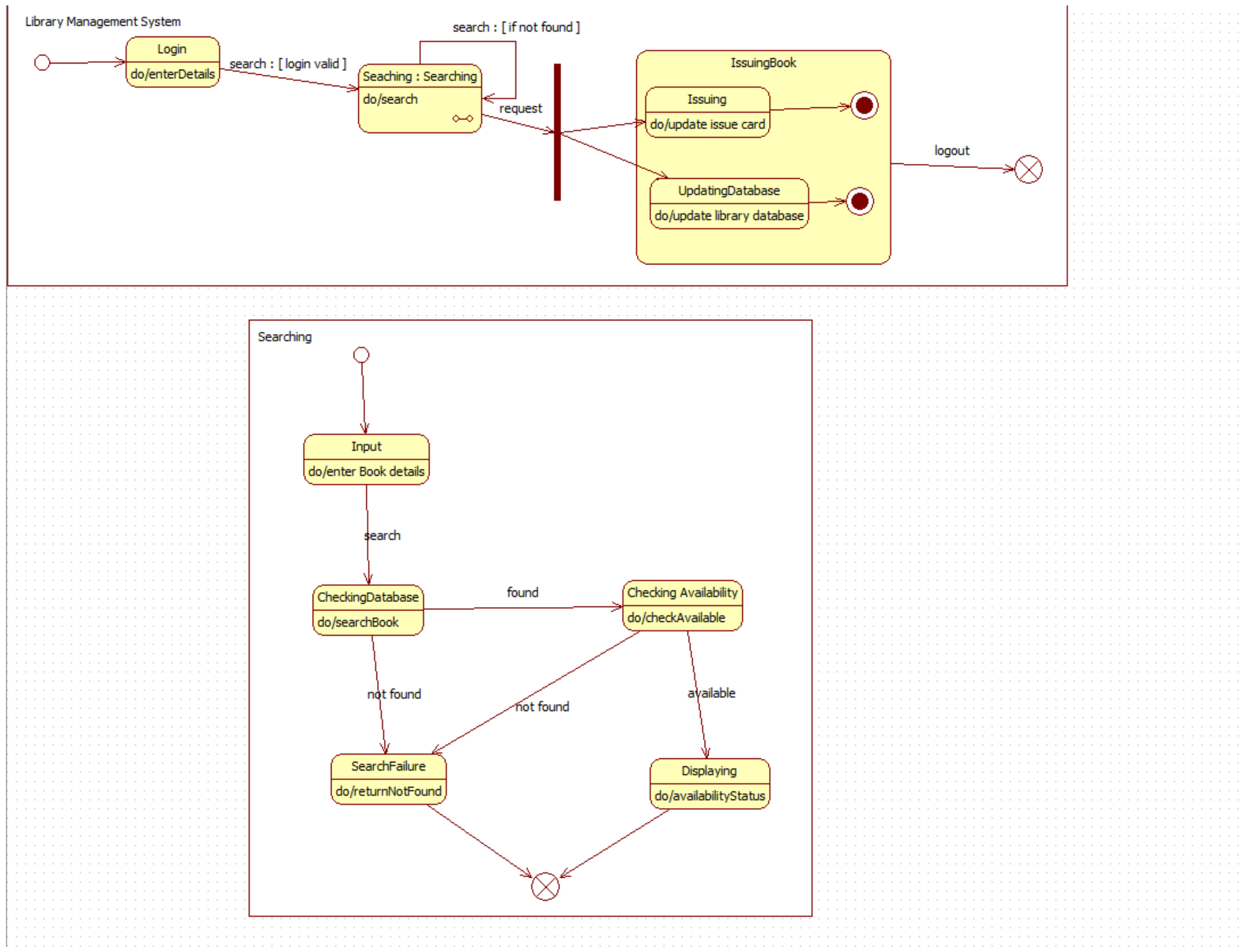
State Diagram

Tracks book lifecycle from available to reserved, issued, returned, or overdue states. Includes transitions for renewal processes, reservation expirations, and fine impositions. Shows concurrent states for member validation and availability checking during book issuance. Manages the complete circulation workflow with conditional transitions for different member types and book categories.

Simple State Diagram



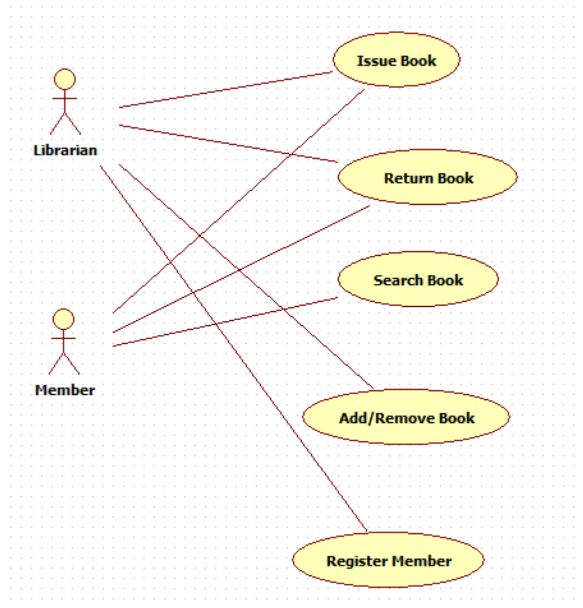
Advance State Diagram



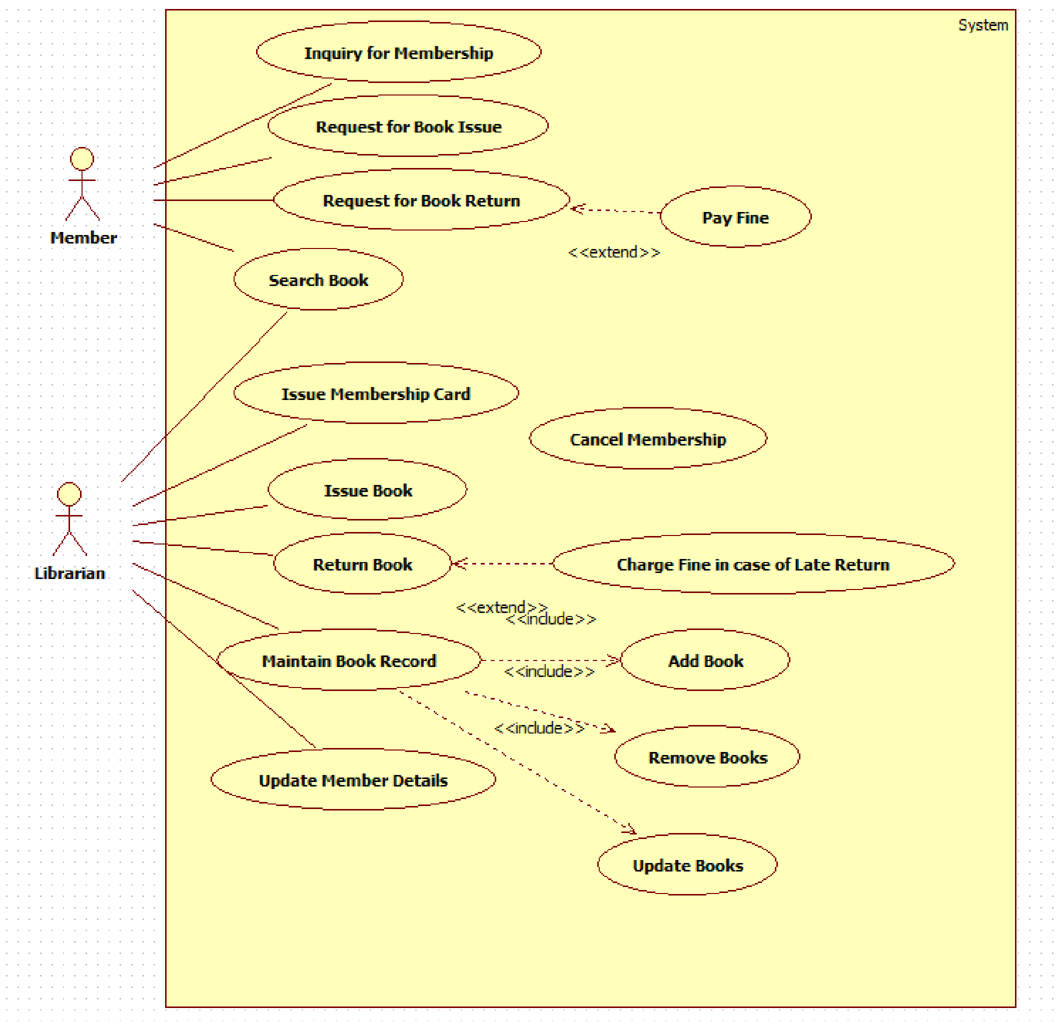
Use Case Diagram

Illustrates system interactions for Librarians managing catalog and members, Members searching and borrowing books, and Administrators generating reports. Extend relationships handle fine calculations for overdue returns and reservation notifications. Include relationships connect book search with availability checking and member validation during borrowing processes.

Simple Use Case Diagram



Advance Use Case Diagram



Sequence Diagram

Details chronological interactions during book borrowing between Member, Librarian, System, and Database objects. Shows message sequences for membership validation, availability checks, loan recording, and due date calculation. Includes return processing with fine calculations and reservation handling for next waiting members. Demonstrates error handling for invalid members or unavailable books.

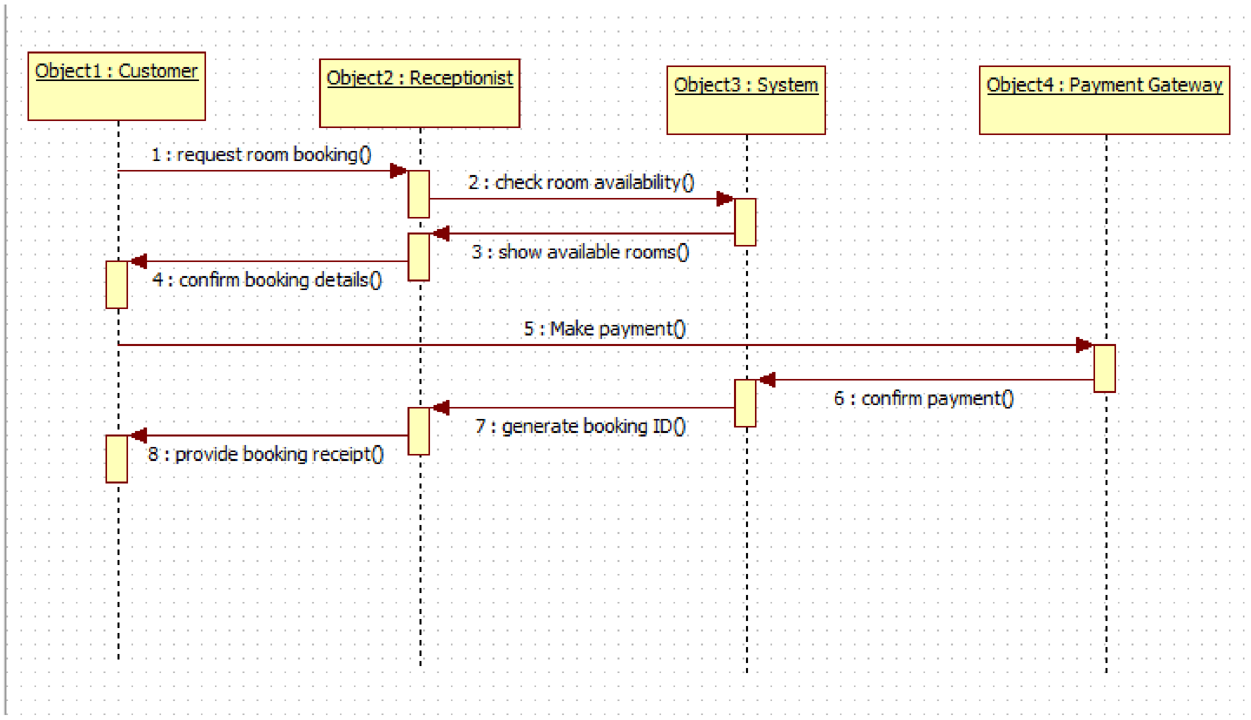
Scenario 1: Issuing a Book

1. Pre-condition: User is registered, book is available.
2. Main Flow: User searches for a book → selects "Issue Book" → system verifies availability → book is issued.
3. Post-condition: Book status updated to issued, user borrowing record updated.

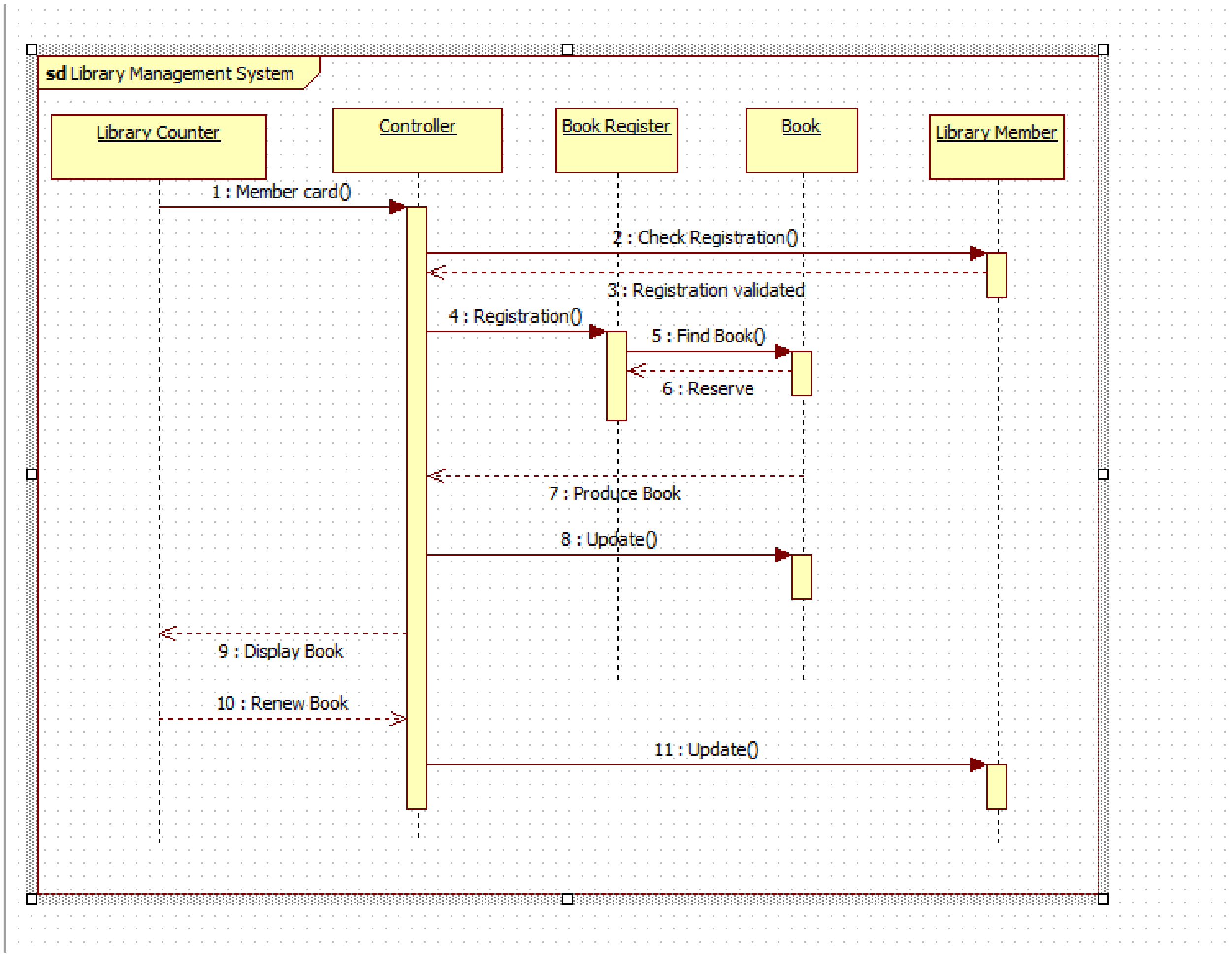
Scenario 2: Returning a Book

1. Pre-condition: User has an issued book.\
2. Main Flow: User selects "Return Book" → system checks book condition and due date → book is marked returned.
3. Post-condition: Inventory updated to available, user loan record updated.

Simple Sequence Diagram

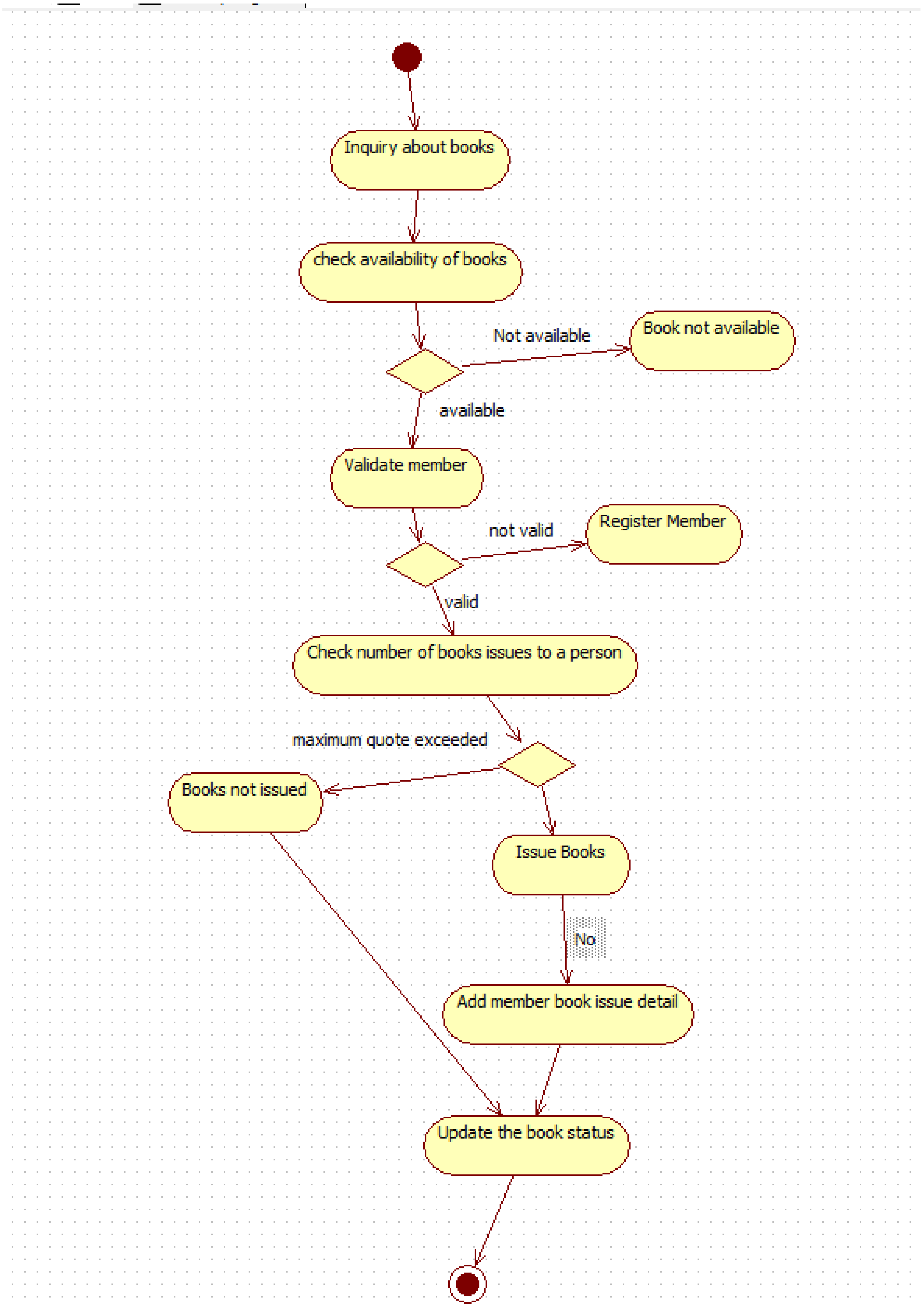

Object1 : Customer
Object2 : Receptionist
Object3 : System
Object4 : Payment Gateway
1 : request room booking()
2 : check room availability()
3 : show available rooms()
4 : confirm booking details()
5 : Make payment()
6 : confirm payment()
7 : generate booking ID()
8 : provide booking receipt()

Advance Sequence Diagram



Activity Diagram

Maps complete book borrowing workflow from search initiation through availability checking, member validation, issuance, to return processing. Swimlanes distribute activities among Member, Librarian, and System responsibilities. Shows parallel processing for multiple book transactions and decision nodes for membership status and book availability. Includes fine calculation and payment activities for overdue returns.


Inquiry about books
check availability of books
Not available
Book not available
available
Validate member
not valid
Register Member
valid
Check number of books issues to a person
maximum quote exceeded
Books not issued
Issue Books
No
Add member book issue detail
Update the book status

4. Stock Maintenance System

Problem Statement

Manually tracking stock levels, supplier information, and purchase orders leads to data inconsistencies and operational delays. A digital Stock Maintenance System is required to ensure real-time inventory management, accurate forecasting, and efficient supply chain operations.

Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this document is to describe the comprehensive requirements and specifications for the Stock Maintenance System. It aims to clarify all project objectives, scope, deliverables, and technical specifications for developers, testers, and stakeholders.

1.2 Scope

This document defines the complete functionality, implementation timeline, budget allocation, and testing requirements of the Stock Maintenance System. It covers all aspects of inventory management from procurement to sales.

1.3 Overview

The Stock Maintenance System is an enterprise-level solution for managing inventory levels, purchase orders, supplier relationships, client information, and stock movements in real-time across multiple locations and warehouses.

2. General description

The Stock Maintenance System is designed to serve various business stakeholders with comprehensive inventory control capabilities:

a) **Inventory Managers** can monitor stock levels in real-time, set reorder points, track inventory across multiple warehouses, and generate comprehensive stock reports.

b) **Procurement Staff** can manage supplier relationships, create and track purchase orders, monitor delivery schedules, and process supplier payments efficiently.

c) **Sales Teams** can check product availability, reserve inventory for customers, track stock movements, and generate sales-related inventory reports.

d) **Warehouse Operators** can perform stock counts, manage item locations, process receipts and dispatches, and handle stock transfers between locations.

The system implements automated inventory tracking while minimizing manual errors, improving stock accuracy, optimizing inventory turnover, and providing real-time visibility into stock positions across the entire organization.

3. Functional Requirements

3.1 Inventory Management

Maintain complete details of all stock items including SKU, description, category, pricing, and supplier information. Automatically update stock levels after purchases, sales, and adjustments. Support multiple warehouse locations and bin management.

3.2 Supplier Management

Comprehensively track supplier information including contact details, performance metrics, and contract terms. Monitor purchase history, track delivery performance, manage outstanding payments, and maintain supplier quality records.

3.3 Purchase Order Management

Automatically generate purchase orders for low-stock items based on predefined reorder points. Track purchase order status from creation to delivery. Manage partial deliveries, returns, and quality inspections. Handle purchase order amendments and cancellations.

3.4 Reporting and Analytics

Generate comprehensive stock reports including inventory valuation, stock turnover ratios, and aging analysis. Provide operational metrics, supplier performance reports, and demand forecasting analytics. Support custom report generation and data export capabilities.

4. Interface Requirements

4.1 User Interface

Unified web-based dashboard for staff and administrative accounts with role-based access control. Support for both web and desktop access with responsive design for mobile devices.

4.2 Integration Interfaces

Seamless integration with existing ERP systems, accounting software, and e-commerce platforms. Comprehensive support for barcode/RFID-based tracking systems. API interfaces for third-party system integration and data exchange.

5. Performance Requirements

5.1 Response time

The system shall provide real-time stock updates and inventory queries within 2 seconds under normal load conditions.

5.2 Availability

The system should efficiently handle up to 50,000 active stock items simultaneously while supporting multiple concurrent users across different locations.

5.3 Data Integrity

Ensure complete consistency and accuracy of stock data across all transactions, locations, and reporting periods. Maintain audit trails for all inventory movements and adjustments.

6. Design Constraints

6.1 Hardware limitations

The system must operate on standard warehouse computers, support various barcode scanners, and be compatible with radio-frequency identification (RFID) systems for automated inventory tracking.

6.2 Software Dependencies

The system shall use enterprise-grade relational databases such as MySQL or PostgreSQL. Implementation to be done using Python with Spring/Sponge framework for backend services and modern web technologies for frontend interfaces.

7. Non-Functional Attributes

7.1 Security

The system shall implement robust security measures to protect supplier data, pricing information, and all transaction records. Role-based access control and data encryption for sensitive information.

7.2 Reliability

The system shall have a fault-tolerant design with automated backup systems, disaster recovery procedures, and high availability configuration for critical inventory operations.

7.3 Scalability

The system architecture should be easily expandable to include more warehouses, support additional users, and handle increased transaction volumes without performance degradation.

7.4 Portability

The system must be accessible on multiple platforms including desktop computers, tablets, and mobile devices with consistent functionality and user experience.

7.5 Usability

The system shall feature an intuitive and user-friendly interface designed specifically for warehouse staff with varying technical expertise, minimizing training requirements.

7.6 Maintainability

The system shall employ a modular architecture that facilitates easy updates, feature additions, and integration of new analytical capabilities with minimal disruption to operations.

7.7 Compatibility

The system should maintain compatibility with multiple devices, browsers, and operating systems while ensuring consistent performance and functionality across all platforms.

7.8 Data integrity

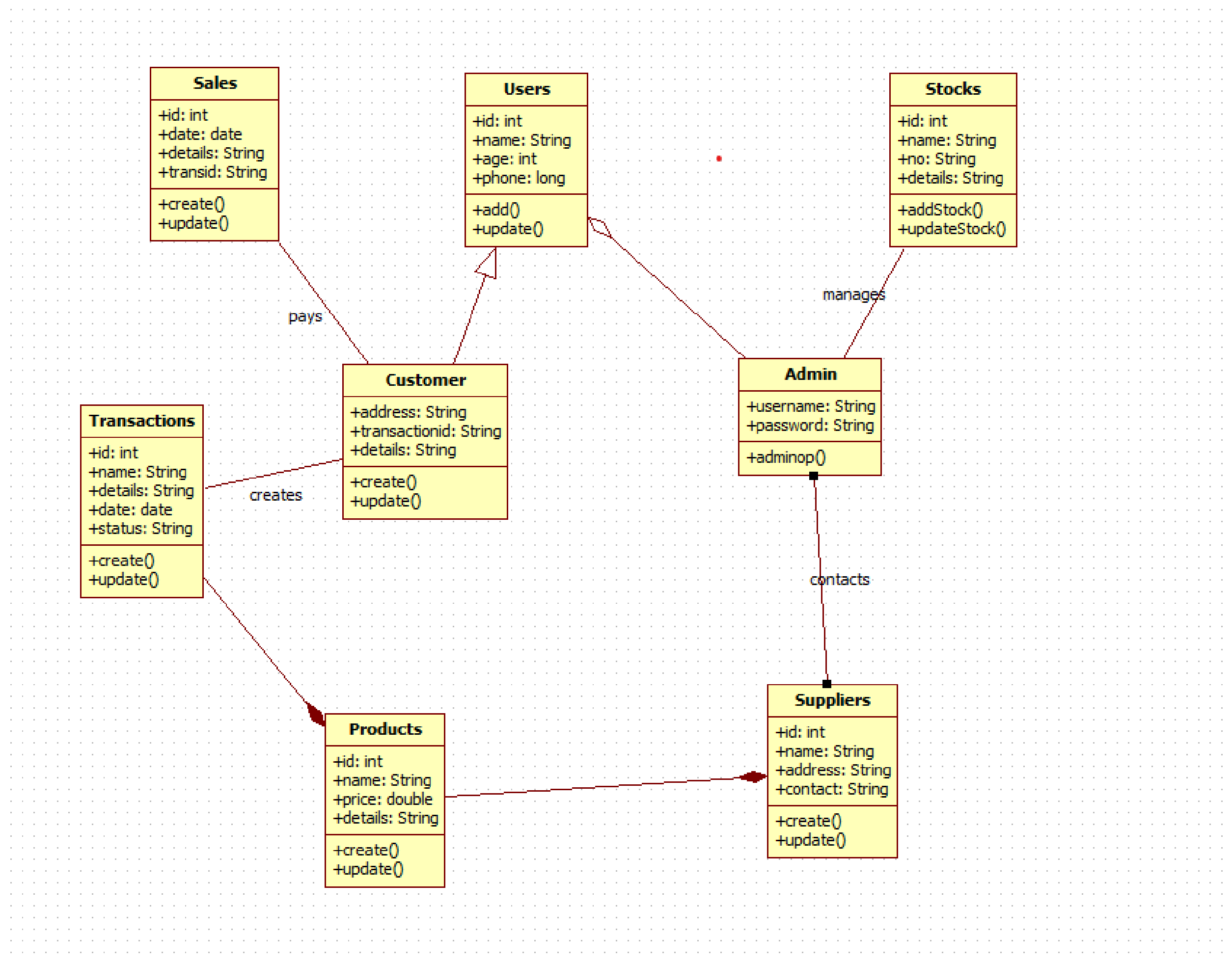
The system must ensure accurate and consistent storage records through validation rules, automated reconciliation processes, and comprehensive audit trails for all inventory transactions.

8. Preliminary schedule and budget

The Stock Maintenance System is estimated to require 5 months for complete implementation with a comprehensive budget of $80,000 including planning, development, testing, training, and deployment phases. The project timeline includes 1 month for requirements analysis, 2 months for development, 1 month for testing, and 1 month for deployment and user training.

Class Diagram

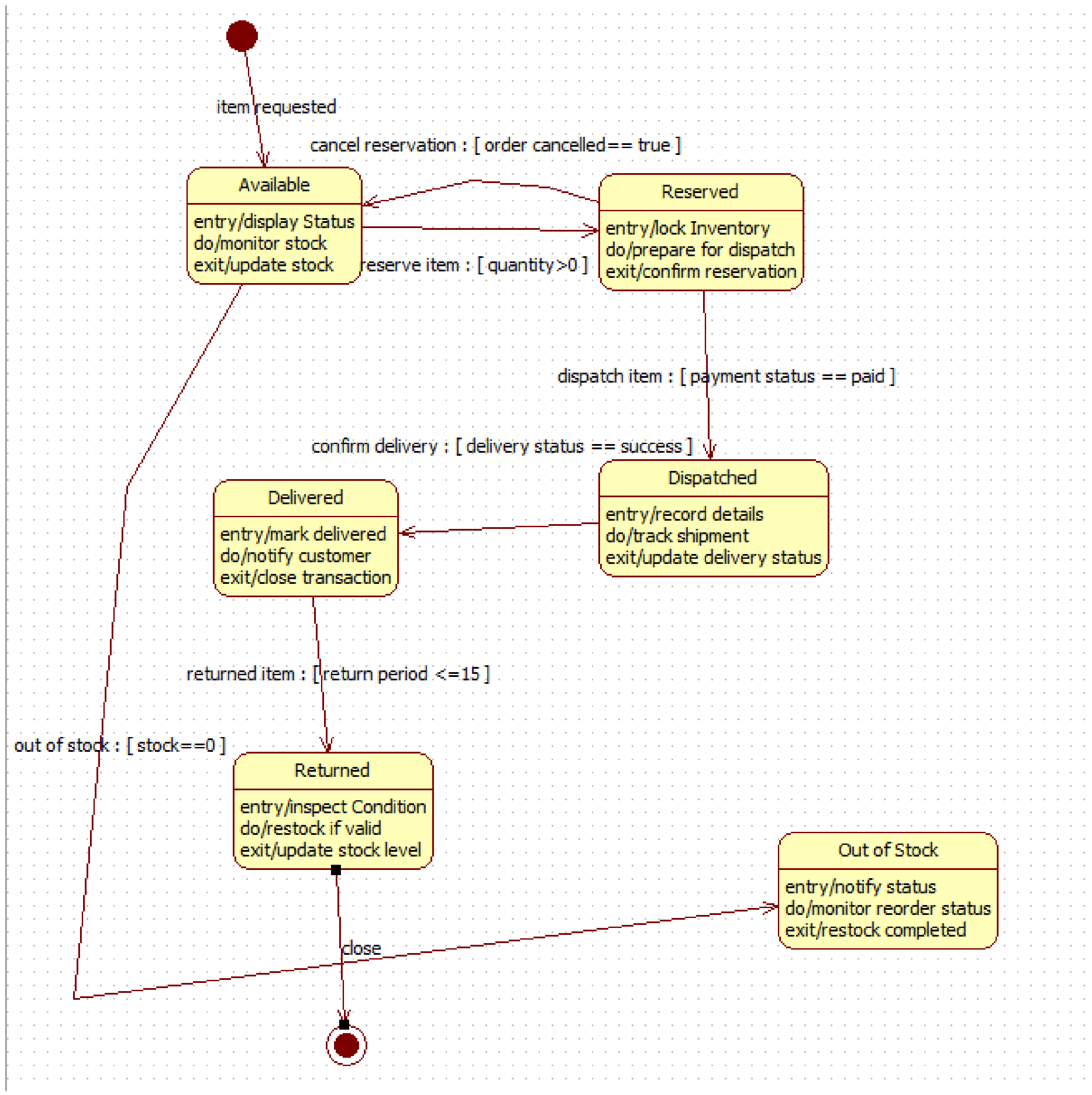
Models inventory management entities including Stock, Supplier, PurchaseOrder, and Transaction with their attributes and operations. Relationships include aggregation between Supplier and Stock, composition linking Inventory to Products, and associations connecting Orders to Suppliers. Demonstrates how stock levels are updated through purchase orders and sales transactions, showing the complete supply chain management structure.



State Diagram

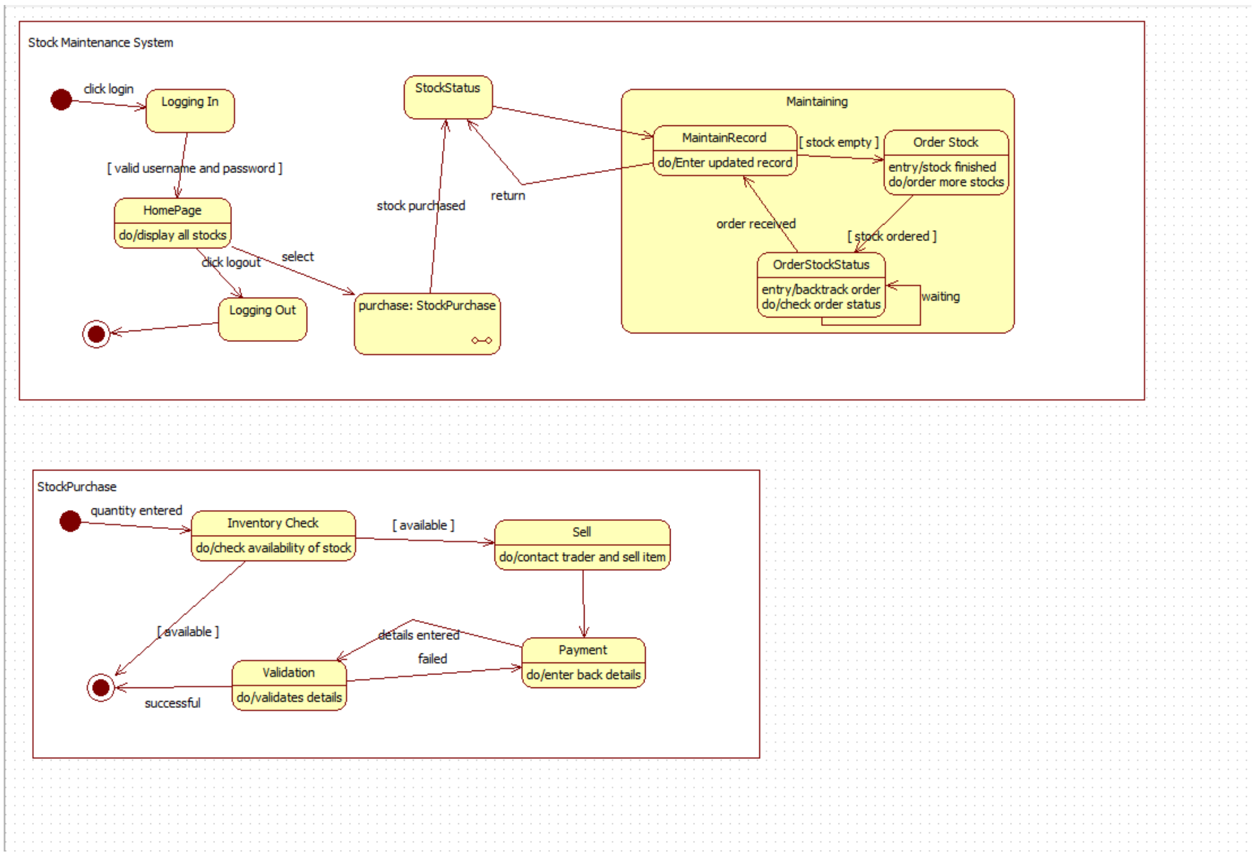
Tracks inventory item lifecycle from available to reserved, out-of-stock, reordered, and delivered states. Includes transitions for stock depletion, reorder triggers, and delivery processing. Shows concurrent states for quality checks and supplier validation during restocking. Manages automated reorder processes and backorder handling with conditional transitions based on stock levels and demand patterns.

Simple State Diagram



Advance State Diagram

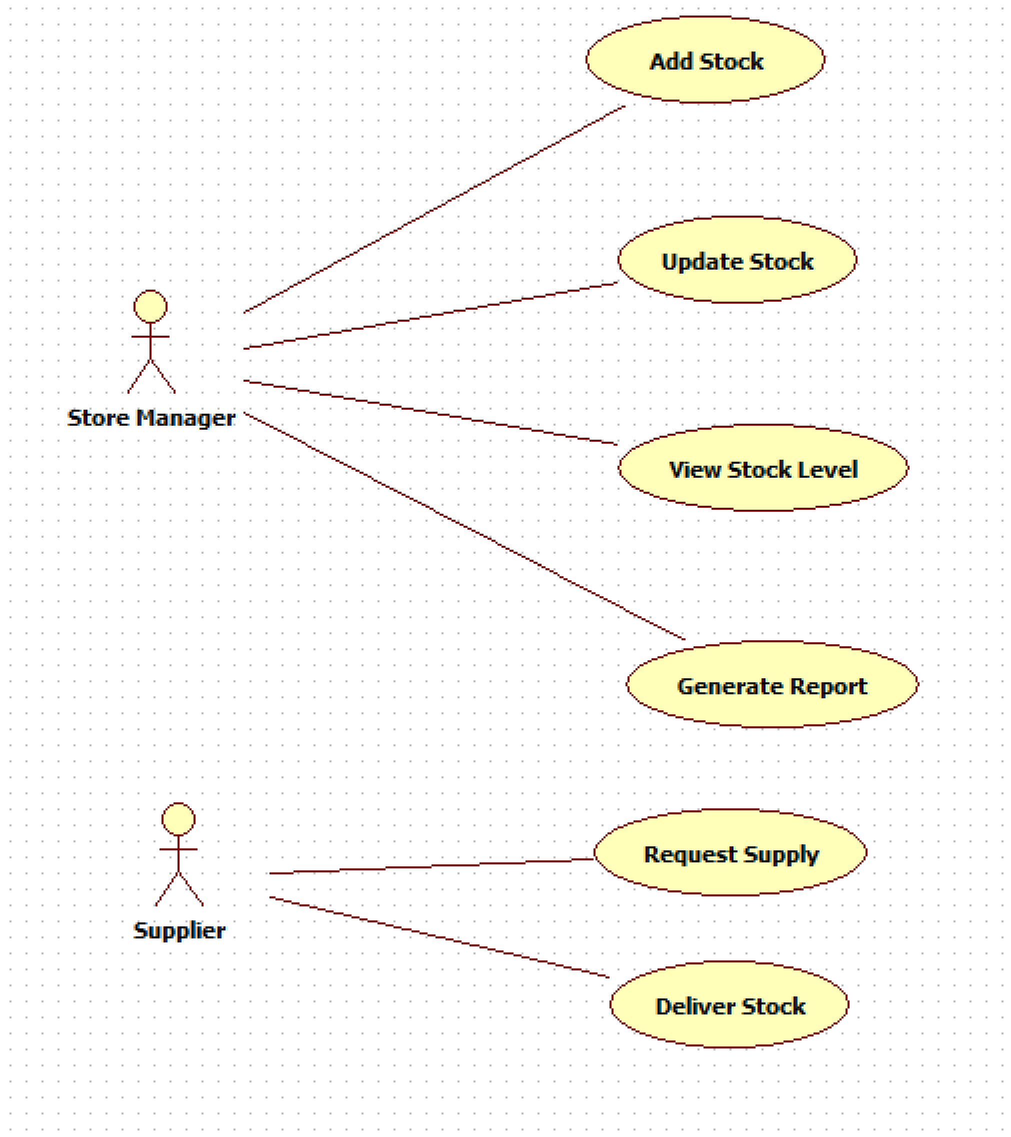
State Diagram: one simple state diagram, one advance state diagram (either Sub Machine or Nested State, include any one of the concurrency in your design)
Requirements: minimum of 6 states for both the state diagram with state name, do activities, events, guard condition, brief description of State diagram



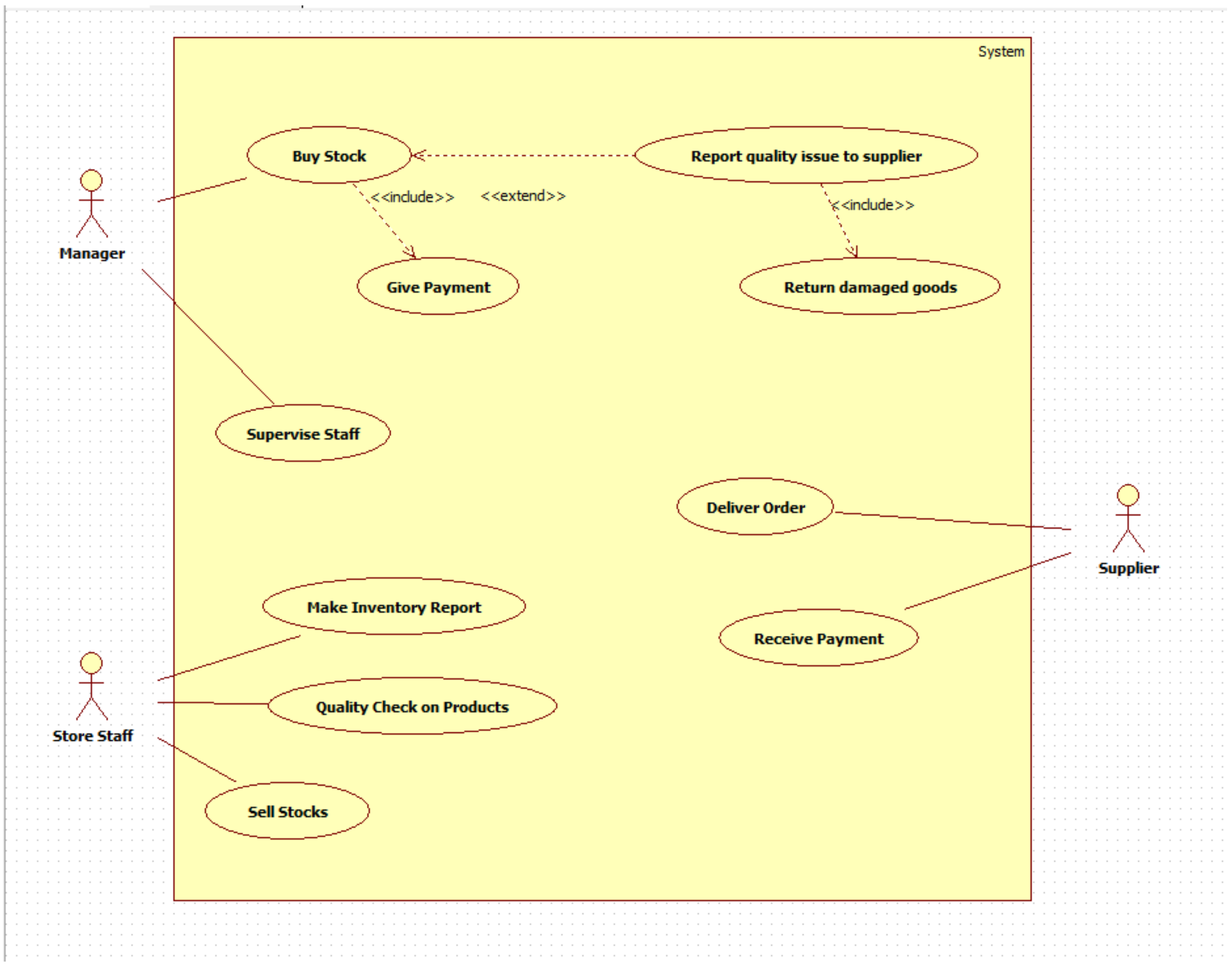
Use Case Diagram

Illustrates system interactions for Inventory Managers monitoring stock, Procurement Staff handling orders, and Warehouse Operators managing receipts. Extend relationships handle low-stock alerts and emergency replenishment scenarios. Include relationships connect purchase order generation with supplier notification and delivery tracking for complete inventory cycle management.

Simple Use Case Diagram



Advance Use Case Diagram



Sequence Diagram

Details chronological interactions during stock replenishment between Inventory System, Supplier, and Warehouse. Shows message sequences for low-stock detection, purchase order generation, supplier confirmation, delivery scheduling, and quality inspection. Includes exception handling for delayed shipments and quality rejections, demonstrating complete order fulfillment workflow.

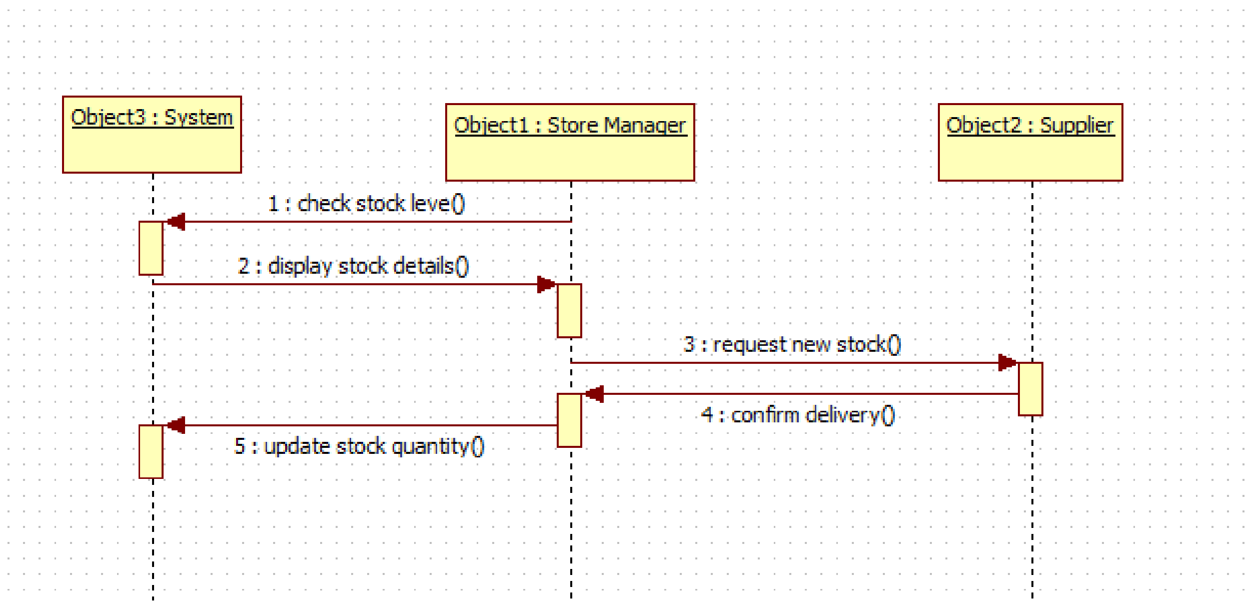
<u>Scenario 1: Adding New Stock</u>

1. Pre-condition: Manager is logged in, new items arrived.
2. Main Flow: Manager selects “Add Stock” → enters item details and quantity → system saves the entry.
3. Post-condition: Inventory count increased, stock record created.

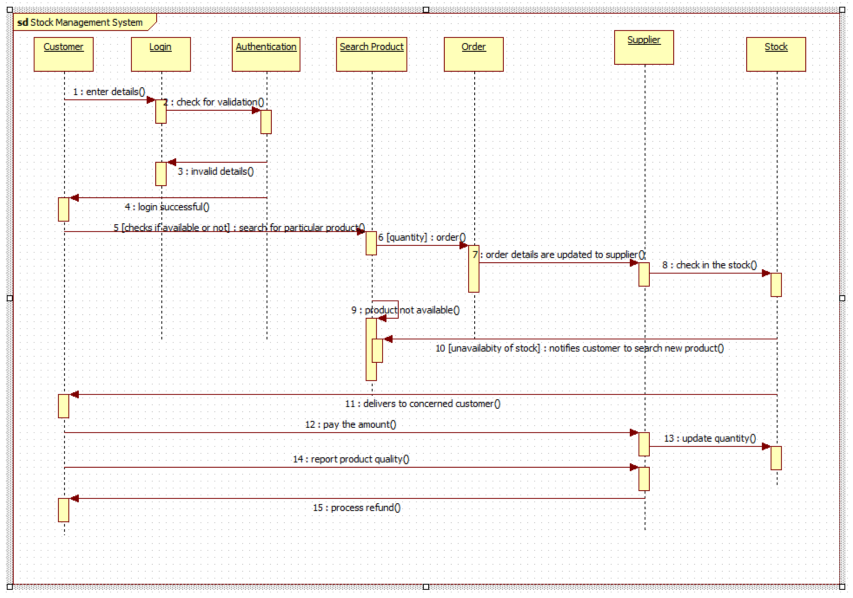
Scenario 2: Removing Stock for Sale

1. Pre-condition: Items are available in inventory.
2. Main Flow: User selects “Remove Stock” → enters product ID and quantity sold → system updates stock count.
3. Post-condition: Inventory reduced, sales record updated.

Simple Sequence Diagram

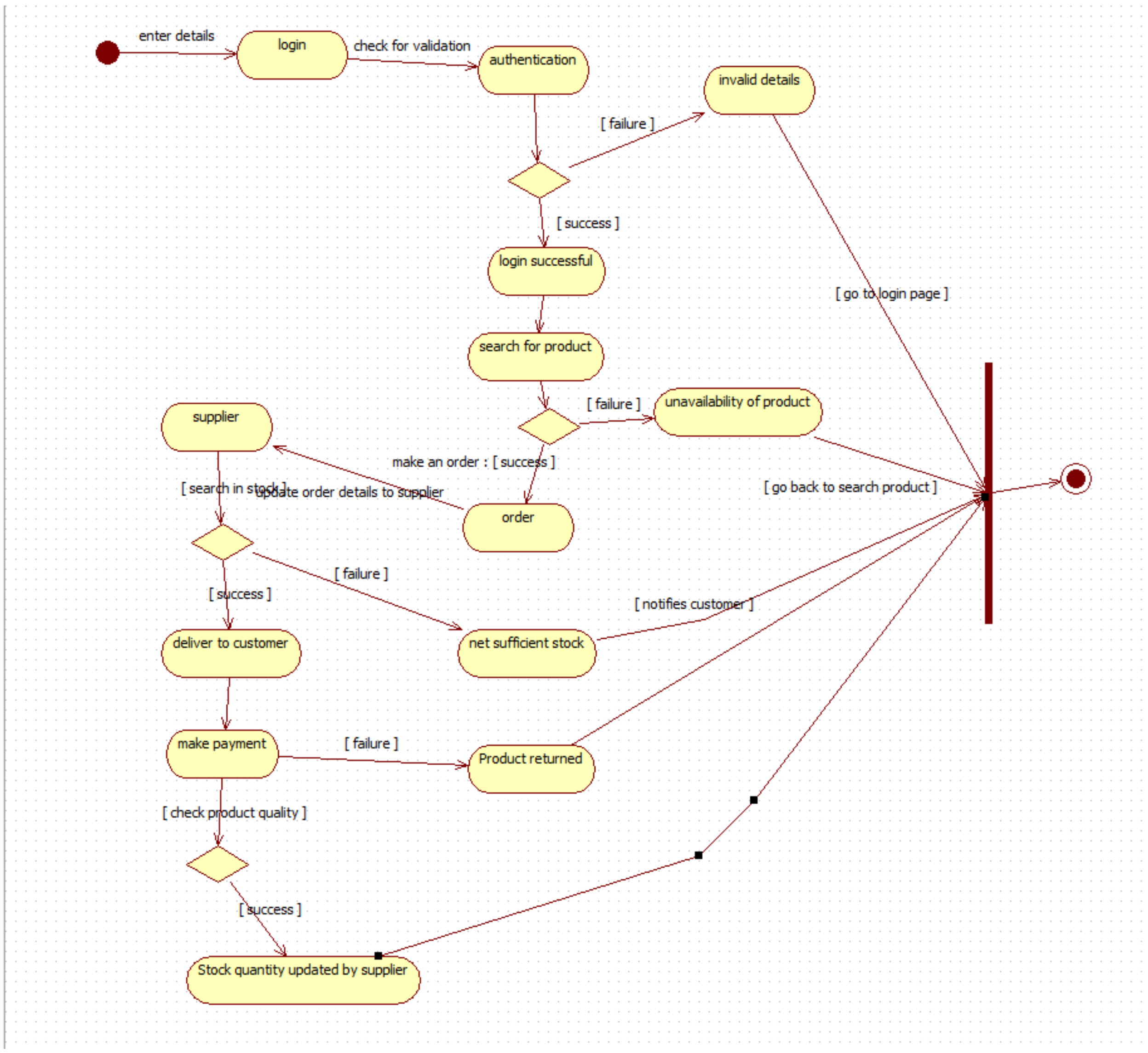


Advance Sequence Diagram


sd Stock Management System
Customer
Login
Authentication
Search Product
Order
Supplier
Stock
1 : enter details()
2 : check for validation()
3 : invalid details()
4 : login successful()
5 [checks if available or not] : search for particular product()
6 [quantity] : order()
7 : order details are updated to supplier()
8 : check in the stock()
9 : product not available()
10 [unavailabity of stock] : notifies customer to search new product()
11 : delivers to concerned customer()
12 : pay the amount()
13 : update quantity()
14 : report product quality()
15 : process refund()

Activity Diagram

It begins with user authentication, followed by product search and order placement. The process includes handling order failures, successful deliveries, payment processing, and customer notifications. It also covers product returns, quality checks, and inventory updates by the supplier, ensuring seamless stock management.



5. Passport Automation System

Problem Statement

The manual passport application and verification process is time-consuming, labor-intensive, and prone to errors and inconsistencies. An automated Passport Automation System is essential to provide faster, more transparent, secure, and efficient passport services to citizens while reducing administrative burden.

Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this document is to comprehensively define the functional and technical requirements of the Passport Automation System, authorizing proper objectives, delineating scope boundaries, and specifying deliverables for all stakeholders involved.

1.2 Scope

The document describes the complete passport issuance process automation, including application processing, verification workflows, appointment management, and production tracking along with detailed time and cost estimates for implementation.

1.3 Overview

The Passport Automation System streamlines and optimizes the entire passport application, verification, approval, and transmission process, significantly reducing manual intervention, improving processing accuracy, and inspecting operational efficiency throughout the passport lifecycle.

2. General Description

The Passport Automation System is designed to serve multiple user roles with specialized functionality:

a) **Passport Applicants** can complete online application forms, upload required documents, schedule appointments, make online payments, track application status, and receive electronic notifications throughout the process.

b) **Verification Officers** can review application details, validate supporting documents, conduct background checks, update application status, request additional information, and recommend approvals or rejections based on established criteria.

c) **Passport Administrators** can manage user accounts, monitor application workflows, generate operational reports, track performance metrics, manage appointment schedules, and oversee system configuration and security settings.

d) **Passport Issuance Staff** can process approved applications, manage passport printing, coordinate quality control, handle dispatch operations, and maintain comprehensive issuance records with tracking capabilities.

The system will comprehensively cater to applicants, government staff, and administrators by enabling seamless online applications, efficient appointment scheduling, robust document verification, secure biometric processing, and systematic passport issuance with end-to-end status tracking.

3. Functional Requirements

3.1 Application Management

The system allows users to fill and submit complete passport applications online with form validation, save draft capability, and payment integration. Provide real-time tracking status for applications through all process stages with automated notification services.

3.2 Document Verification

The system should enable secure upload and validation of required documents with format and size restrictions. Implement robust verification workflows, integrate with government databases for identity verification, and maintain audit trails for all verification activities.

3.3 Appointment Scheduling

The system should allow users to schedule, reschedule, and cancel appointments based on availability. Manage payment verification for appointments, optimize slot allocation, and automatically notify applicants of confirmed appointments with reminders.

3.4 Passport Issuance

The system should generate and print secure machine-readable passports with quality control checks. Maintain complete applicant history and issuance records, manage inventory of passport booklets, and track dispatch and delivery status.

4. Interfaces Requirements

4.1 User Interface

The system should provide an easy-to-use, intuitive interface for applicants and staff with clear navigation, accessibility features, and multilingual support. Must be fully accessible via web browsers and mobile devices with responsive design.

4.2 Integration Interfaces

The system shall ensure secure integration with government identity databases, police verification systems, payment gateways for application fees, and biometric devices for identity confirmation and photo capture.

5. Performance requirements

5.1 Response Time

The system shall process passport applications and verification responses within 2 seconds for standard transactions, ensuring smooth user experience even during peak demand periods.

5.2 Scalability

The system should be capable of handling up to 10,000 concurrent applications and user sessions during peak hours without performance degradation, supporting nationwide deployment.

5.3 Data integrity

The system should ensure complete accuracy, consistency, and security of applicant data throughout the entire passport lifecycle, with comprehensive backup and recovery mechanisms.

6. Design Constraints

6.1 Hardware limitations

The system must work seamlessly with standard government server infrastructure, biometric capture devices, document scanners, photography equipment, and secure printing systems.

6.2 Software boundaries

The system must be implemented using enterprise relational databases with strict data governance, and have implementation using Java/Spring Boot framework for robustness, security, and maintainability.

7. Non-functional attributes

7.1 Security

The system shall implement strong encryption, access controls, and audit mechanisms to protect sensitive applicant data throughout the entire application and storage lifecycle.

7.2 Reliability

The system shall ensure high uptime availability during government working hours with redundant systems, automated backups, and disaster recovery procedures for continuous operation.

7.3 Scalability

The system shall be architecturally designed to easily expand and support more passport centers, additional services, and increased applicant volumes without significant reengineering.

7.4 Accessibility

The system must ensure comprehensive web and mobile support for accessibility, complying with government digital service standards and accessibility regulations.

7.5 Usability

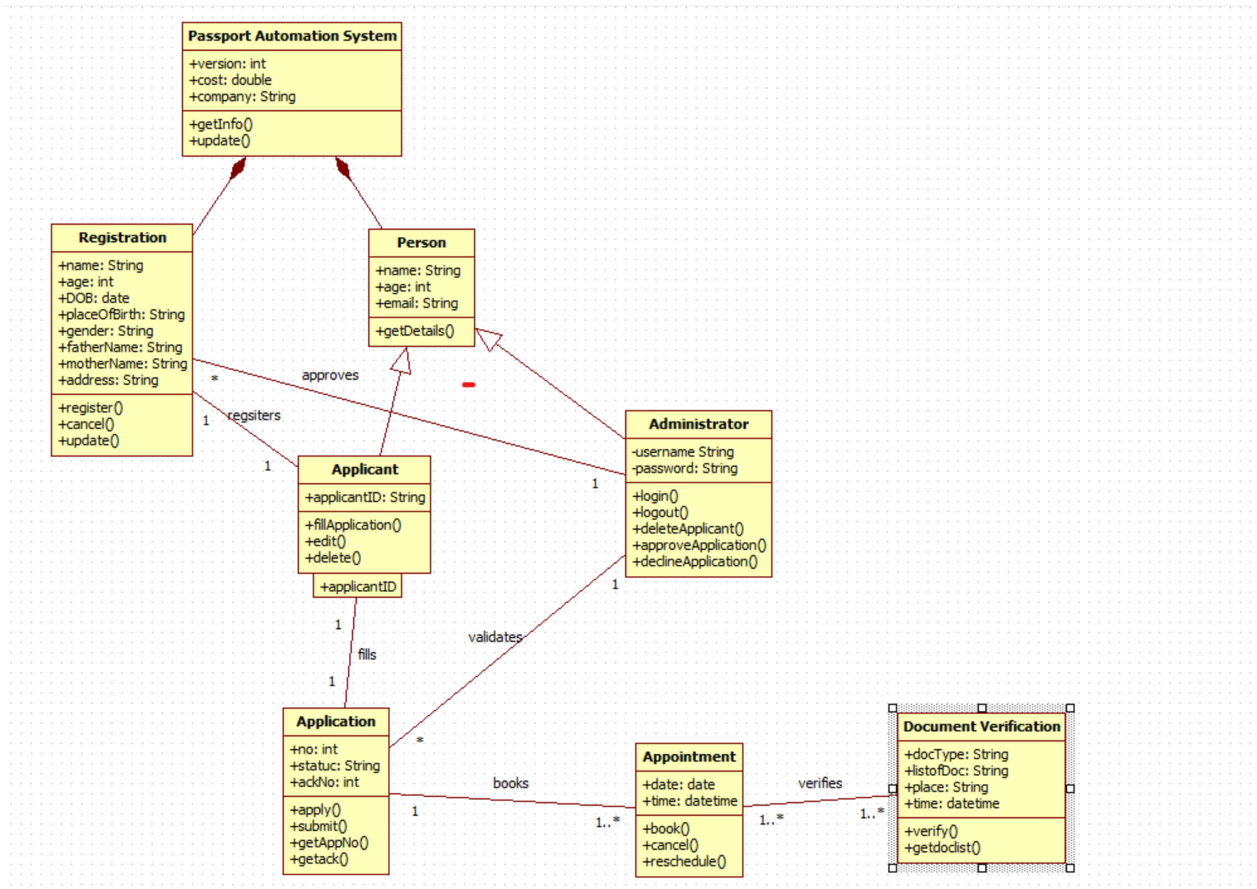
The system must present a simple, intuitive interface for non-technical users with clear instructions, contextual help, and progressive disclosure of complex information.

8. Preliminary schedule and budget

The Passport Automation System is estimated to require 6 months for complete implementation with a comprehensive budget of $150,000 including planning, development, security testing, compliance certification, training, and deployment. The project timeline includes 1 month for requirements analysis, 3 months for development, 1 month for security testing, and 1 month for deployment and user training.

Class Diagram

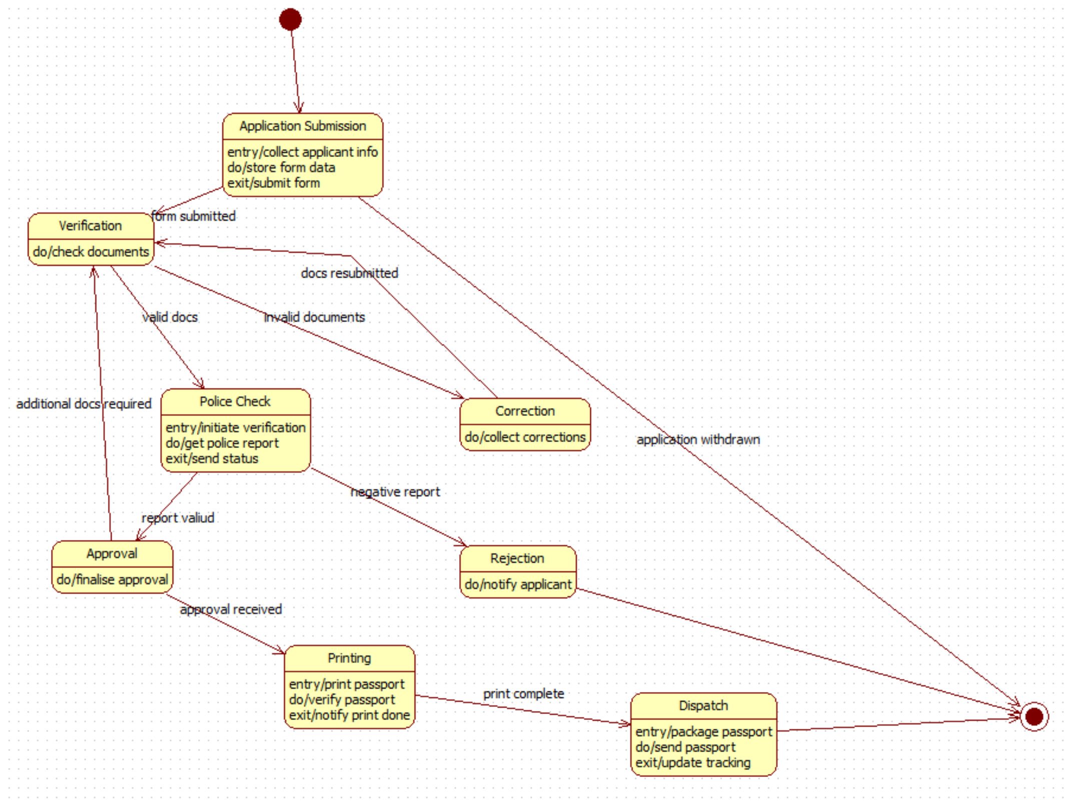
Core classes include Person, Applicant (who registers and applies), Administrator (who approves/declines applications), and Application itself. The process involves registration, document verification, and appointment scheduling, outlining the key entities and their relationships for managing passport issuance.



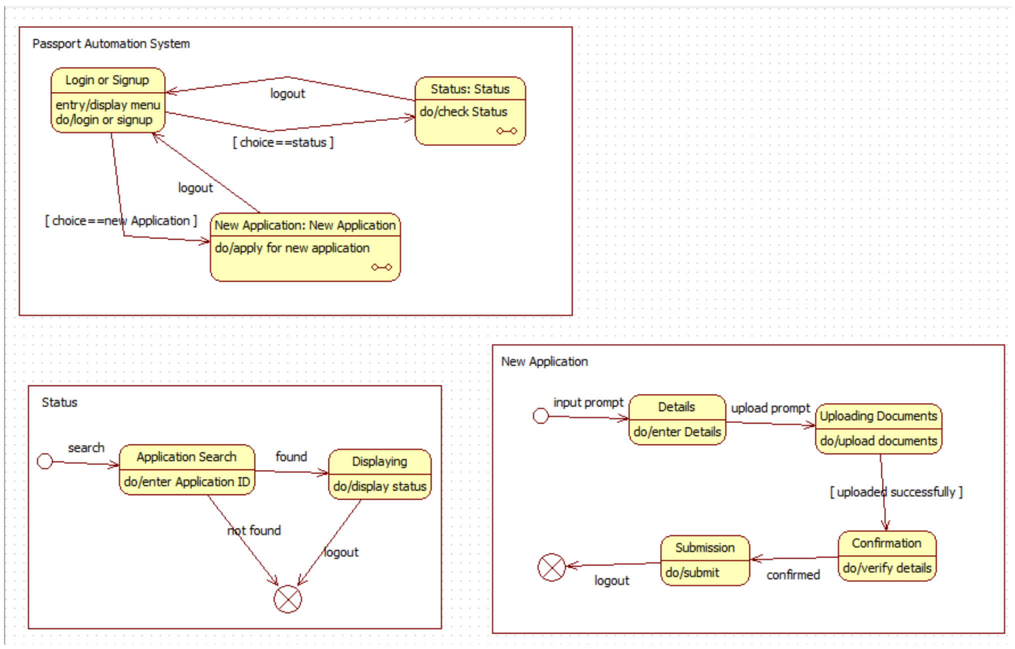
State Diagram

The simple user journey involves logging in, choosing to check status or apply, and then logging out. The advanced, detailed state model covers the entire backend process from application submission and document verification to police checks, approval, printing, and final distribution, including paths for rejection or requests for correction.

Simple State Diagram



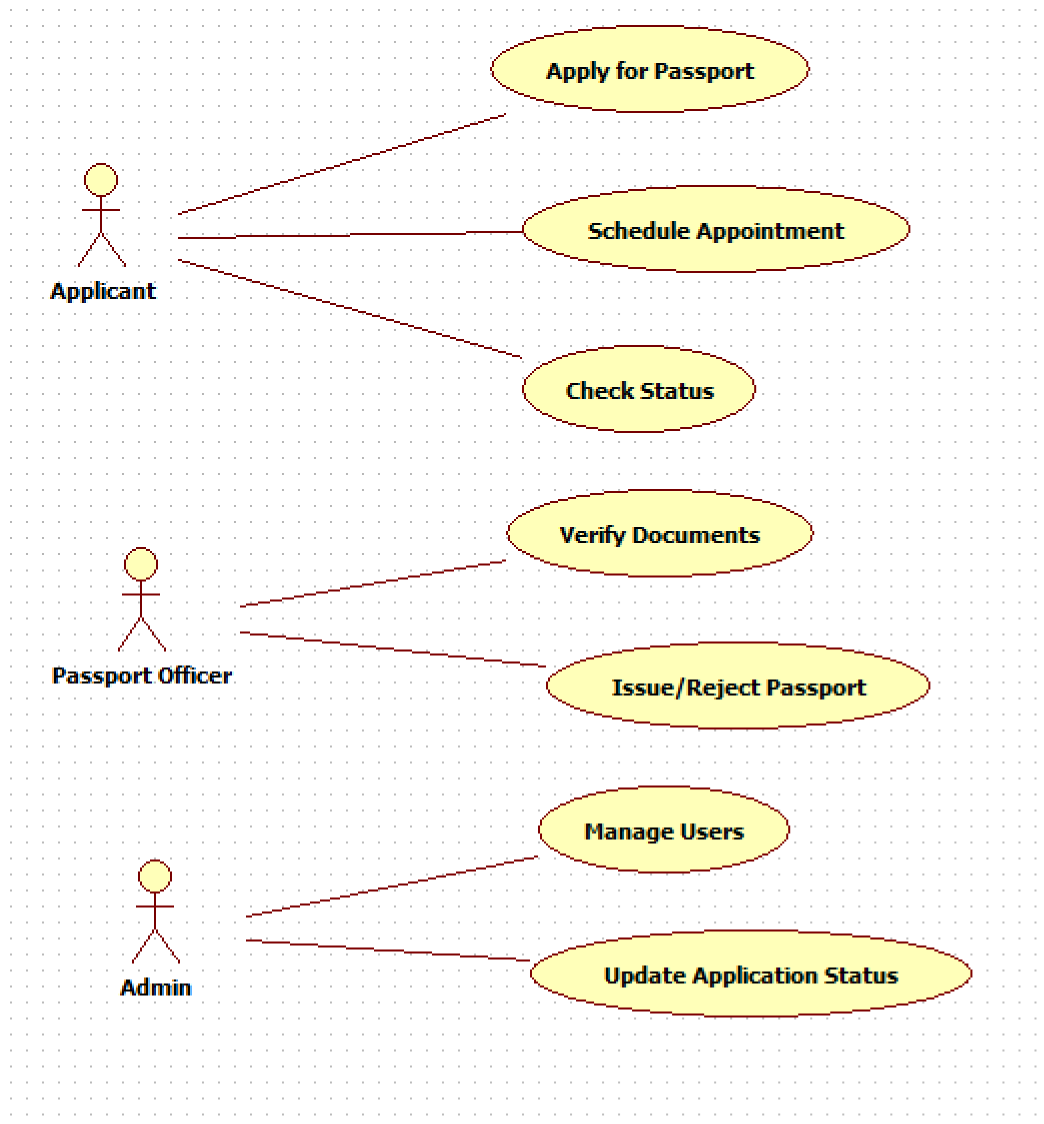
Advance State Diagram



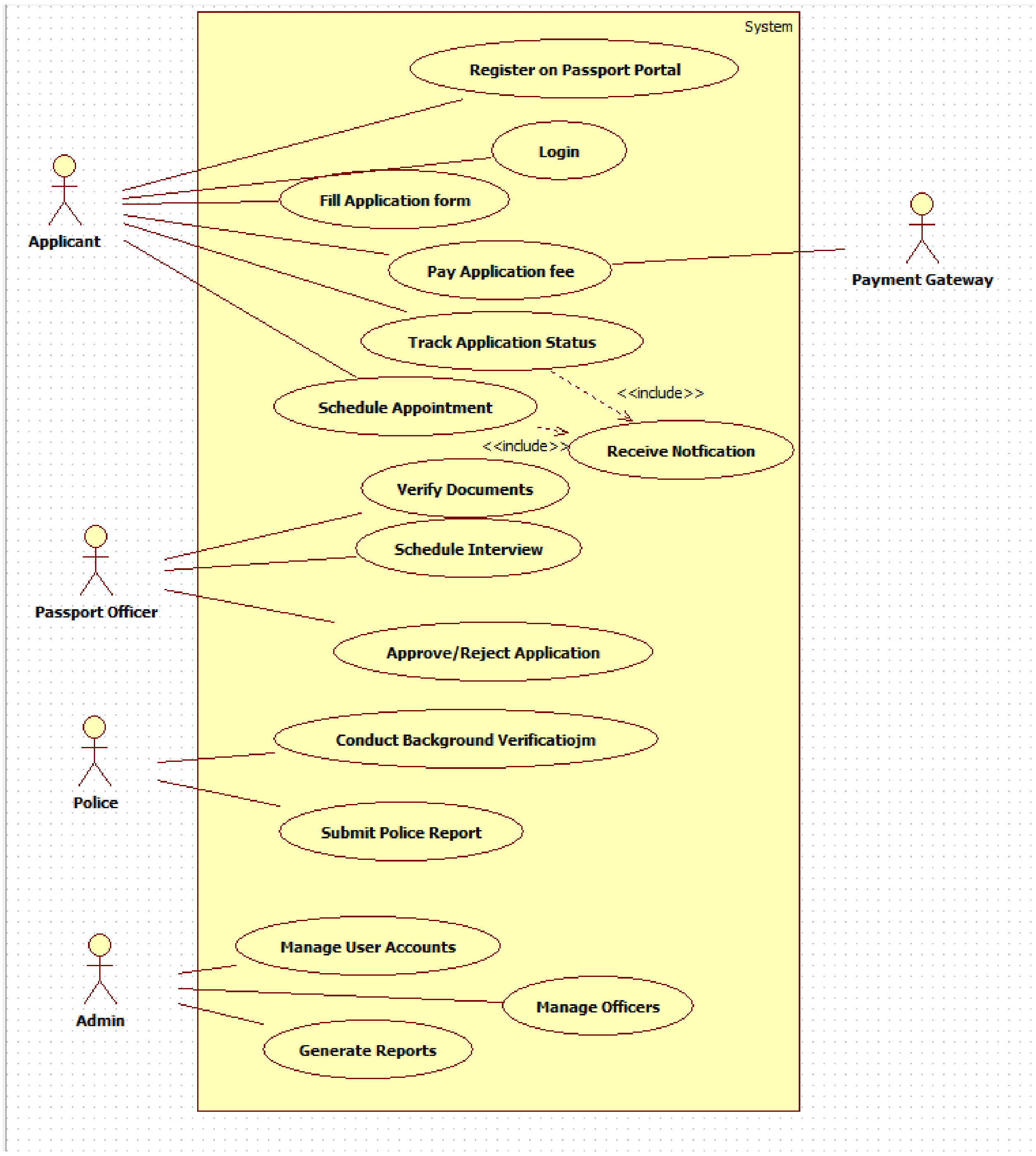
Use Case Diagram

The Applicant can apply for a passport, schedule an appointment, check their application status, and verify documents. The Passport Officer is responsible for issuing or rejecting passports, managing user accounts, and updating the status of applications.

Simple Use Case Diagram



Advance Use Case Diagram



Sequence Diagram

The Applicant logs in and submits details to the Passport Administrator, who stores them. The Regional Administration and Police then verify these details. Upon successful verification, the Database is updated, and the passport is issued. The sequence shows the interaction between all key system entities.

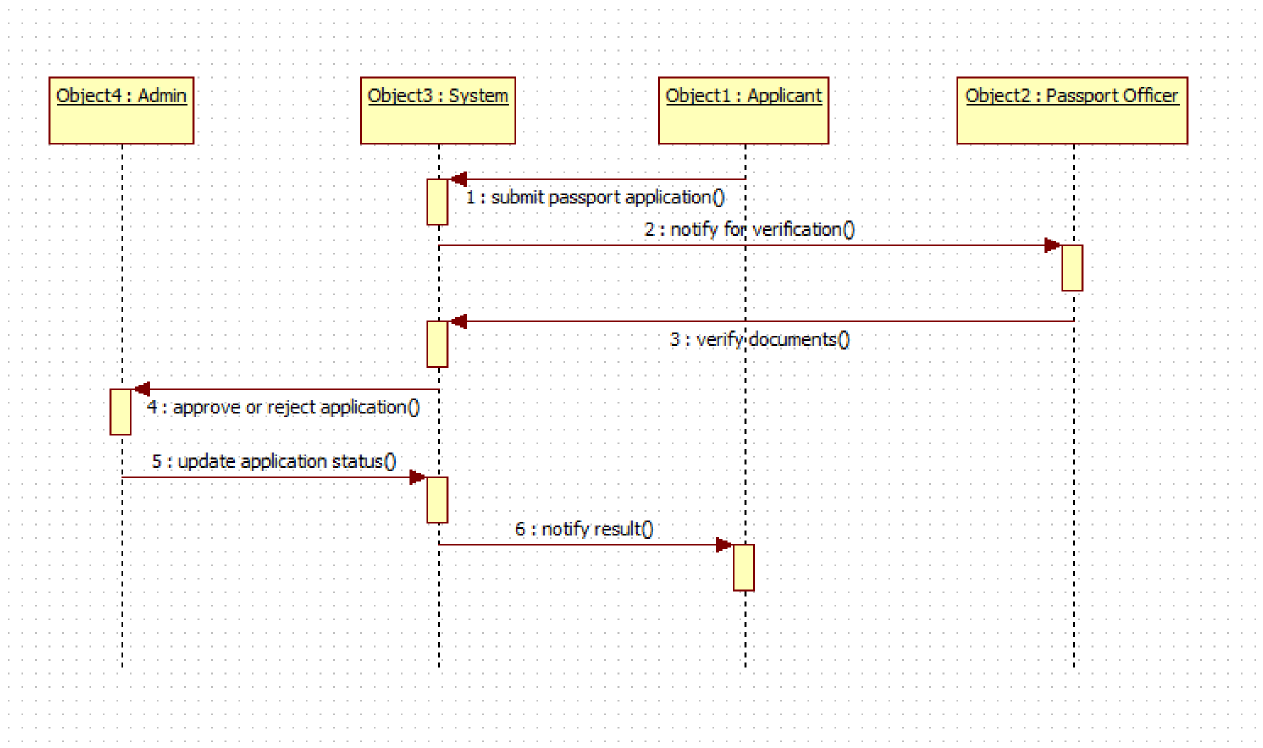
Scenario 1: New Passport Application

1. Pre-condition: User is registered on the portal.
2. Main Flow: User selects "Apply for Passport" → fills application → uploads documents → submits form.
3. Post-condition: Application record created, status set to "Under Review."

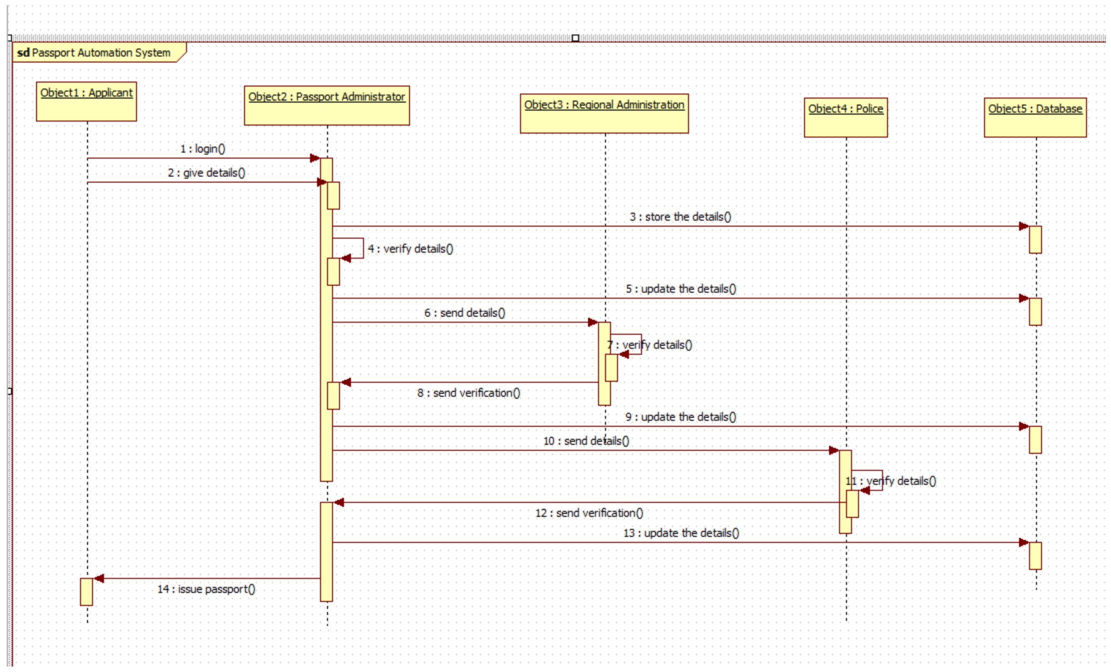
Scenario 2: Booking an Appointment

1. Pre-condition: User has submitted an application.
2. Main Flow: User opens appointment section → selects preferred date → system checks availability → appointment confirmed.
3. Post-condition: Appointment slot booked and saved, confirmation sent.

Simple Sequence Diagram

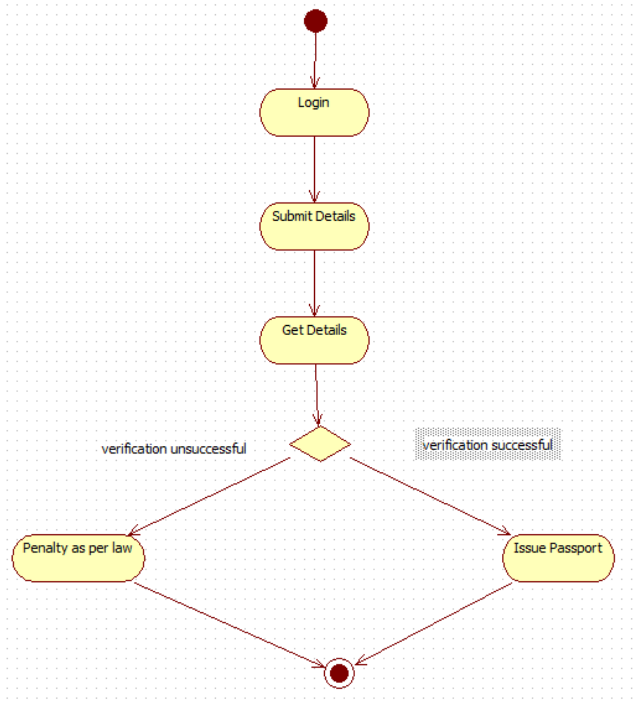

Object4 : Admin
Object3 : System
Object1 : Applicant
Object2 : Passport Officer
1 : submit passport application()
2 : notify for verification()
3 : verify documents()
4 : approve or reject application()
5 : update application status()
6 : notify result()

Advance Sequence Diagram



Activity Diagram

It begins with login and detail submission, followed by a verification check. If verification is successful, the passport is issued. If unsuccessful, a penalty is applied as per law. The process clearly defines the two possible outcomes after the verification stage.


Login
Submit Details
Get Details
verification unsuccessful
verification successful
Penalty as per law
Issue Passport