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

Lab Record

OBJECT-ORIENTED MODELING

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

ALOK AERI
1BM23CS024

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 Analysis and Design(22CS6PCSEO) laboratory has been carried out by Alok Aeri(1BM23CS024) during the 5th Semester August 2025-December 2025

.

Signature of the Faculty Incharge:

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

Table of Contents

HOTEL MANAGEMENT SYSTEM

1. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.1 Introduction

1.1.1 Purpose

The purpose of the Hotel Management System (HMS) is to computerize the day-to-day activities of a hotel. The system helps staff to manage room booking, customer check-in and check-out, room services, food orders, billing and payments in a single application.

This SRS describes the functional and non-functional requirements of the system in a clear and simple form.

1.1.2 Scope

The Hotel Management System will be used by hotel receptionists, managers, restaurant staff and customers.

The main features are:

- Manage hotel branches, rooms and room types.
- Allow customers to reserve rooms and check room availability.
- Support check-in and check-out operations.
- Record food orders and other room services.
- Generate bills and accept different payment modes.
- Maintain staff details and compute their salary.
- Maintain stock information (for example, food items or other supplies).

The system is meant for a single hotel with multiple branches. It will be used as a small to medium-scale desktop/web application for academic and training purposes.

1.1.3 Definitions, Acronyms and Abbreviations

- HMS – Hotel Management System
- GUI – Graphical User Interface
- DB – Database

1.2 General Description

1.2.1 Product Perspective

The HMS is a standalone application that connects to a central database.

It replaces the manual, paper-based process of maintaining registers for booking, guests, billing and staff management.

The system will have separate interfaces for receptionists, managers and restaurant staff, with role-based access control.

1.2.2 Product Functions

At a high level, the system will provide the following functions:

- Room Management
- Add, update and view room details and room types.
- Mark rooms as available, reserved, occupied or under maintenance.
- Customer and Booking Management
- Register new customers and store their contact details.
- Search for empty rooms based on room type and dates.
- Reserve rooms, confirm bookings, cancel bookings.
- Perform check-in and check-out.
- Service and Food Management
- Record food items and other services requested by a customer.
- Map orders to the correct room and customer.
- Transfer kitchen orders to the restaurant/chef.
- Billing and Payment
- Calculate room charges, service charges and taxes.
- Generate bill at check-out.
- Accept multiple payment modes such as cash, credit card, debit card and UPI.
- Staff and Branch Management
- Maintain details of employees, including name, address and date of birth.
- Link employees to hotel branches and compute their salary.
- Stock Management

- Maintain stock of food items and other supplies.
- Check and update stock based on orders.

1.2.3 User Characteristics

- **Customer –**

Basic computer or kiosk usage; interacts mainly for room reservation, check-in, check-out and payments.

- **Receptionist –**

Regular computer user; responsible for bookings, customer record entry, and billing.

- **Manager –**

Experienced user; handles staff details, salary, room and price configurations, and system reports.

- **Restaurant Staff / Chef –**

Basic user; views food orders and updates order status.

1.2.4 Constraints and Assumptions (High Level)

- The system assumes reliable network connectivity between branches and the central database, if deployed in multiple locations.
- All monetary values are stored in a fixed currency (for example, INR).
- Users are expected to have valid login credentials created by the manager or system admin.

1.3 Functional Requirements

FR1: User Authentication

- The system shall allow staff to log in using a username and password.
- Only authenticated staff can access booking, billing and management functions.

FR2: Manage Room Types and Rooms

- The system shall allow the manager to define room types (Single AC, Single Non-AC, Double AC, Double Non-AC, etc.).
- The system shall allow the manager to add, edit and delete room records with room number, room type and status.

FR3: Search Room Availability

- The system shall allow the receptionist to search for empty rooms based on date range, room type and branch.
- The system shall show a message if no room is available.

FR4: Customer Registration and Booking

- The system shall store customer details such as name, mobile number and address.
- The system shall allow the receptionist to book a selected room for a customer with check-in and check-out dates.

FR5: Check-in and Check-out

- The system shall record check-in when the customer arrives.
- The system shall record check-out, close the room booking and mark the room as available again.

FR6: Room Services and Food Orders

- The system shall allow staff to add room service requests and food orders for a checked-in customer.
- The system shall update the bill whenever a new service or food item is added.

FR7: Billing and Payment

- The system shall compute the bill including room charges, food charges and other services.
- The system shall support payments using the allowed payment modes (cash, credit card, debit card, UPI).
- The system shall generate a final bill/receipt after successful payment.

FR8: Employee and Salary Management

- The system shall maintain employee details like ID, name and date of birth.
- The system shall link each employee to a branch and store salary details.
- The system shall compute monthly salary for each employee.

FR9: Stock Management

- The system shall maintain a list of stock items with ID and name.
- The system shall update stock when items are consumed.

- The system shall allow staff to check stock levels.

FR10: Feedback Handling (Optional)

- The system may record customer feedback about their stay and services.

1.4 Interface Requirements

1.4.1 User Interface Requirements

- The system shall provide a simple GUI with menus for Booking, Customer, Rooms, Services, Billing and Reports.
- Forms should follow a clear layout with labels, text fields, drop-down lists and buttons for Save, Update, Delete, Search, etc.
- Error messages and confirmations should be displayed in simple language, for example "Room not available" or "Payment successful".

1.4.2 Hardware Interface Requirements

- A standard computer or terminal for receptionist and manager.
- Optional networked terminals in restaurant/kitchen for viewing food orders.
- Printer support for printing bills and reports.

1.4.3 Software Interface Requirements

- Database management system (for example, MySQL / PostgreSQL / any RDBMS).
- Operating system such as Windows/Linux.
- The application may be implemented using Java, .NET or any object-oriented language as per lab requirement.

1.4.4 Communication Interface Requirements

- If multiple branches are used, the system will need a local network or internet connection for accessing the central database.
- Basic TCP/IP network stack will be used for communication.

1.5 Performance Requirements

- **3–5 seconds**The system should display room availability results within for normal load.

- **20–50 concurrent users**The system should support at least in a typical hotel environment.
- Database operations such as booking, check-in and bill generation should complete without noticeable delay for the user.
- The system should be able to store details of several years of customers, bookings and payments without loss of performance, assuming regular database maintenance.

1.6 Design Constraints

- **object-oriented design principles**The system must follow as per OOMD lab, using classes such as Hotel, Room, Customer, Employee, Payment, etc.
- The design must support extensibility so that new room types, services or payment modes can be added later.
- Security constraints: only authorized users can make changes to bookings, payments and staff details.
- The user interface should be simple enough so that new staff can learn it quickly.

1.7 Non-Functional Attributes

Reliability

- The system should handle common errors gracefully (for example, wrong input, unavailable room, payment failure).
- Data must be stored reliably in the database with proper backup mechanisms.

Usability

- Screens should be consistent in layout and navigation.
- Help text or tooltips should be provided for key fields where needed.

Security

- Passwords should not be visible while typing.
- Only logged-in staff can access staff management and stock management modules.
- Sensitive data such as payment details should not be exposed to unauthorized users.

Maintainability

- Code should be modular (layered architecture with UI, business logic and data access).
- New features, such as loyalty points or online booking, should be easy to add.

Portability

- The application should be portable across different operating systems if implemented using a cross-platform technology.

1.8 Schedule and Budget

Proposed Schedule (Example for Lab Project)

- Week 1–2: Requirement gathering and preparation of SRS.
- Week 3–4: UML modelling (use case, class, sequence, activity and state diagrams).
- Week 5–7: Implementation of core modules (room management, booking, check-in/check-out).
- Week 8–9: Implementation of billing, payments and reports.
- Week 10: Testing, fixing defects and preparation of final documentation.

Proposed Budget (Conceptual)

- Hardware: existing lab computers (no extra cost assumed).
- Software: open-source tools (StarUML, database, IDE).

---

2. CLASS DIAGRAM

The class diagram for the Hotel Management System contains the following main classes:

- **Hotel –**

Represents the hotel as a whole, with attributes like id, hname, hno. It has operations like CheckRA() (check room availability), bookno() and GenerateBill().

- **Branch –**

Represents different branches of the hotel, with attributes bName and bAddress. A Hotel aggregates one or more Branch objects (1-to-many relationship).

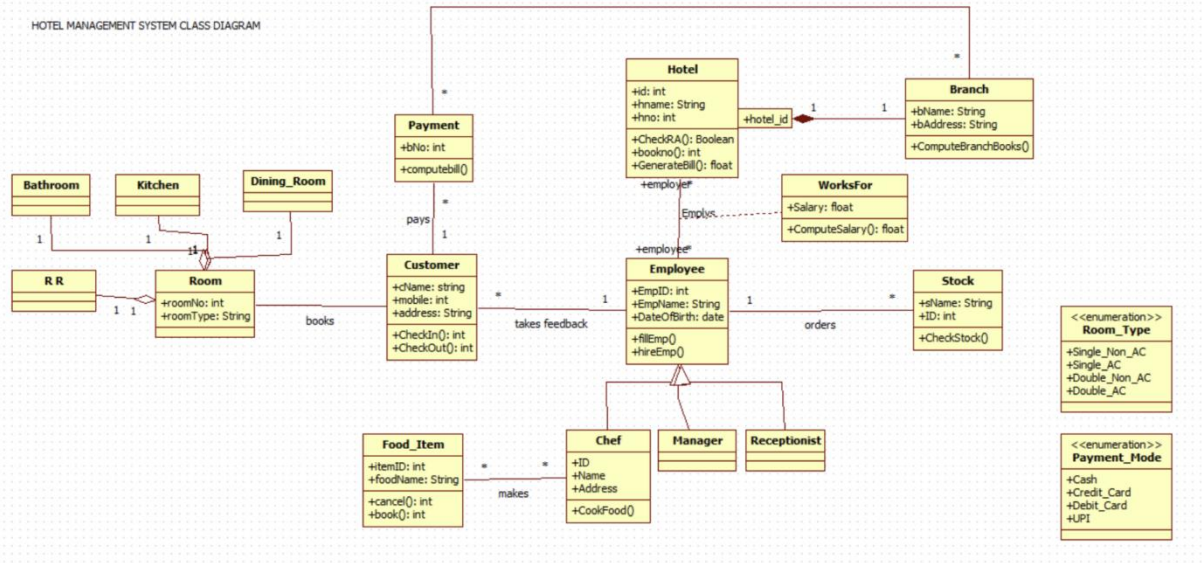

fig A1 - Advanced Class Diagram of Hotel Management System

- **Room –**

Contains roomNo and roomType. Each room is part of a hotel/branch and is associated with Customer and Payment through bookings.

- **Bathroom, Kitchen, Dining_Room, R R –**

These are facilities related to a room. They are connected to Room using a composition relationship, meaning that if the room is removed, its facilities also cease to exist.

- **Customer –**

Stores customer data such as cName, mobile, address, CheckIn() and CheckOut(). A customer books one or more rooms.

- **Payment –**

Represents payment details with attribute bNo and operation computebill(). Each Customer makes one or more payments for their bookings.

- **Employee –**

Has attributes EmpID, EmpName and DateOfBirth. It provides operations such as fillEmp() and hireEmp().

- **Chef, Manager, Receptionist –**

These are specializations (subclasses) of Employee and use inheritance. They inherit common employee attributes and behaviors and add their own roles (for example, CookFood() in Chef).

- **WorksFor –**

Connects Employee to the hotel with attributes Salary and ComputeSalary(). It shows that an employee works for a specific hotel/branch.

- **Food_Item –**

Represents food items that can be ordered. It has itemID, foodName and operations cancel() and book(). A Chef makes food items, and customers request them as room service.

- **Stock –**

Holds information about stock items with sName, ID and CheckStock() operation. It is related to Employee or Manager who updates the stock.

- **Room_Type (enumeration) –**

Contains constant values such as Single_Non_AC, Single_AC, Double_Non_AC, Double_AC. This enumeration is used by the Room class.

- **Payment_Mode (enumeration) –**

Contains values Cash, Credit_Card, Debit_Card, UPI. This is used by the Payment class.

Key Relationships

- Association:
- **Customer – Room:** A customer books one or more rooms; a room can be booked by many customers over time.
- **Customer – Payment:** A customer makes payments; payment belongs to a customer.
- **Employee – WorksFor – Hotel/Branch:** Shows which employee works for which hotel/branch.
- **Employee – Stock:** Employees, usually managers or storekeepers, check and update stock.
- Aggregation/Composition:
- **Hotel ◇ Branch:** Hotel aggregates its branches.
- **Room ◆ Bathroom / Kitchen / Dining_Room / R R:** A room is composed of these facilities.
- Generalization (Inheritance):
- **Employee → Chef / Manager / Receptionist:** These specialized staff roles inherit properties and behaviors of Employee.

---

3. STATE DIAGRAM

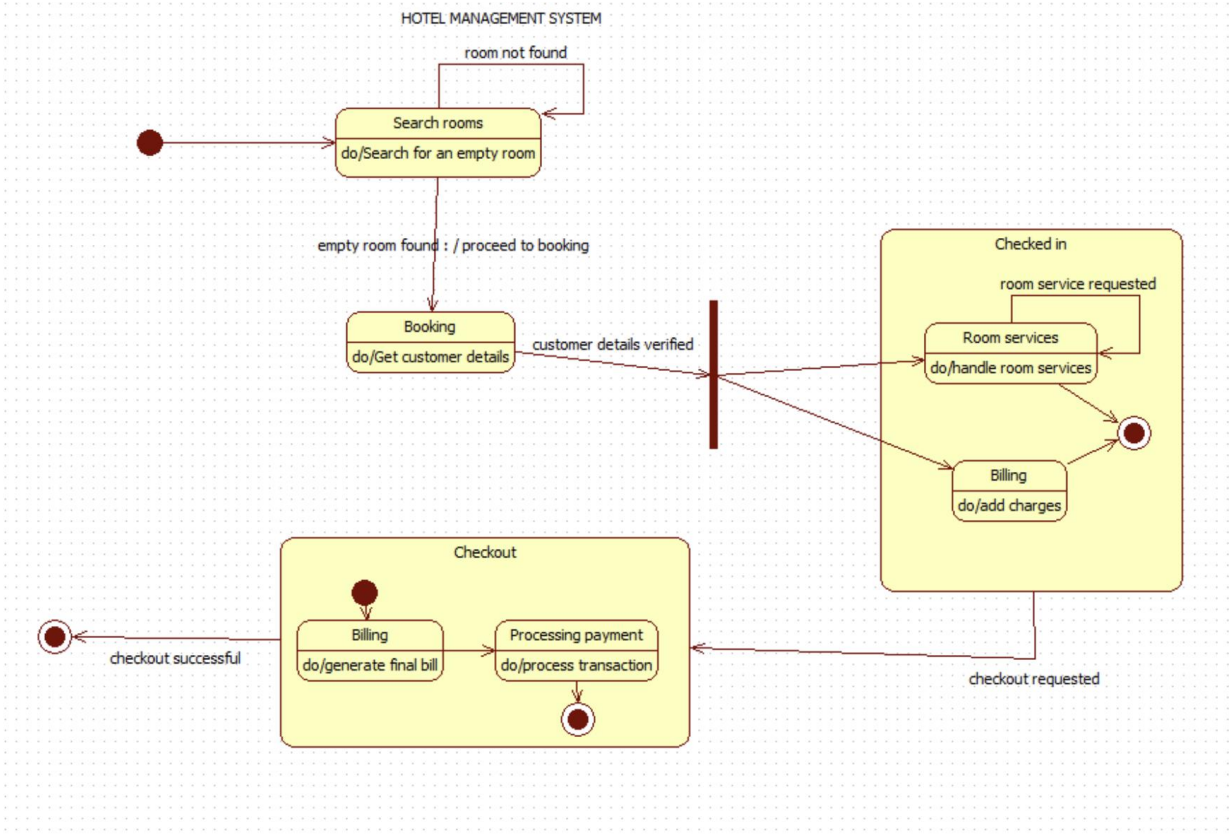

fig A2 - Advanced State Diagram of Hotel Management System

The state diagram models the life cycle of a room booking in the Hotel Management System.

1. The process starts in the "Search rooms" state where the system searches for an empty room based on user input.

- If no room is found, control loops back and the user may search again.

2. When an empty room is found, the system moves to the "Booking" state, where customer details are collected and stored.

3. After customer details are verified, the system transitions to the "Checked in" composite state.

- Inside this, there are substates such as "Room services" (handling requests like food, cleaning, etc.) and "Billing" (adding the cost of services).

4. When the customer requests to leave, the system moves to the "Checkout" composite state.

- Here, "Billing" generates the final bill.
- "Processing payment" handles the payment transaction.

5. If the payment is successful, the state moves to the final state, indicating checkout successful and the room is marked as available again.

This diagram clearly shows how the booking moves from searching to booking, staying, and finally checking out.

---

4. USE CASE DIAGRAM

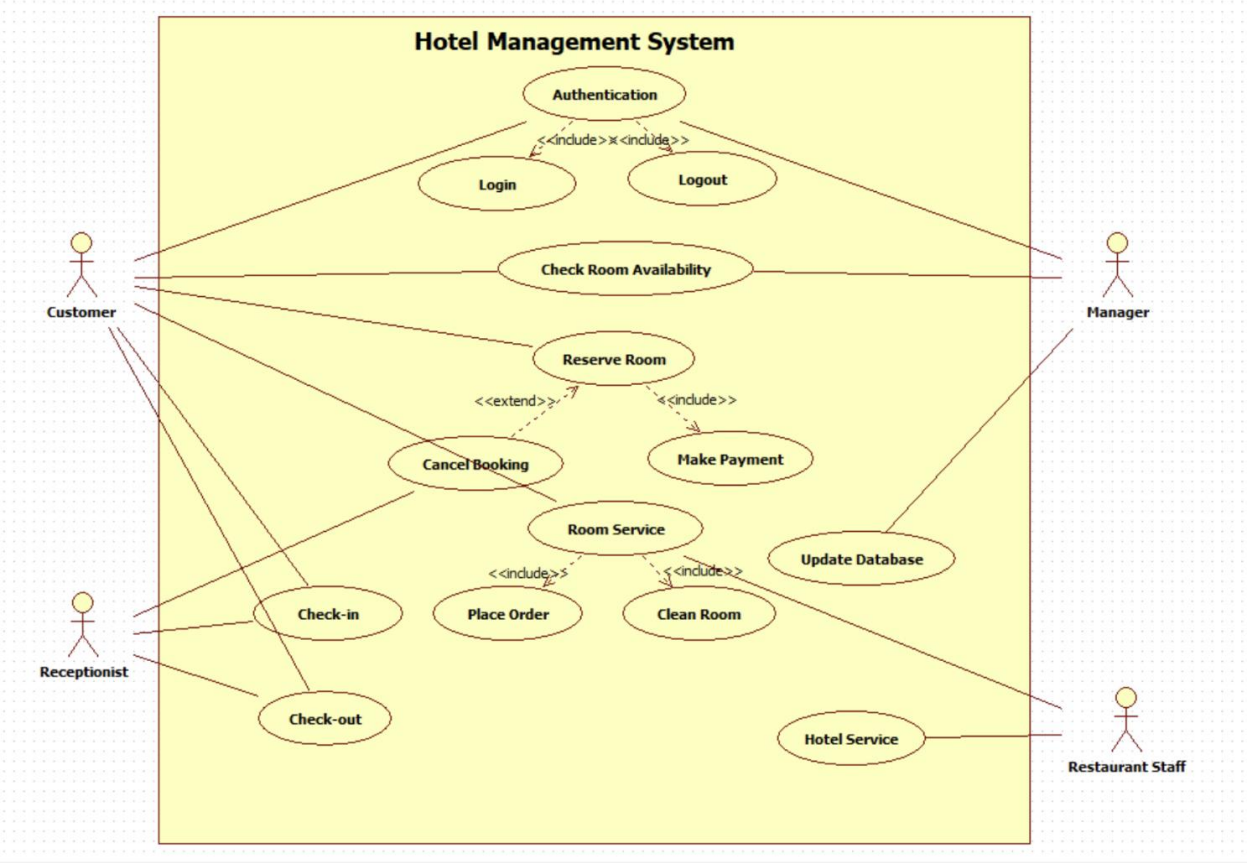

fig A3 - Advanced Use Case Diagram of Hotel Management System

The use case diagram for the Hotel Management System involves the following actors and use cases:

Actors

- **Customer**
- Interacts with the system to log in, check room availability, reserve rooms, cancel bookings, request room service, place orders, and make payments.
- **Receptionist**
- Handles check-in, check-out, booking, cancellation and payment on behalf of customers.
- **Manager**

- Updates the database with new rooms, prices, staff details and stock details.
- Monitors hotel services and reports.
- **Restaurant Staff**
- Handles restaurant-related activities such as hotel service and preparing/serving food based on room orders.

Main Use Cases

- Authentication (with **Login** and **Logout** as included use cases).
- **Check Room Availability** – Check if rooms are available for given dates and room type.
- **Reserve Room** – Reserve rooms for a customer after checking availability.
- **Cancel Booking** – Cancel an existing reservation.
- **Check-in / Check-out** – Perform arrival and departure procedures.
- **Room Service** – Handle service requests like food orders and cleaning (includes **Place Order** and **Clean Room**).
- **Make Payment** – Accept payment for room charges and services.
- **Update Database** – Manager updates information such as room data, rates, staff and stock.
- **Hotel Service** – General services provided by restaurant staff for in-house guests.

---

5. SEQUENCE DIAGRAM

1. The **Staff** then **provide services** to the customer during their stay (room service, cleaning, etc.).
2. If the customer decides to cancel, a **Cancel Booking()** request is sent to the **Receptionist**, who then **Update the database()** to reflect the change.
3. The **Receptionist** may then **Retrieve Current Availability** from the **Database** for future bookings.
4. At departure time, the **Customer Ask for room and keys** (or final formalities).
5. The **Customer** then **Pay()** through the **Receptionist**.
6. The **Receptionist** sends **requestInformation()** to the **Database** to compute final charges.
7. The **Database** returns the data, and the **Receptionist retrieve information and generate bill**.
8. Finally, the **Receptionist provide the bill** to the **Customer**.

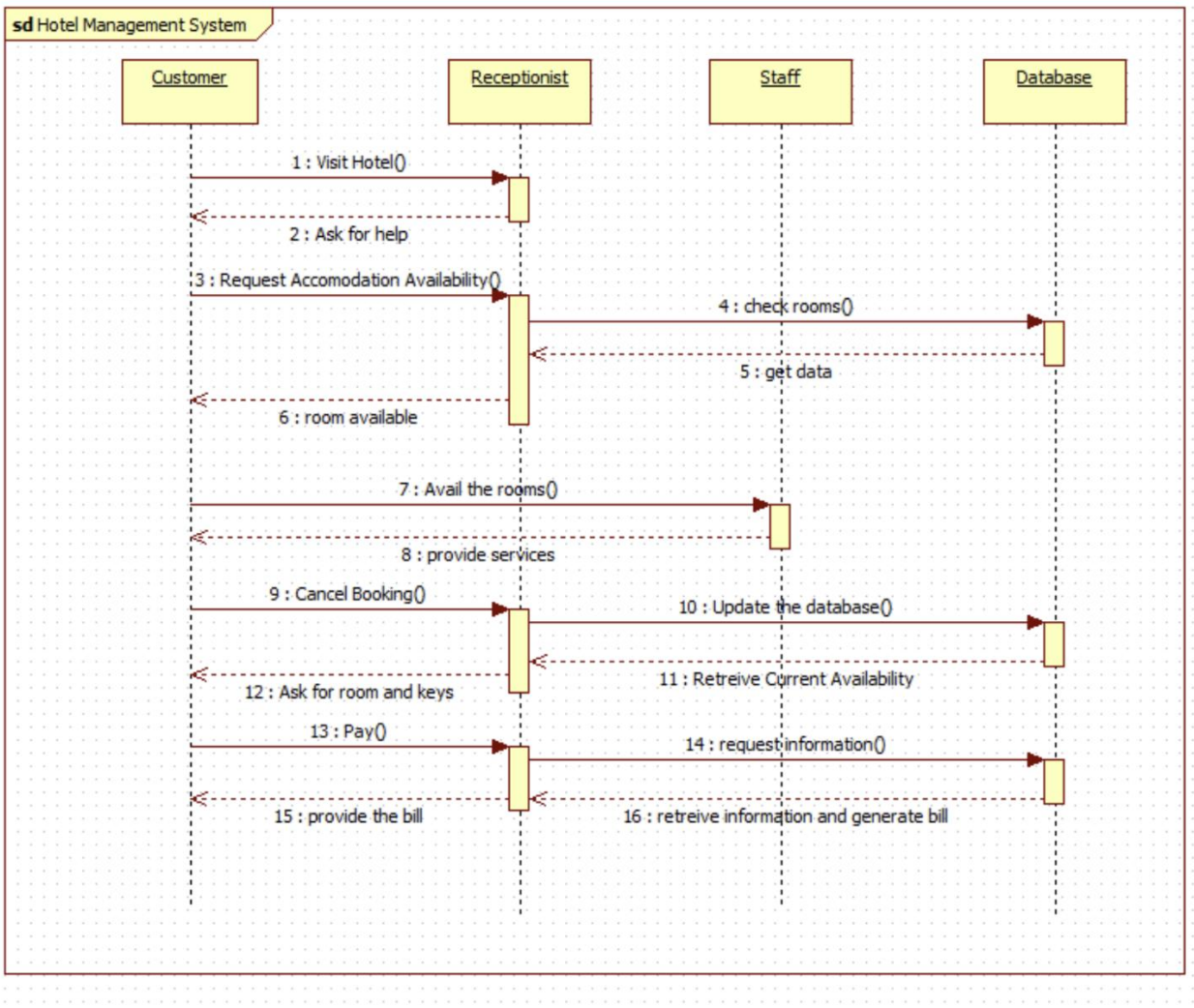

fig A4 - Advanced Sequence Diagram of Hotel Management System

---

6. ACTIVITY DIAGRAM

The activity diagram includes three main swimlanes: Customer, Database, and Payment.

Swimlanes

- **Customer** – Actions performed by the customer such as login, booking room and making payment.
- **Database** – Backend operations such as checking availability and reserving rooms.
- **Payment** – Activities related to payment processing, error handling and cost calculation.

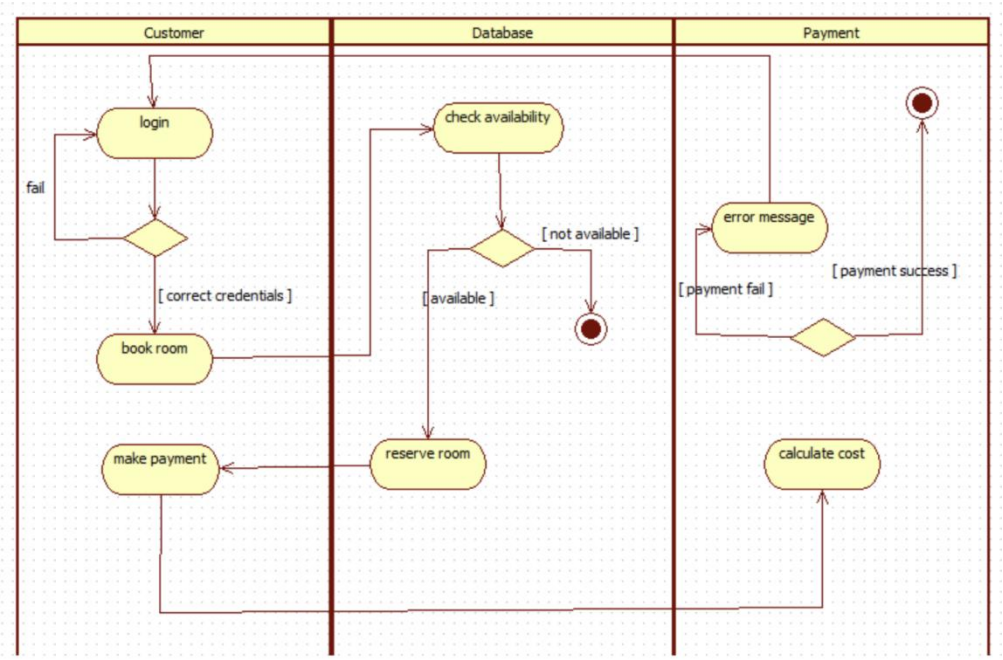

fig A5 - Advanced Activity Diagram of Hotel Management System

Flow Description

1. In the **Customer** lane, the process starts with the **login** activity.

- If credentials are incorrect, the flow goes to a failure end state.
- If credentials are correct, the customer proceeds to **book room**.

2. In parallel, in the **Database** lane, the system performs **check availability** for requested rooms.

- If rooms are **not available**, the flow ends in that lane (no booking is made).
- If rooms are **available**, the system moves to **reserve room**, confirming the booking.

3. After successful reservation, the customer moves to **make payment** in the **Customer** lane.
4. The control then moves to the **Payment** lane where the system first **calculate cost** based on booking and services.
5. A decision is made:

- If **payment fails**, an **error message** is shown and the customer may try again or cancel.
- If **payment succeeds**, the process ends in the **Payment** lane with a successful completion node.

6. The booking is now confirmed, and the room is reserved for the customer.

CREDIT CARD PROCESSING SYSTEM

1. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.1 Introduction

1.1.1 Purpose

The purpose of the Credit Card Processing System is to manage credit card transactions, validate customer details, authenticate card usage, and maintain payments securely. This system automates the interaction between the bank, customer, merchants, and processing units. The aim is to ensure fast, safe, and reliable credit card processing.

1.1.2 Scope

The system supports:

- Credit card issuance and activation
- Secure verification and validation of card details
- Merchant payment processing
- Transaction recording and status tracking
- Customer account linking and balance updates
- Generation of transaction receipts and reports

The system benefits banks, merchants, customers, and clearing house/payment providers. It ensures fast, fraud-free transactions and replaces manual processing methods.

1.1.3 Definitions / Abbreviations

Table 1: Definitions and Abbreviations

Term	Meaning
PAN	Primary Account Number
CC	Credit Card
ATM	Automated Teller Machine
DBMS	Database Management System
UPI	Unified Payments Interface
API	Application Programming Interface

1.2 General Description

1.2.1 Product Perspective

This system works between multiple actors: customer, merchant, bank office, clearing house and payment provider. It acts as a middleware between the user and financial institutions and works as a secure processing platform.

1.2.2 Product Functions

Major functionalities include:

- Create and issue credit cards
- Verify card details and validate customer identity
- Manage bank accounts and link them to cards
- Handle transaction types (purchase, refund, cash advance)
- Secure payment processing
- Generate receipts and update balances
- Track transaction status: pending, authorised, settled, reversed

1.2.3 User Types

Table 2: User Types

User	Role in System
Customer	Uses card for payment or cash withdrawal
Bank	Issues cards, authorizes transactions
Merchant	Accepts payments and initiates requests
Payment Provider	Finalizes and clears payments
Clearing House	Verifies card info and routes payment

1.2.4 Constraints & Assumptions

- Must comply with banking security rules (2FA, encryption, PCI standards)
- Internet connectivity is required
- Database must be highly secure
- UPI/Net banking/Card payments must be supported
- Transactions should be atomic (no partial update)

1.3 Functional Requirements

Table 3: Functional Requirements

No.	Requirement
FR1	The system shall authenticate customers during card usage.
FR2	The system shall validate card number, expiry date and status.
FR3	The system shall process purchase, refund and cash advance transactions.
FR4	The system shall connect with merchant and bank systems during payments.
FR5	The system shall update transaction status after payment.
FR6	The system shall support UPI, NetBanking and Card methods.
FR7	The system shall generate receipts after successful payment.
FR8	The system shall allow banks to block/unblock cards.
FR9	The system shall maintain transaction history for every user.

1.4 Interface Requirements

1.4.1 User Interface

- Simple GUI for banks, merchants and card-holders
- Input fields for card number, amount, PIN, date, etc.
- Clear messages: “Payment successful”, “Incorrect PIN”, “Insufficient Funds”

1.4.2 Hardware Interface

- ATM machine / POS terminal
- Web/mobile application for users
- Printer for receipts

1.4.3 Software Interface

- DBMS (MySQL / PostgreSQL)
- API for merchant-bank communication
- OOP-based software (Java/.NET recommended)

1.4.4 Communication Interface

- HTTPS encrypted communication
- Third-party payment provider integration
- Secure transaction through tokenization / masking PAN

1.5 Performance Requirements

- Must process transactions within 3 seconds
- System should support 500+ concurrent users
- Should handle 24×7 uptime with minimal network failure
- Transaction failure rate should be below 2%

1.6 Design Constraints

- Must follow object-oriented design
- Must use PCI-DSS security standards
- Only authorized users can access transaction logs
- Must support future addition of new payment types

1.7 Non-Functional Attributes

- **Reliability –**

High reliability is required as it handles financial data.

- **Usability –**

Easy UI for customers and merchants.

- **Security –**

Strong encryption, no sensitive data visible.

- **Maintainability –**

Modular code and extendable classes.

- **Availability –**

System must run continuously without fail.

- **Scalability –**

Should support multiple branches and users.

1.8 Schedule & Budget (academic assumption)

Table 4: Schedule

Phase	Duration
Requirements & SRS	2 weeks
UML Modelling	2 weeks
System Design	3 weeks
Implementation	4 weeks
Testing & Report	2 weeks

---

2. CLASS DIAGRAM

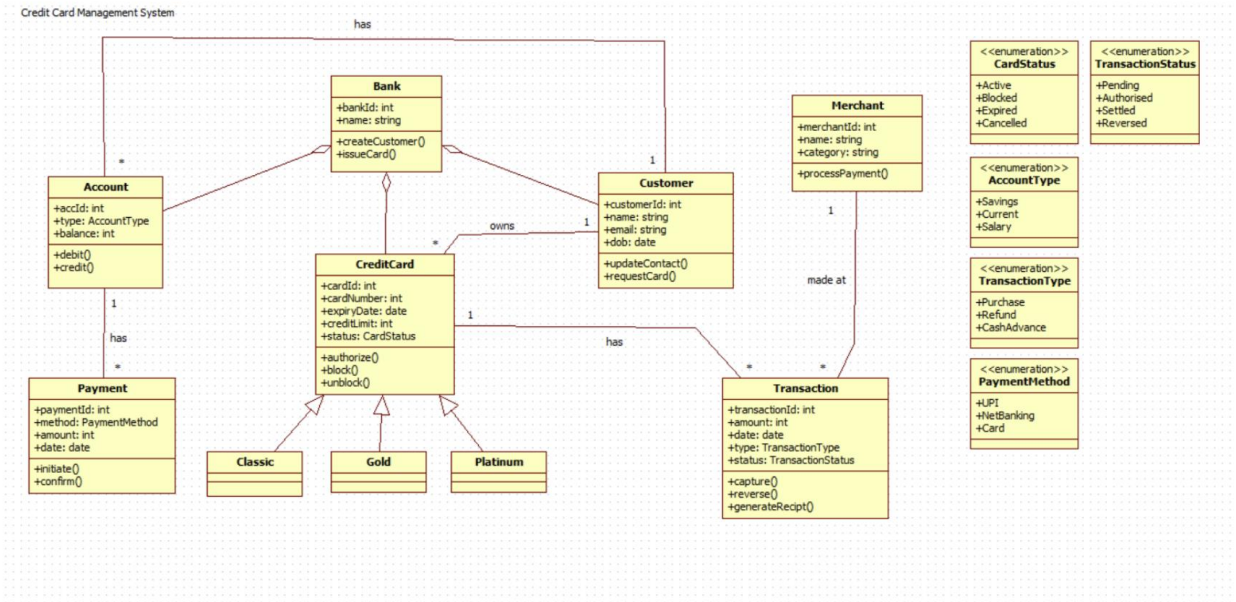

fig B1 - Advanced Class Diagram of Hotel Management System

Main Classes:

- Bank – Creates customers and issues cards.
- Customer – Holds account & credit card. Can request new cards and update contact.
- Account – Linked with credit card, used for transactions.
- CreditCard – Holds card number, expiry, credit limit and status.
- Transaction – Represents purchase/refund/cash advance with ID, amount, type and status.
- Merchant – Accepts payments and processes transactions.
- Payment – Stores payment-related details.
- Classic, Gold, Platinum – Specialized types of credit cards (inheritance).

Enumerations Used:

- CardStatus – Active, Blocked, Expired…
- TransactionStatus – Pending, Settled, Reversed…
- AccountType – Savings / Current / Salary
- TransactionType – Purchase, Refund, CashAdvance
- PaymentMethod – UPI, NetBanking, Card

Relationships:

- A Bank issues many CreditCards
- A Customer owns multiple CreditCards
- A CreditCard has many Transactions
- A Merchant processes Transactions
- Inheritance → CreditCard → Classic / Gold / Platinum
- Association → Payment is linked with Transaction

---

3. STATE DIAGRAM

The lifecycle of a credit card transaction is shown:

- Reading Card
- Network Check
- Security Check

- If successful → Reading PIN (retry allowed)
- Choosing Transaction (withdrawal/purchase/refund)
- Performing Transaction
- Validating account
- Deducting amount
- Printing receipt
- End State → Eject Card

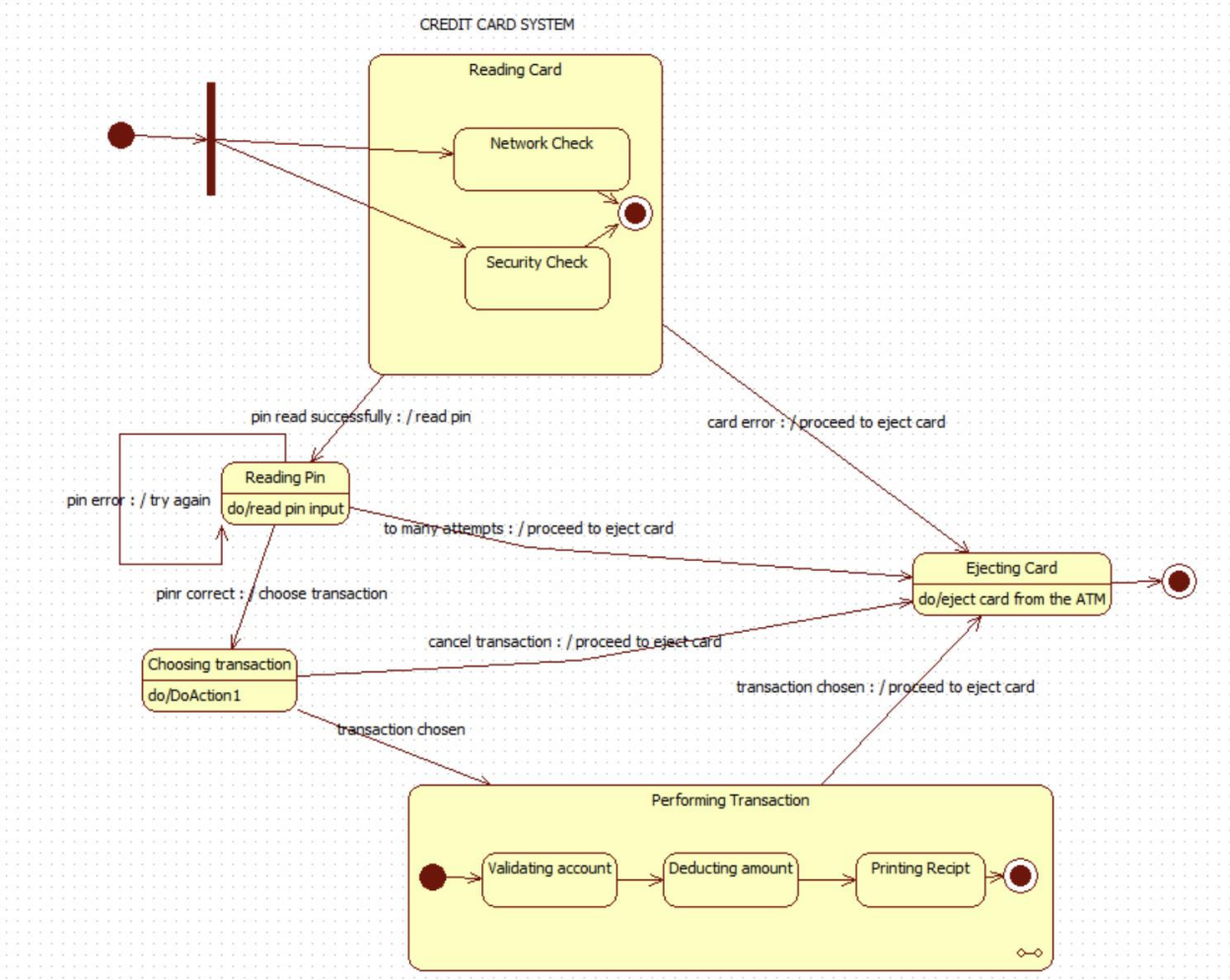

fig B2 - Advanced State Diagram of Hotel Management System

4. USE CASE DIAGRAM

Actors

Table 5: Actors and Roles

Actor	Role
Customer	Initiates payment by presenting card
Merchant	Requests transaction approval
Bank	Verifies card/account
Clearing House	Mediates & validates payments
Credit Card Provider	Updates available credit

Major Use Cases

- Present Credit Card
- Send Card & Merchant Information
- Verify Card Information
- Includes: Check Available Credit
- Includes: Verify Expiry Date
- Includes: Verify Card Number
- Send Confirmation
- Update Available Credit

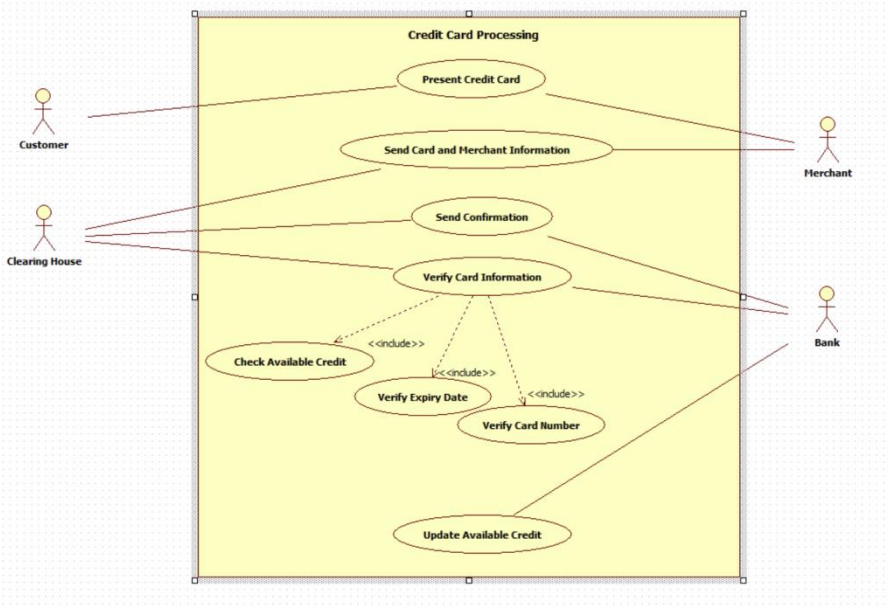

fig B3 - Advanced Use Case Diagram of Hotel Management System

5. SEQUENCE DIAGRAM

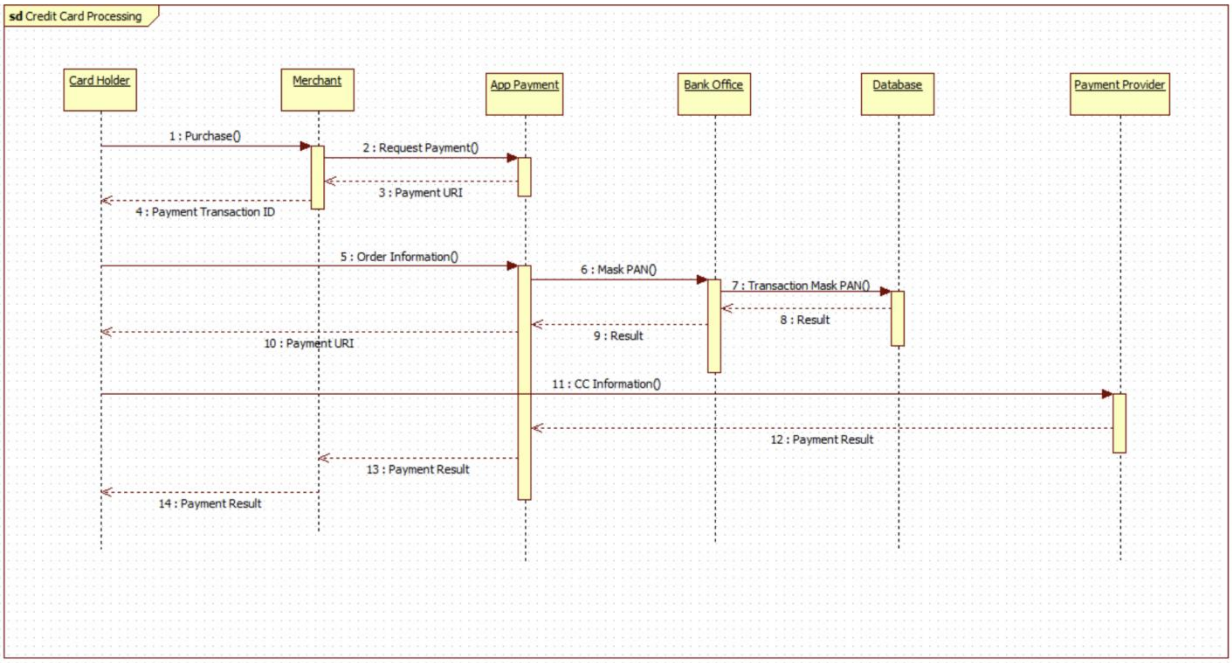

fig B4 - Advanced Sequence Diagram of Hotel Management System

Actors involved:

Card Holder, Merchant, App Payment System, Bank Office, Database, Payment Provider

Flow:

- Card holder initiates Purchase()
- Merchant Requests Payment()
- App Payment generates Payment URI
- Order information is sent forward
- PAN is masked (security)
- Bank office sends transaction request to database
- Database validates and returns result
- Credit card information is returned to app
- Final Payment Result is delivered to merchant and card holder

6. ACTIVITY DIAGRAM

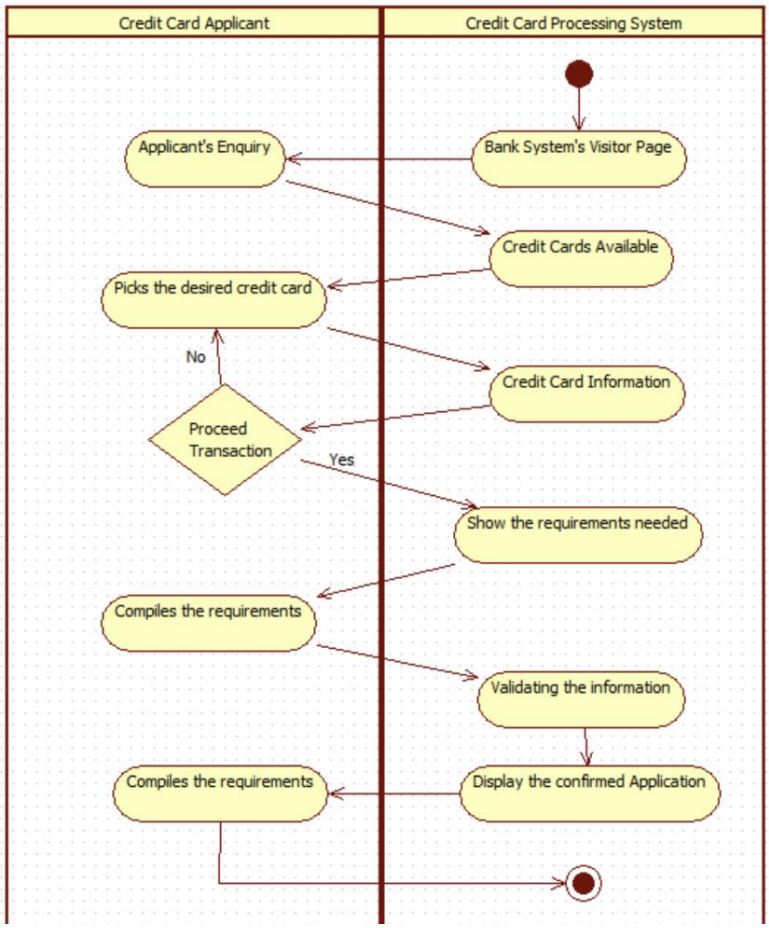

fig B5 - Advanced Sequence Diagram of Hotel Management System

Swimlanes

Table 6: Activity Swimlanes

Lane	Actions
Credit Card Applicant	Enquiry, select card, submit application
Processing System	Show options, validate info, confirm application

Flow Summary:

- Applicant enquires the system
- Bank shows card options
- Applicant selects a card

- System shows requirements
- Applicant submits documents
- System validates details
- If successful → display confirmed application
- End process

LIBRARY MANAGEMENT SYSTEM

1. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.1 Introduction

1.1.1 Purpose

The objective of the Library Management System is to simplify and computerize the process of managing books, members, loans, reservations, fines and staff duties inside the library. It reduces manual work and helps maintain accurate records of issued and returned books while also enforcing rules such as issuing limits and fines.

1.1.2 Scope

The system will:

- Store book details and maintain book availability status.
- Manage member registration and book issuing process.
- Maintain staff roles like librarian and assistants.
- Handle book reservations and loans.
- Calculate fines for late returns.
- Provide search functionality by book title, author, genre, or ISBN.

It is intended for small-to-medium libraries, colleges, schools and institutions.

1.1.3 Definitions & Acronyms

Table 1: Definitions and Acronyms

Term	Meaning
LMS	Library Management System
ISBN	International Standard Book Number
GUI	Graphical User Interface
Loan	Issued book to a member
Reservation	Request for a book in advance
Fine	Penalty for late return

1.2 General Description

1.2.1 Product Perspective

This system replaces traditional paper-based library registers with an automated software system. It stores and processes library transactions in a structured way and allows quick retrieval of information through database queries.

1.2.2 Product Functions

- Add and remove books from the library
- View available books
- Issue and return books
- Maintain reservations
- Calculate fines for late return
- Register new members
- Validate membership status
- Staff management and role assignment

1.2.3 User Characteristics

Table 2: User Characteristics

User	Role
Librarian	Controls issuing, member registration, fine calculation
Assistant	Arranges books, assists with issuing/returning
Member/User	Searches and requests books, borrows books

1.2.4 Assumptions & Constraints

- Only valid members can issue books.
- Every book must have a unique ID/ISBN.
- The system requires a database to store transactions.
- Maximum issue limit is predefined for each member.
- Book must be returned by due date; otherwise, fine is imposed.

1.3 Functional Requirements

FR-1: System shall allow users to search books by title, author, genre or ISBN.

FR-2: System shall allow librarian to add and remove books.

FR-3: System shall allow issuing and returning of books.

FR-4: System shall validate member before issuing a book.

FR-5: System shall maintain reservation requests.

FR-6: System shall calculate fine if return date exceeds due date.

FR-7: System shall generate reports of pending returns.

FR-8: System shall enable librarians to add/remove members.

FR-9: System shall update book status (available/reserved/issued).

1.4 Interface Requirements

1.4.1 User Interface

- Simple GUI with forms for books, members and loans.
- Buttons for search, issue, return, remove, update.
- Pop-up messages for:
- "Book not available"
- "Member not found"
- "Maximum issue limit reached"

1.4.2 Hardware Interface

- Computer terminal for librarian
- Barcode scanner/ID scanner (optional)
- Printer for issuing cards and reports (optional)

1.4.3 Software Interface

- Database (MySQL / PostgreSQL)
- Front-end GUI application
- OOP-based platform such as Java/C++/C#/.NET

1.5 Performance Requirements

- Book search must return results within 2–3 seconds
- Support minimum 20 concurrent users
- Handle minimum 50,000 book records efficiently
- Data retrieval and updates should be fast and reliable

1.6 Design Constraints

- Must use object-oriented design
- Each book must have a unique ID
- System should handle exceptions (e.g., wrong ID)
- Only librarian has admin privileges
- Future modules like e-books and online reservations must be easily addable

---

1.7 Non-Functional Attributes

Table 3: Non-Functional Attributes

Attribute	Requirement
Reliability	No data loss; backup must be possible
Security	Only authorized staff can make updates
Usability	Simple interface for quick learning
Maintainability	Easy to update and modify code
Scalability	Should handle a growing collection
Availability	System should work 24/7 in libraries

1.8 Schedule & Budget

Table 4: Schedule

Phase	Time
Requirement Study	1 week
UML Design	2 weeks
Database Setup	1 week
Development	4 weeks
Testing	1 week
Documentation	1 week

2. CLASS DIAGRAM

Classes

- **Library –**

Contains ID, name, address and operations for adding books and checking available books.

- **Book –**

Includes title, author, ISBN and status; can update book information.

- **Member –**

Represents users of the library and allows them to borrow and return books.

- **Reservation –**

Allows members to reserve books in advance.

- **Loan –**

Stores issue date, due date and return date; calculates fine.

- **Fine –**

Represents overdue penalties with amount and status.

- **Staff –**

Generic staff class with name, phone and role.

- **Librarian (inherits Staff) –**

Can add/remove members.

- **Assistant (inherits Staff) –**

Arranges books and assists users.

Relationships

- **Library — Book →**

One library has many books.

- **Member — Loan / Reservation →**

A member can have multiple loans or reservations.

- **Book — Reservation / Loan →**

A book may be reserved or issued.

- **Staff — Librarian / Assistant →**

Inheritance.

- **Member — Fine →**

A member may receive multiple fines.

This supports well-organized library operations using OOP concepts.

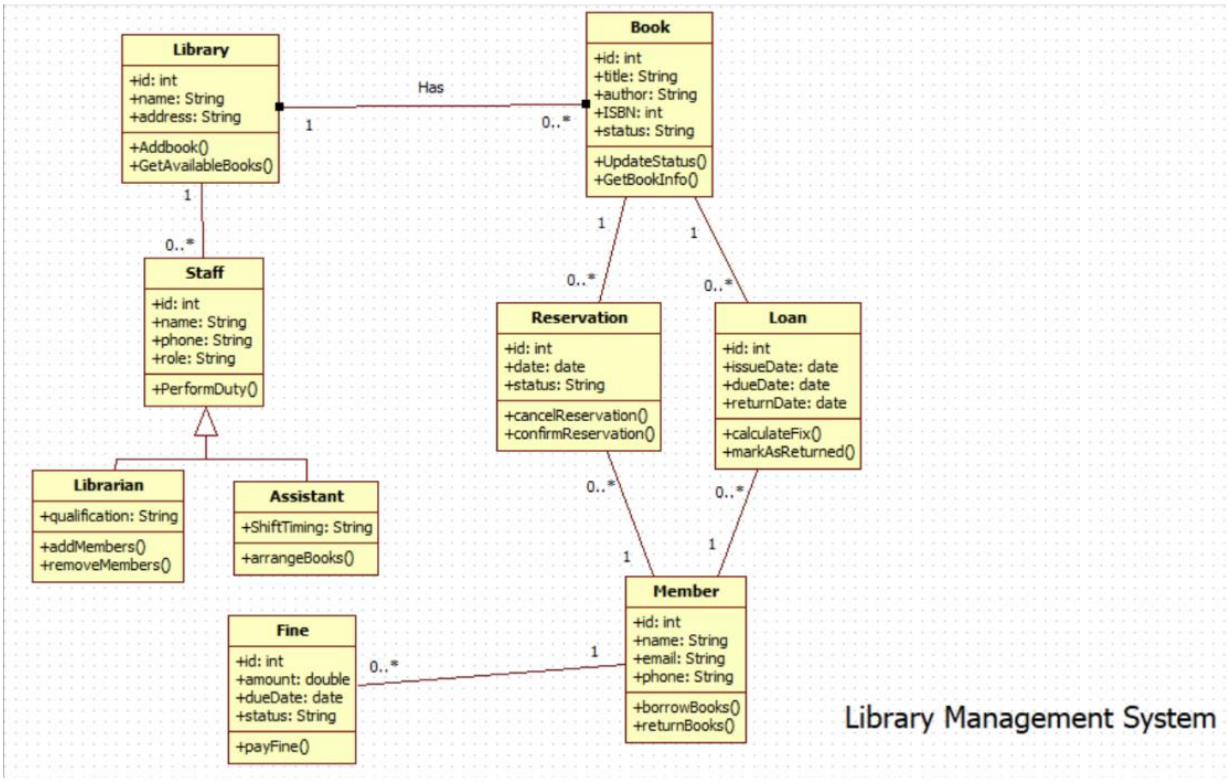

fig C1 - Advanced Class Diagram of Hotel Management System

---

3. STATE DIAGRAM

The state diagram shows the process of issuing a book:

- **Login –** Staff enters login details.
- If validated → transition to Input Book Details.
- System checks availability:
- If available → display status
- If not found → show error
- If request is valid → move to Issuing Book section.
- Two simultaneous updates occur:
- Update issue card

- Update library database
- System returns to login or logout.

This diagram clearly explains how a book request is processed step by step.

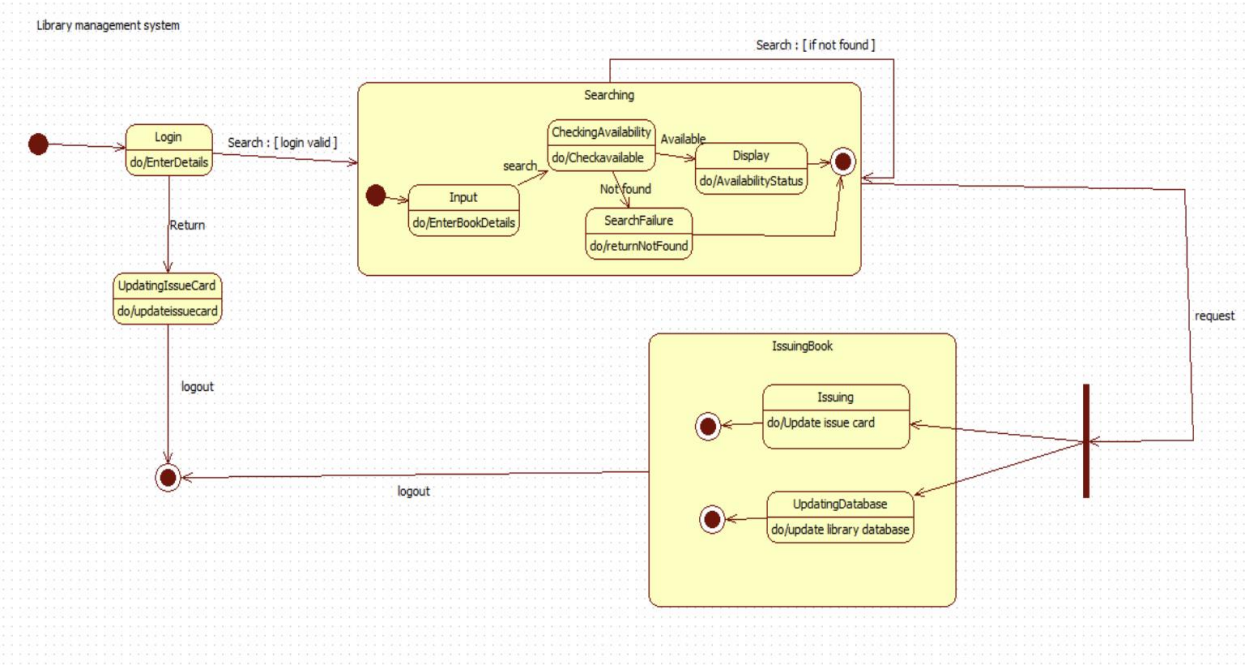

fig C2 - Advanced State Diagram of Hotel Management System

---

4. USE CASE DIAGRAM

Actors

Table 5: Actors and Roles

Actor	Role
Librarian	Full access – issue, return, add, remove, impose fines
User/Member	Search books, request or borrow books

Main Use Cases

- Search book (by author/name/genre)
- Validate member
- Request book
- Issue book

- Add/Delete book
- Return book
- Check issue limit
- Impose fine (extend use case)
- Pay fine
- View member record
- View pending returns
- View membership validity

The librarian mainly manages operations, while users can only search, request or borrow books.

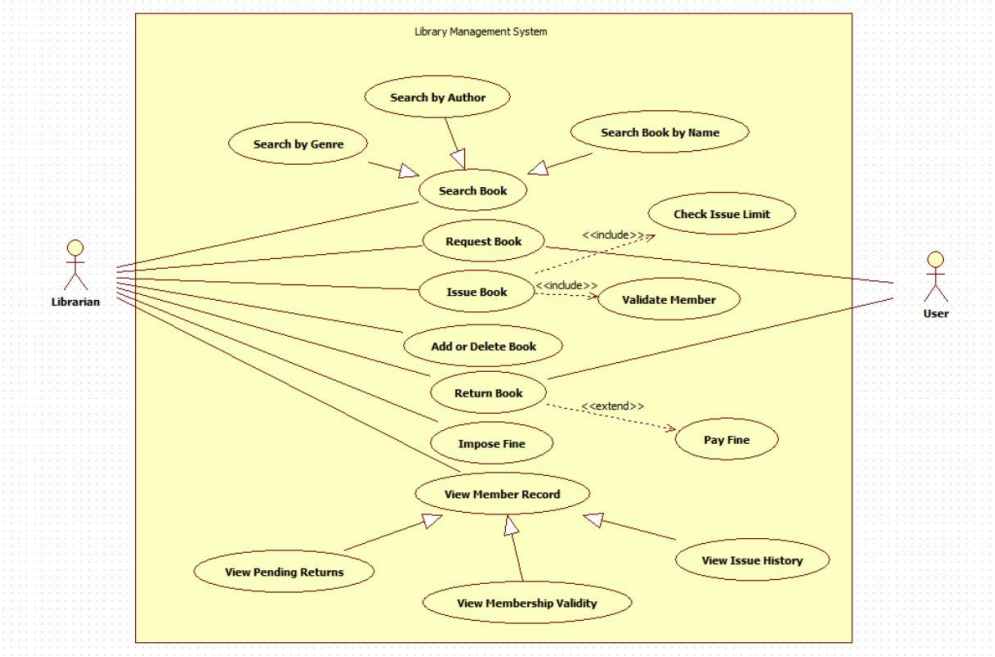

fig C3 - Advanced Use Case Diagram of Hotel Management System

---

5. SEQUENCE DIAGRAM

Objects involved:

Library Counter, Controller, Book Register, Book, Library Member

Workflow:

- Member shows library card.
- Controller checks registration.

- Book register validates member.
- Book register finds book and reserves it.
- Book status is updated.
- System displays book to member.
- Member can request renewal.
- System updates record again.

This shows the communication between UI, logic and database layers during issuing/reserving a book.

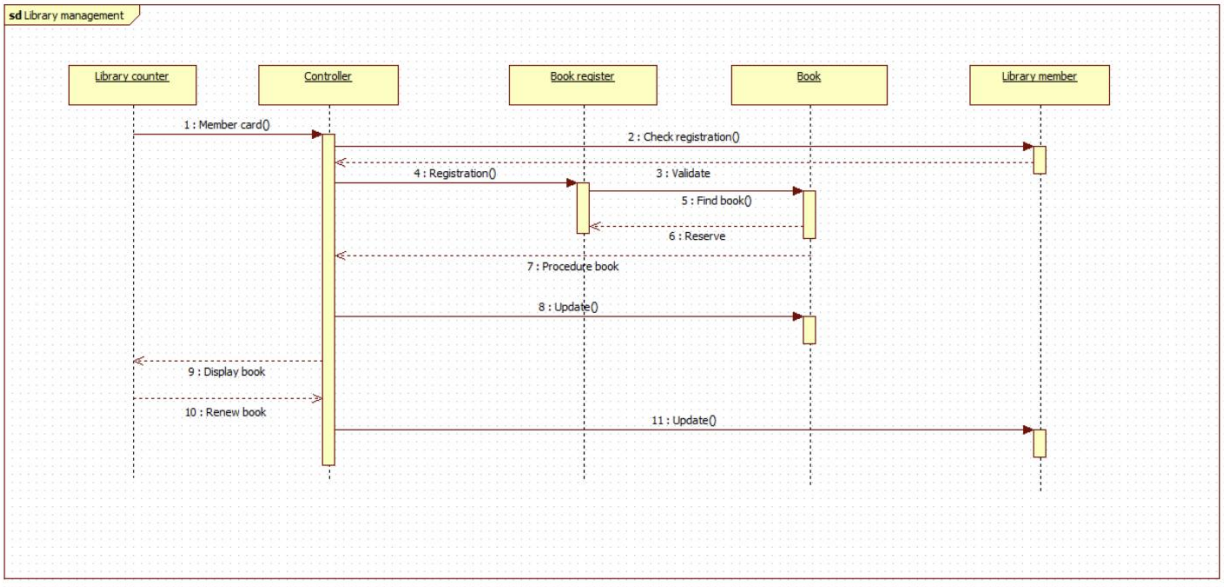

fig C4 - Advanced Sequence Diagram of Hotel Management System

---

6. ACTIVITY DIAGRAM – DESCRIPTION

The activity diagram shows full issuing process:

- Inquiry about books
- Check availability
- If not available → exit
- Validate member
- If not valid → register new member
- Check book issue limit
- If exceeded → book not issued

- If all valid → issue books
- Add issue details to member record
- Update book status
- End

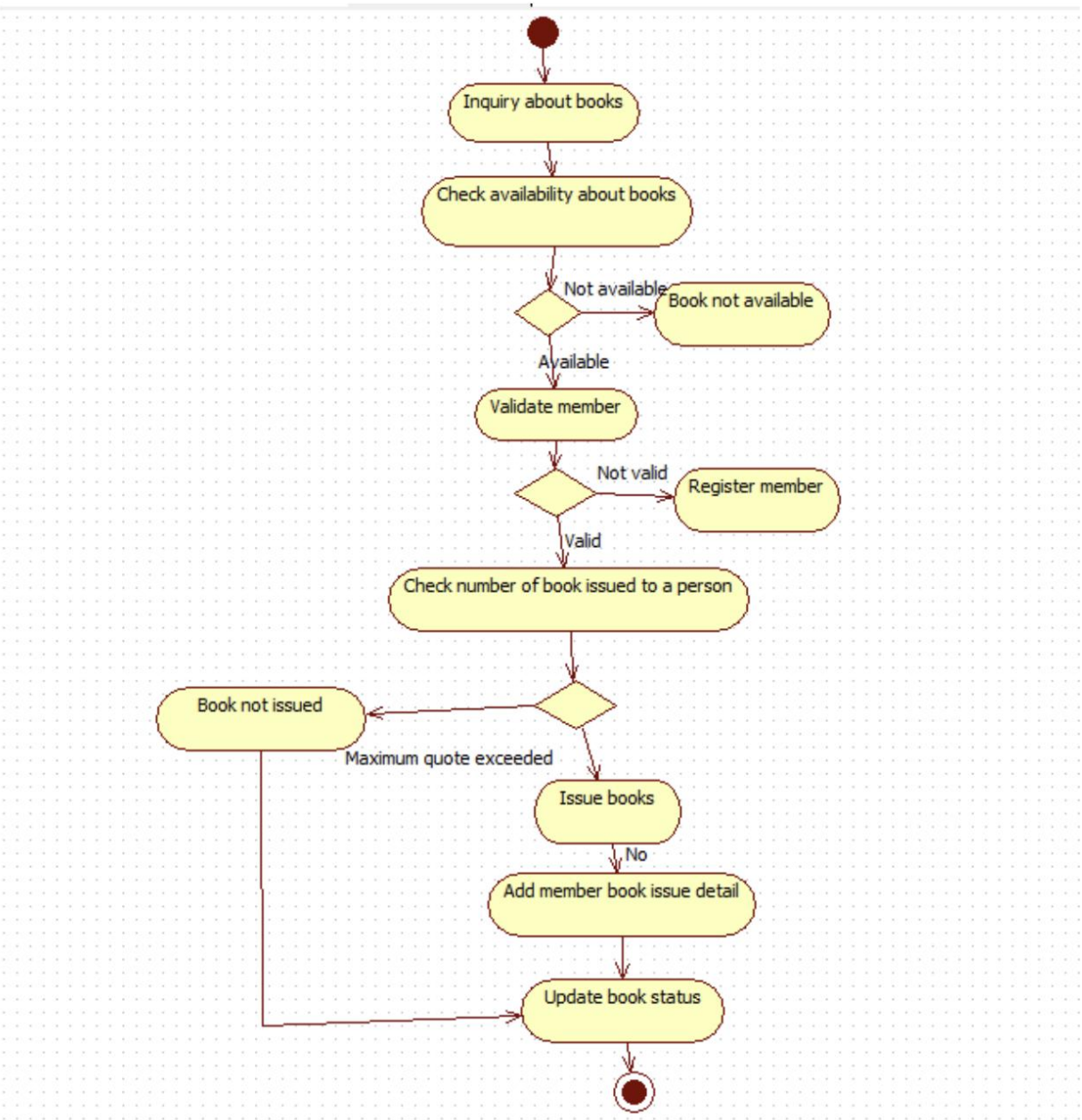

fig C5 - Advanced Activity Diagram of Hotel Management System

STOCK MAINTENANCE SYSTEM

1. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.1 Introduction

1.1.1 Purpose

The purpose of the Stock Maintenance System is to manage inventory, handle purchase orders, verify stock availability, track quantities, update transaction logs, and process payments. This system provides an organized way for admins, suppliers, and customers to manage stock-related activities without manual effort.

1.1.2 Scope

The system will:

- Allow admin users to maintain stock data.
- Enable users (consumers) to place purchase orders.
- Verify stock availability and update inventory.
- Create, change or delete purchase orders.
- Process payments and generate transaction logs.
- Display notifications on success or failure of any request.

It will primarily be used by small shops, warehouses, online stores or any organization that maintains stock.

1.1.3 Definitions & Acronyms

Table 1: Definitions and Acronyms

Term	Meaning
SMS	Stock Maintenance System
DBMS	Database Management System
PO	Purchase Order
Stock	Total available product quantity
Admin	Warehouse / Stock handler
Consumer	Customer who places order

1.2 General Description

1.2.1 Product Perspective

The system replaces manual stock registers with a digital, automated system. It includes login features, purchase order management, payment processing and stock update modules. It can later be extended to support multiple warehouses and suppliers.

1.2.2 Product Functions

- Login & authentication
- Add or update stock items
- Place purchase orders
- Change or delete existing orders
- Verify stock levels
- Handle supply from vendors
- Process payments
- Maintain transaction logs

1.2.3 User Characteristics

Table 2: User Characteristics

User	**Role**
Admin	Adds stock, manages transactions
Consumer	Searches products and places orders
Supplier	Delivers required stock items

1.2.4 Constraints

- Stock must be updated after every order
- Only admin can modify product quantity
- Consumers must be logged in to place orders
- Payment must be confirmed before final stock reduction
- Database must store complete purchase history

1.3 Functional Requirements

Table 3: Functional Requirements

No	Requirement
FR1	System shall allow user authentication before operations.
FR2	System shall allow admin to add, update stock items.
FR3	System shall allow consumer to place purchase orders.
FR4	System shall check stock availability before confirming order.
FR5	System shall process payment and confirm order.
FR6	System shall generate transaction history & logs.
FR7	System shall allow admin to delete/change purchase orders.
FR8	System shall display stock status to consumers on request.
FR9	System shall notify users on success or failure.

1.4 Interface Requirements

1.4.1 User Interface

- Simple GUI with forms for Login, Purchase Order, Stock Update, Payment.
- Error/Success messages shown clearly (e.g., “Stock Not Available”, “Payment Successful”).

1.4.2 Hardware Interface

- PC or mobile device with internet
- Optional barcode scanner

1.4.3 Software Interface

- Database (MySQL/PostgreSQL)
- Implementation language: Java / C# / Python / C++
- Requires login authentication module

1.5 Performance Requirements

- Up to 50 transactions per minute should be handled
- Response time <3 seconds for stock search
- Minimum 10 concurrent users should be supported
- Lightweight UI for smooth performance

1.6 Design Constraints

- Must follow OOP principles
- Each order and stock item must have a unique ID
- Should allow easy future addition of suppliers
- Only admin should modify product quantity

1.7 Non-Functional Attributes

Table 4: Non-Functional Attributes

Attribute	Description
Reliability	Must not lose transaction data
Usability	Easy interface with clear options
Security	Restricted access to admin features
Availability	Should work 24×7
Maintainability	Well-structured code for updates
Scalability	Should support more suppliers later

1.8 Schedule & Budget

Table 5: Schedule

Phase	Duration
Requirement Analysis	1 week
UML Modelling	2 weeks
Design & Database Setup	2 weeks
Implementation	4 weeks
Testing & Documentation	2 weeks

2. CLASS DIAGRAM

Main classes in the system:

Classes and Roles

Table 6: Class Roles

Class	Role
SignIn	Authenticates user login
Admin	Maintains stock and product details
Consumer	Places purchase orders
PurchaseOrder	Creates, changes, deletes orders
StockMaintain	Holds product quantity and price
Payment	Handles order payments

Inheritance

- PurchaseOrder has specialized operations: CreatePurchaseOrder, ChangePurchaseOrder, Deletion.

Associations

- SignIn → Admin (validation of access)
- Consumer → PurchaseOrder (places order)
- PurchaseOrder → StockMaintain (modifies stock)
- PurchaseOrder → Payment (payment required)

This structure ensures OOP-based modular design, making updates easier.

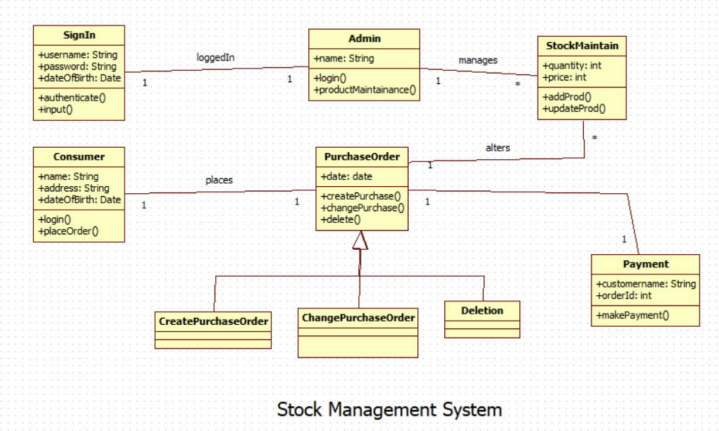

fig D1 - Advanced State Diagram of Hotel Management System

3. STATE DIAGRAM

This state diagram describes how an order request is handled:

- Request Item
- Move to Check Availability
- Query database
- Verify stock level
- If Not Available → Reject request
- If Available → Inventory Update
- Deduct quantity
- Update record
- Transaction Log
- Add log entry
- Save changes
- System returns Success or Request Rejected

This diagram shows a structured decision-based process of stock handling.

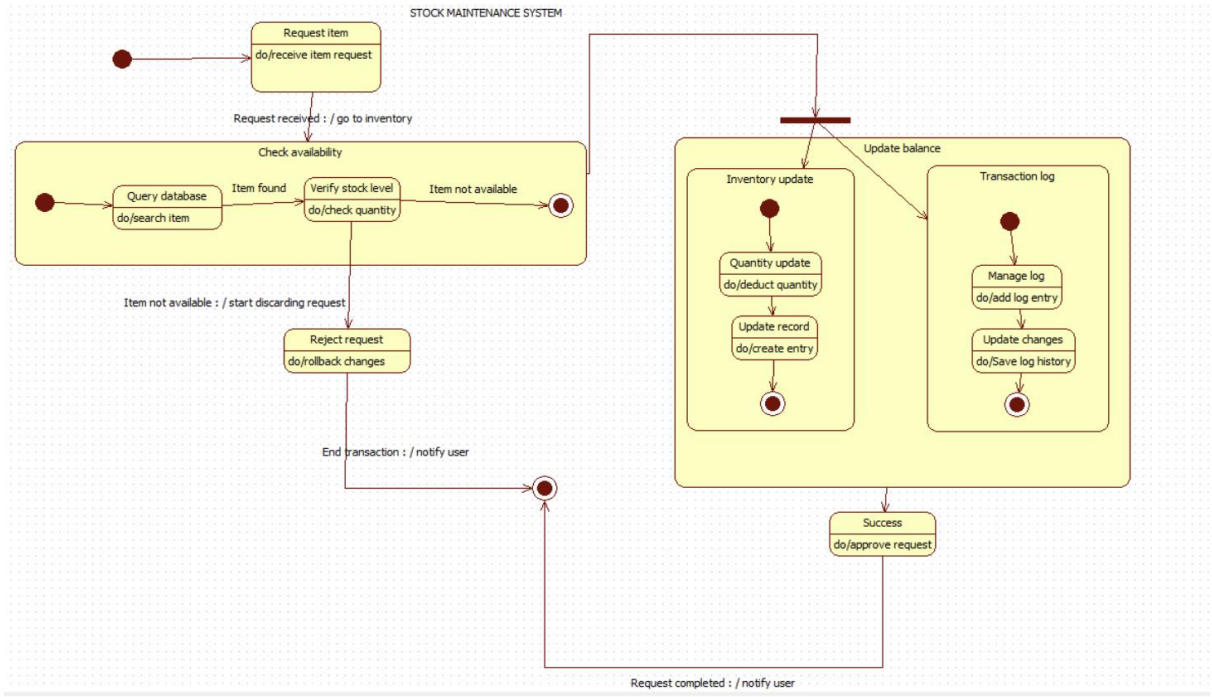

fig D2 - Advanced Class Diagram of Hotel Management System

4. USE CASE DIAGRAM

Actors

Table 7: Actors and Roles

Actor	Role
Customer	Purchase & order items
Manager/Admin	Manage stock, reports & database
Supplier	Supply ordered stock

Major Use Cases

- Authentication (Login/Logout)
- Purchase (includes: Place Order & Make Payment)
- Check Stock (Warehouse Stock / Store Stock)
- Update Database
- Order Supply
- Supply to Customer
- Generate Report

The use case diagram clearly divides responsibilities among the three actors.

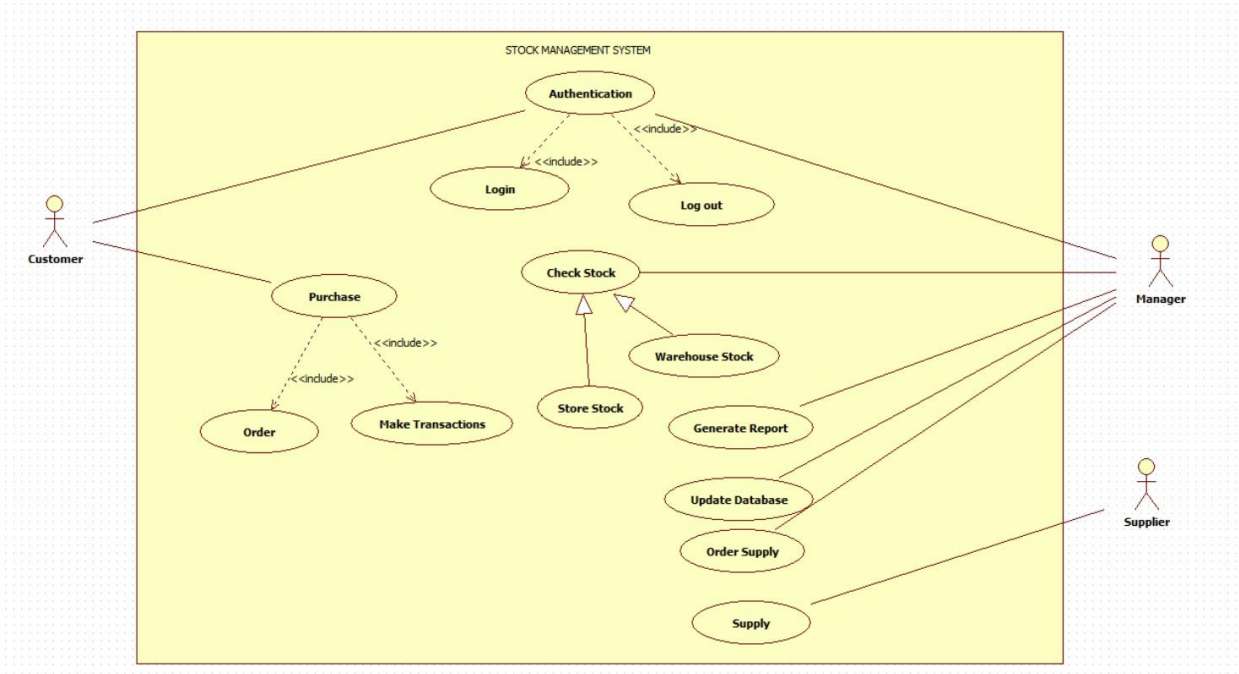

fig D3 - Advanced Use Case Diagram of Hotel Management System

5. SEQUENCE DIAGRAM

Objects involved:

Customer → Login → Authentication → Search Product → Order → Supplier → Stock

Flow:

- Customer enters credentials
- Login sends validation request
- Authentication approves or rejects
- Once logged in – search product
- If available → order placed
- Supplier receives order
- Stock checked and updated
- Payment processed
- If product quality issue → refund procedure
- Return stock (inventory updated)

This explains interaction between system components in real-time.

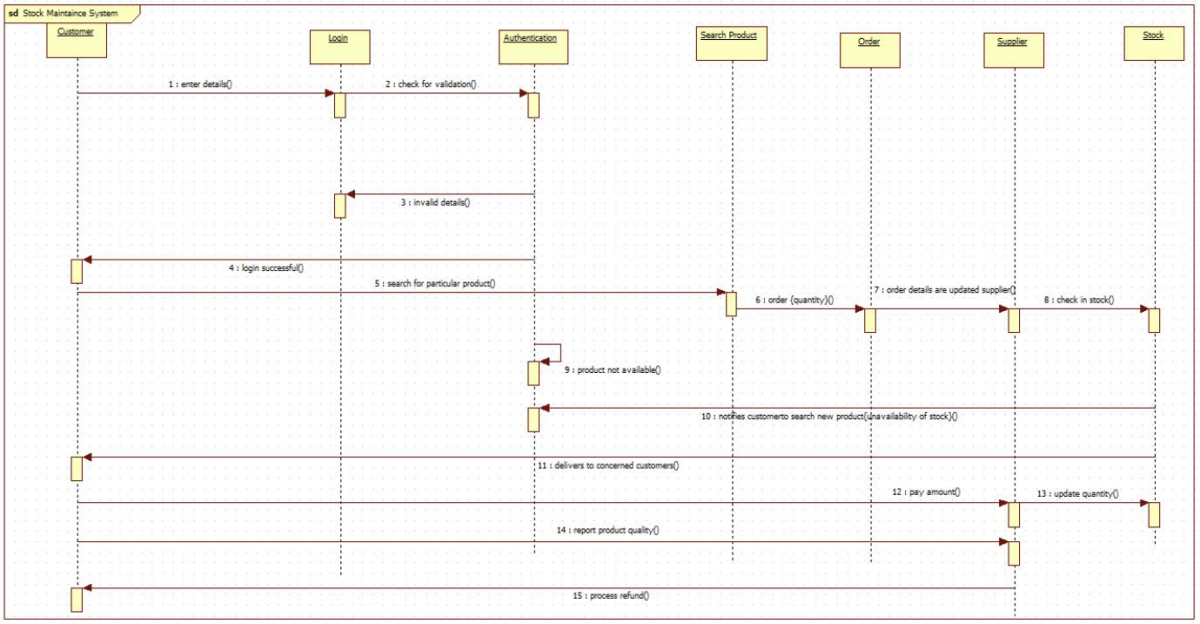

fig D4 - Advanced Sequence Diagram of Hotel Management System

6. ACTIVITY DIAGRAM

Steps shown:

- Login → Authentication
- Search for product
- Check availability
- If failure → notify & go back
- Place order
- Supplier checks stock
- If stock available → deliver item
- Customer makes payment
- If payment fails → return product
- If success → stock quantity updated in system
- End process

This gives clear understanding of full transaction cycle from login to final update.

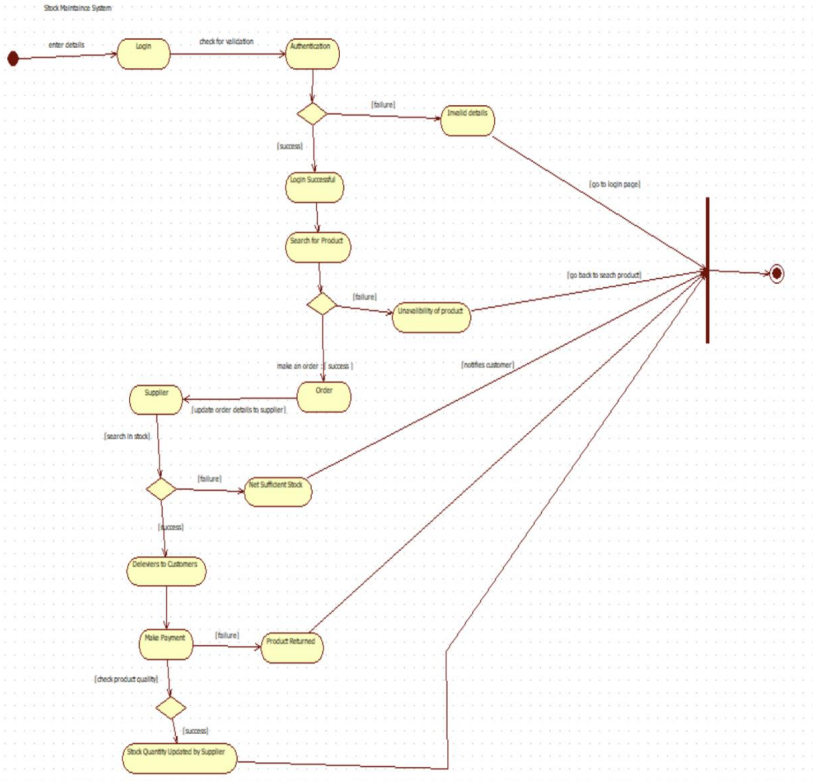

fig D5 - Advanced Activity Diagram of Hotel Management System

PASSPORT AUTOMATION SYSTEM

1. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.1 Introduction

1.1.1 Purpose

The purpose of the Passport Automation System is to simplify and automate the process of applying for, verifying, and issuing passports. The system allows applicants to submit details and documents online, make payments, track application status, and receive notifications. It also supports government officials in verification and approval processes.

1.1.2 Scope

The system includes:

- Submitting a new passport application
- Uploading documents and biometric data
- Scheduling appointments
- Making payments
- Police verification
- Final approval and passport dispatch

The system will be used by the Applicant, Verification Officer, and Passport Office.

1.1.3 Definitions

Table 1: Definitions

Term	Meaning
PAS	Passport Automation System
ID	Identification Number
Biometrics	Fingerprint & photograph
Dispatch	Sending passport to applicant

1.2 General Description

1.2.1 Product Perspective

The system replaces the traditional manual passport process with an online platform. It integrates document submission, biometrics collection, police verification and payment modules.

1.2.2 Product Functions

- User registration and login
- Submit passport application
- Upload documents and biometrics
- Book appointment
- Make payment
- Police verification
- Issue or reject passport
- Track status of application

1.2.3 User Characteristics

Table 2: User Characteristics

User	Description
Applicant	Applies for new/renewal passport
Verification Officer	Reviews documents and biometrics
Passport Office	Issues passport and manages overall process

1.2.4 Constraints

- All required documents must be valid and uploaded
- Payment must be completed before further processing
- Police verification must be passed before issuing passport
- Only authorized officials can access sensitive data

1.3 Functional Requirements

Table 3: Functional Requirements

No	Requirement
FR1	System shall allow applicant registration and login.
FR2	System shall accept new passport applications.
FR3	System shall allow uploading of required documents.
FR4	System shall verify biometrics and user details.
FR5	System shall allow payment and issue receipts.
FR6	System shall initiate police verification.
FR7	System shall allow officials to approve/reject application.
FR8	System shall update user on status of application.
FR9	System shall issue passport when approved.
FR10	System shall allow tracking of dispatched passport.

1.4 Interface Requirements

1.4.1 User Interface

- Online form for application
- Upload section for documents and photos
- Payment gateway interface
- Status tracking screen

1.4.2 Software Interface

- Database (SQL/MySQL)
- Payment API
- Biometric verification system

1.4.3 Hardware Interface

- Computer / Mobile with internet
- Scanner or biometric device

1.5 Performance Requirements

- Response time <3 seconds for any user action
- Must support up to 1000 users at a time
- Safety and privacy of data is mandatory
- Backup and recovery must be active

1.6 Design Constraints

- Must follow government security policies
- Each user must have unique ID and password
- Only verified documents will be accepted
- Biometric data must be securely stored

1.7 Non-Functional Attributes

Table 4: Non-Functional Attributes

Attribute	Description
Security	Strong encryption for personal data
Reliability	Should not lose any application data
Usability	Simple and clear interface
Availability	24×7 online support
Maintainability	Database should support easy updates
Scalability	Should work for nationwide usage

1.8 Schedule & Budget (Assumed)

Table 5: Schedule

Phase	Duration
Requirement Analysis	2 weeks

UML Modelling	2 weeks
Development	6 weeks
Testing & Deployment	3 weeks
Documentation	1 week

---

2. CLASS DIAGRAM

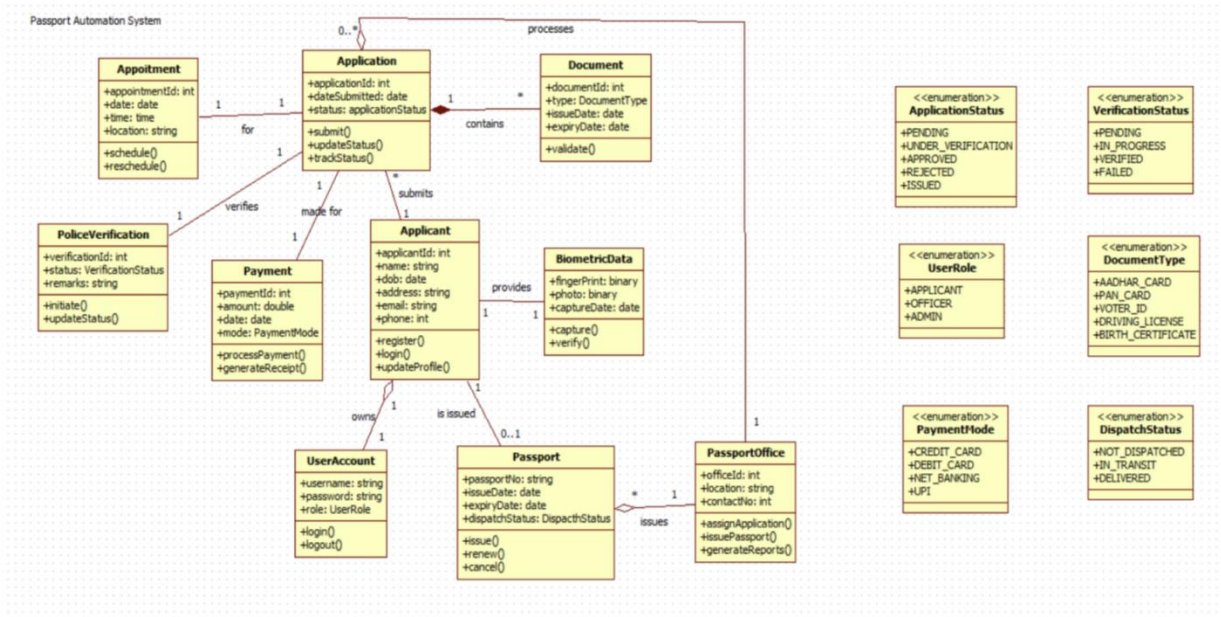

fig E1 - Advanced Class Diagram of Hotel Management System

Main Classes

Table 6: Class Descriptions

Class	Description
Applicant	Registers, submits application
Application	Holds application details/status
Appointment	Schedules biometric visit
Document	Stores uploaded documents
Biometrics	Stores fingerprint/photo
Payment	Handles fees and receipts
PoliceVerification	Verifies applicant
Passport	Final passport issued
PassportOffice	Manages issue & approval
UserAccount	Login and role management

Key Relationships

- Applicant submits Application
- Application contains Document
- Applicant provides BiometricData
- Payment must be completed before verification
- PassportOffice issues passport
- PoliceVerification verifies identity
- UserAccount is owned by Applicant

The design follows proper object-oriented structure for all major tasks.

---

3. STATE DIAGRAM

Main states involved in the process:

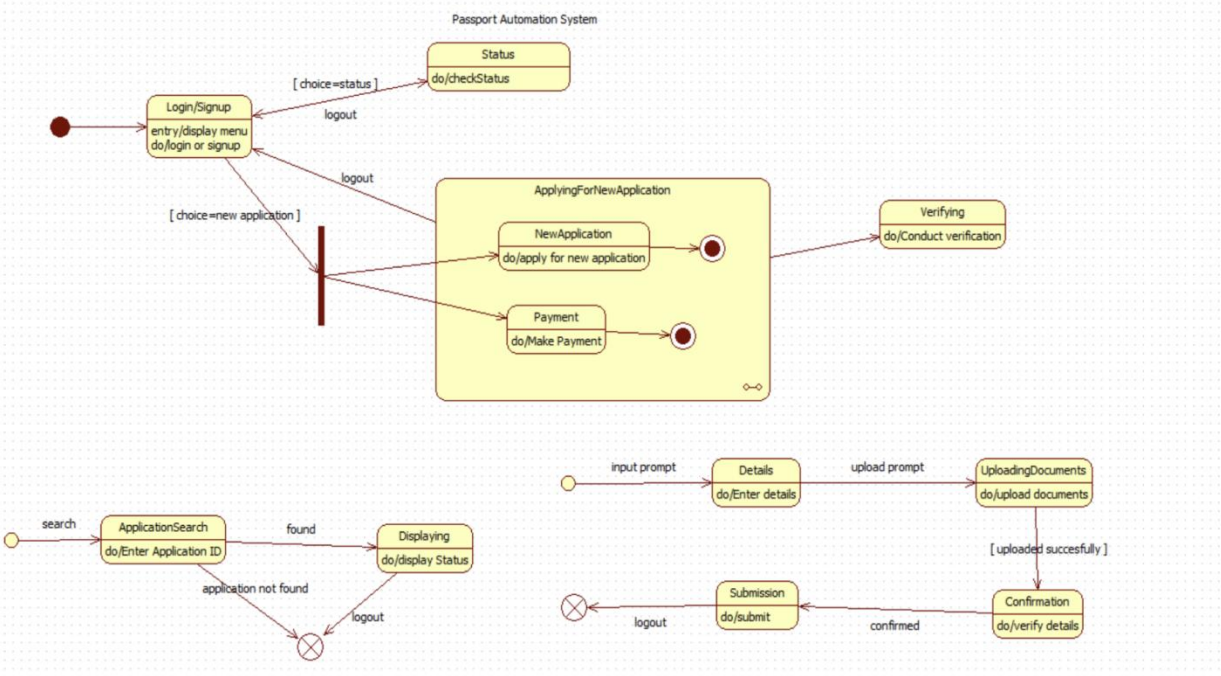

fig E2 - Advanced State Diagram of Hotel Management System

- Login / Signup
- Apply for new application
- Enter details

- Upload documents
- Make payment
- Verification (documents + police + biometrics)
- Submission
- Status check
- Confirmation or Rejection

This state diagram shows complete lifecycle of a passport application from initial login to final decision.

---

3. USE CASE DIAGRAM

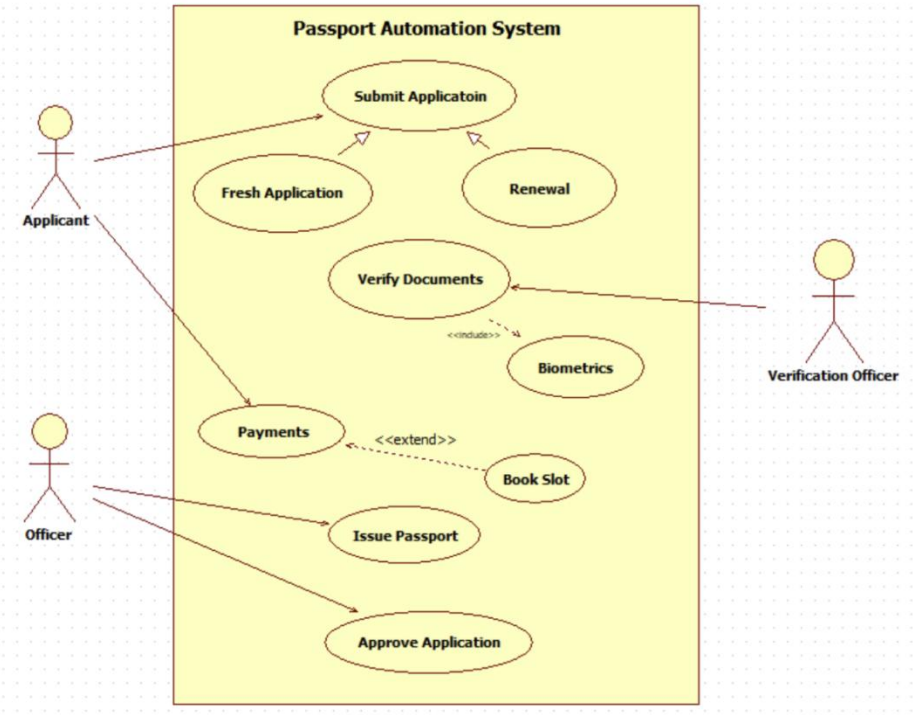

fig E3 - Advanced Use Case Diagram of Hotel Management System

Actors

Table 7: Actors and Roles

Actor	Role
Applicant	Applies for/Renews passport

Officer	Verifies documents & approves
Verification Officer	Conducts biometric & police checks

Major Use Cases

- Submit Application
- Fresh Application / Renewal
- Verify Documents
- Biometrics
- Book Slot
- Make Payment
- Issue Passport
- Approve / Reject Application

Use cases are clearly divided based on user type and responsibility, ensuring proper access control.

---

5. SEQUENCE DIAGRAM

Objects involved:

Applicant → Passport Administrator → Regional Administrator → Police → Database

Order of operations:

- Login
- Submit personal details
- Store details in database
- Admin verifies
- Forwarded to regional administrator
- Police conducts verification
- Database updated
- Final approval
- Passport issued

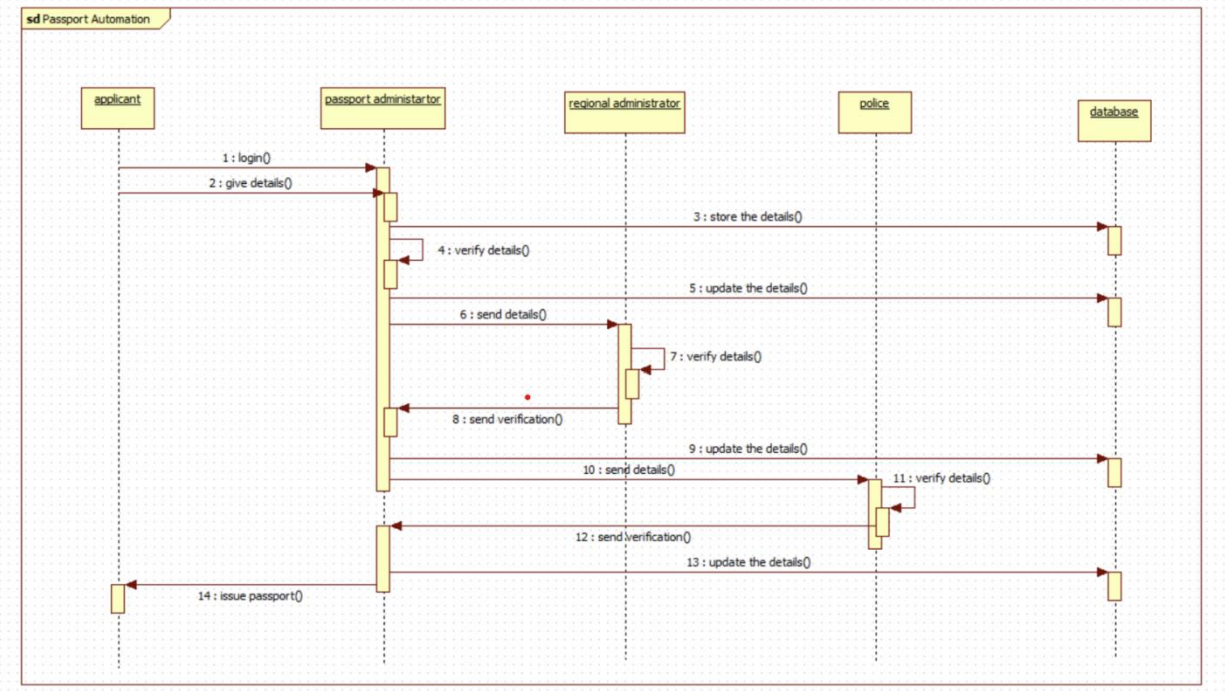

fig E4 - Advanced Sequence Diagram of Hotel Management System

---

4. ACTIVITY DIAGRAM

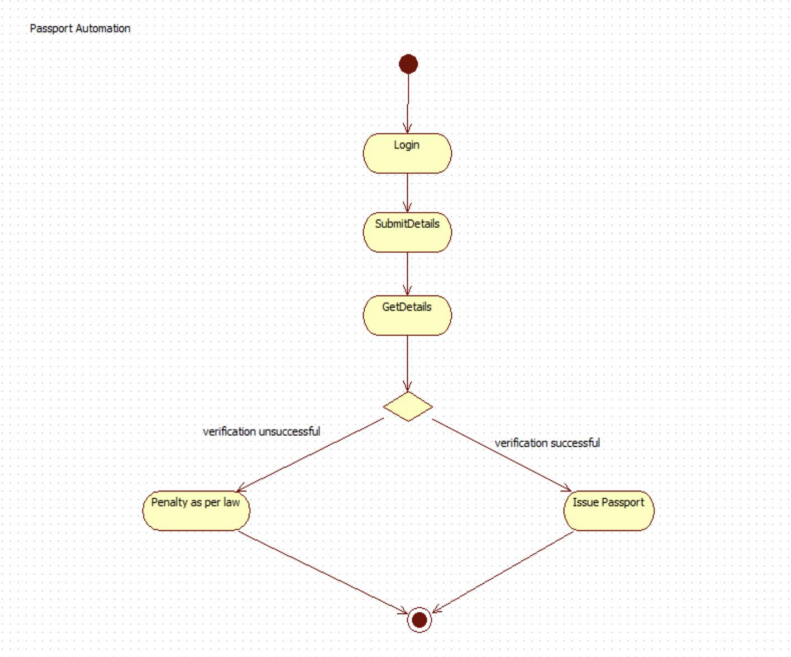

fig E5 - Advanced Activity Diagram of Hotel Management System

Process steps:

- Applicant logs in
- Submits details
- System retrieves and checks them
- If verification is successful → passport issued
- If verification fails → penalty or rejection
- Process ends

This shows the decision-based flow followed by government officers.
