**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:*

**BHOOMIKA M**

**1BM23CS068**

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 **BHOOMIKA M(1BM23CS068)** during the 5th Semester August 2025-December 2025.

Signature of the Faculty Incharge:

[Vikranth B.M](https://bmsce.ac.in/home/facultyProfile/52/Vikranth-BM)

Assistant Professor

Department of Computer Science and Engineering

B.M.S. College of Engineering, Bangalore

Table of Contents

|  |
| --- |
| 1. Hotel Management System |
| 2. Credit Card Processing |
| 3. Library Management System |
| 4. Stock Maintenance System |
| 5. Passport Automation System |

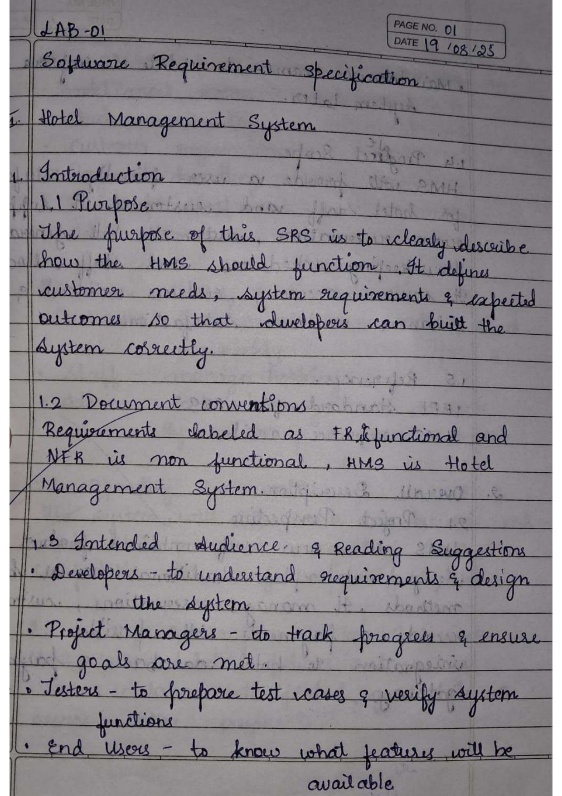
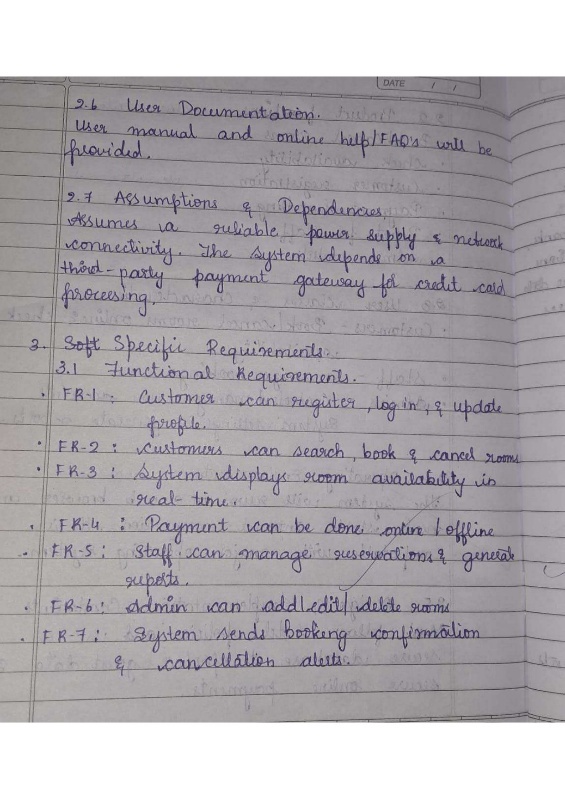
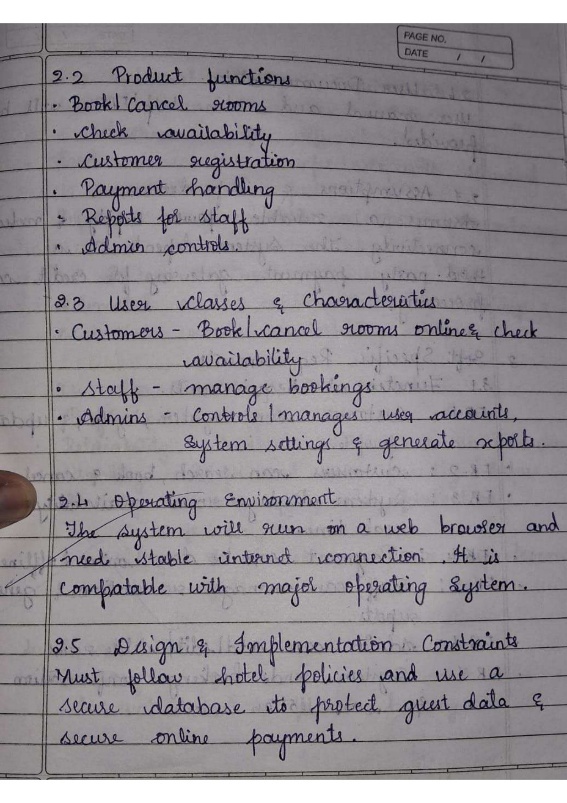
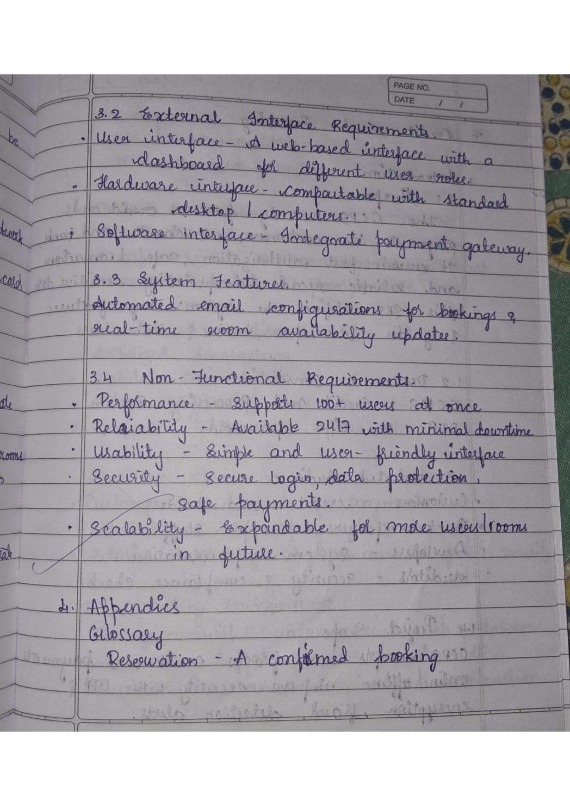
1. **Hotel Management System**

**Problem statement :**

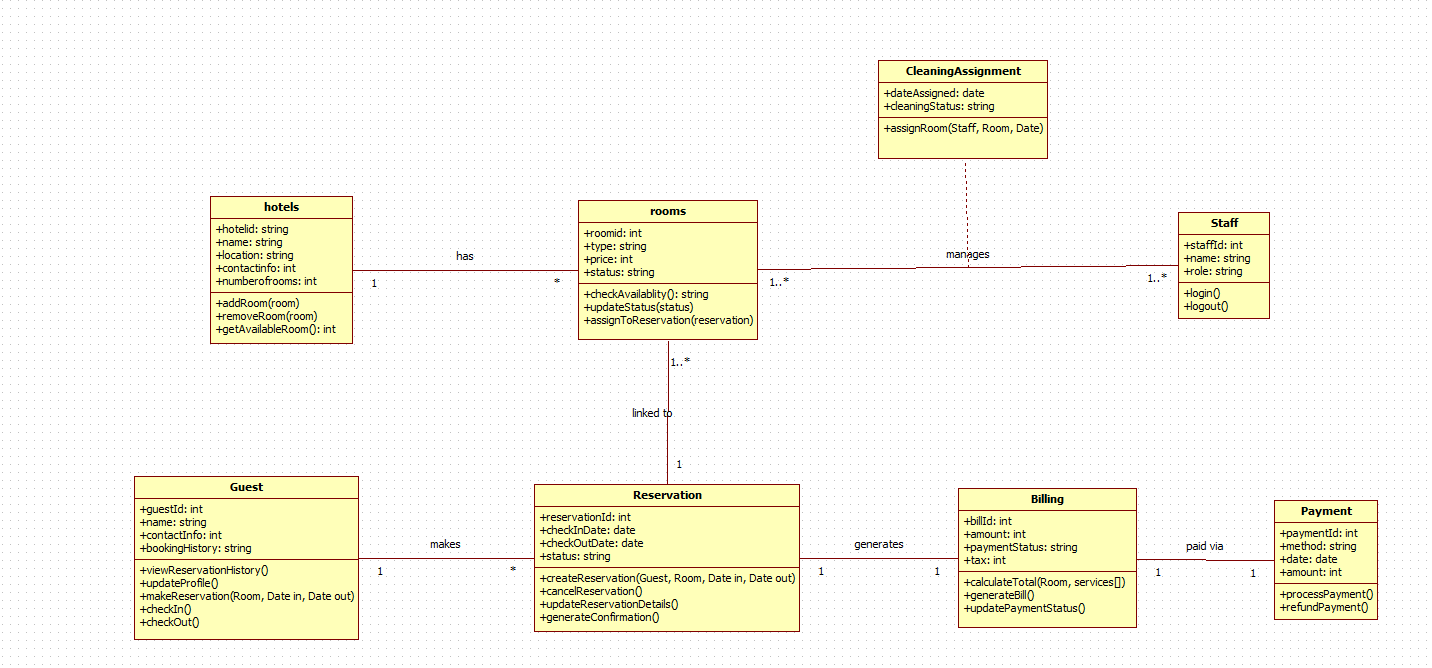
A hotel needs a computerized system to streamline its daily operations, improve customer experience, and reduce manual workload. Currently, most tasks such as room booking, check-in/check-out, staff allocation, and bill generation are handled manually, which leads to errors, delays, and difficulty in maintaining records.

The Hotel Management System (HMS) should automate all major hotel activities and maintain accurate data in a structured way. The system must allow receptionists, managers, and customers to efficiently perform their respective tasks.

**SRS-Software Requirements Specification**

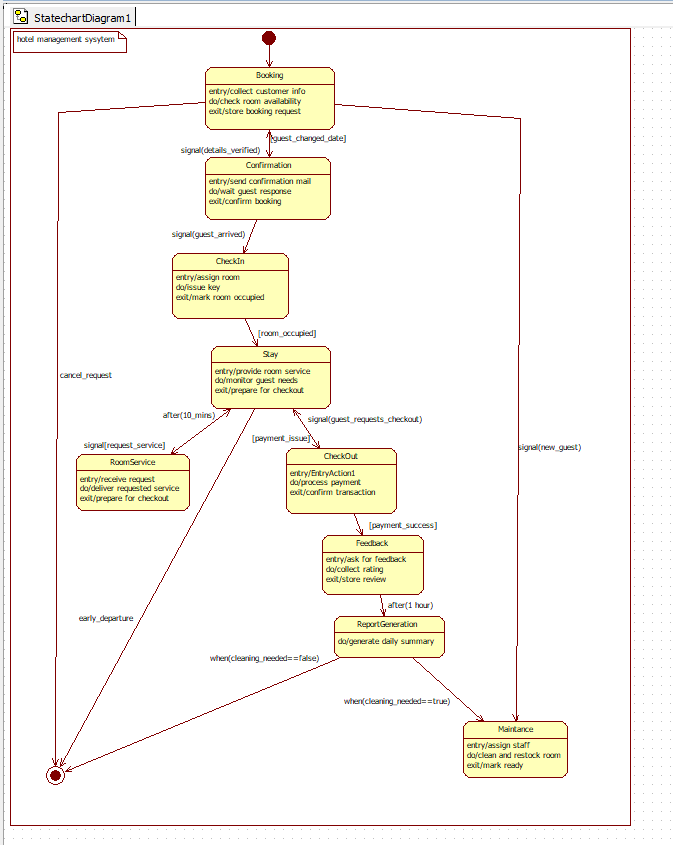
  

**Class Diagram**



|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Purpose / What the Class Does** | **Relationship With Other Classes** | **Description of Relationship** |
| Hotel | Stores hotel details like name, location, and total rooms. Manages adding, removing, and listing available rooms. | 1 → \* Rooms | One hotel has many rooms. The hotel acts as the container for all rooms in the system. |
| Room | Represents each room with details like type, price, availability, and assignment. Handles booking, status updates, and availability checks. | \* → 1 Hotel \* → 1 Reservation | Each room belongs to one hotel. A room can be linked to multiple reservations over time, but only one active reservation at a time. |
| Guest | Stores guest information, contact details, and booking history. Responsible for making reservations and performing check-in/check-out. | 1 → \* Reservation | A guest can make many reservations, but each reservation is linked to only one guest. |
| Reservation | Handles the full reservation process including booking dates, status, and confirmation. Represents room booking by a guest. | \* → 1 Guest \* → 1 Room 1 → 1 Billing | Reservation is the central link: one guest + one room = one reservation. Each reservation generates exactly one bill. |
| Billing | Generates bills for reservations including room charges, services, and taxes. Updates payment status. | 1 → 1 Reservation 1 → \* Payment | One reservation produces one bill. A bill can have multiple payments (partial payments allowed). |
| Payment | Handles payment methods (cash, card, UPI) and transaction amounts. Processes and refunds payments. | \* → 1 Billing | Payments belong to a bill. Multiple payments can be made toward one bill. |
| Staff | Represents hotel employees, including receptionists and cleaning staff. Handles login and assigned duties. | 1 → \* CleaningAssignment | One staff member can have many cleaning assignments. Staff manages room status indirectly through cleaning tasks. |
| CleaningAssignment | Stores cleaning details like assigned date, status, and staff-room mapping. Handles room cleaning tasks. | \* → 1 Staff \* → 1 Room | A cleaning assignment links one staff member to one room at a time. Over time, many assignments occur for each room and staff member. |

**State Diagram:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description (What Happens)** | **Trigger / Event** | **Next State** | **Explanation** |
| **Booking** | Collects customer info, checks availability, stores booking request. | details\_verified | Confirmation | User begins the process by booking a room. |
| **Confirmation** | Sends confirmation mail, waits for response, finalizes booking. | guest\_arrived | CheckIn | Confirms booking and prepares room allocation. |
| **CheckIn** | Assigns room, issues key, marks room as occupied. | room\_occupied | Stay | Guest arrives and check-in is completed. |
| **Stay** | Guest stays in hotel; room service and monitoring happen. | guest\_requests\_checkout | CheckOut | Active stay phase. Guest can request service or check out. |
| **RoomService** | Staff provides requested services. | request\_service | Stay | Temporary state for service, then returns to Stay. |
| **CheckOut** | Processes payment and closes stay. | payment\_success | Feedback | Guest initiates checkout. |
| **Feedback** | Collects guest feedback and rating. | after(1 hour) | ReportGeneration | Optional step after checkout. |
| **ReportGeneration** | Generates daily summary reports. | cleaning\_needed == true | Maintenance | Final reporting phase. |
| **Maintenance** | Cleans and prepares room for next guest. | cleaning\_needed == false | Booking | Final reset before next guest cycle. |

**Use-Case Diagram:**

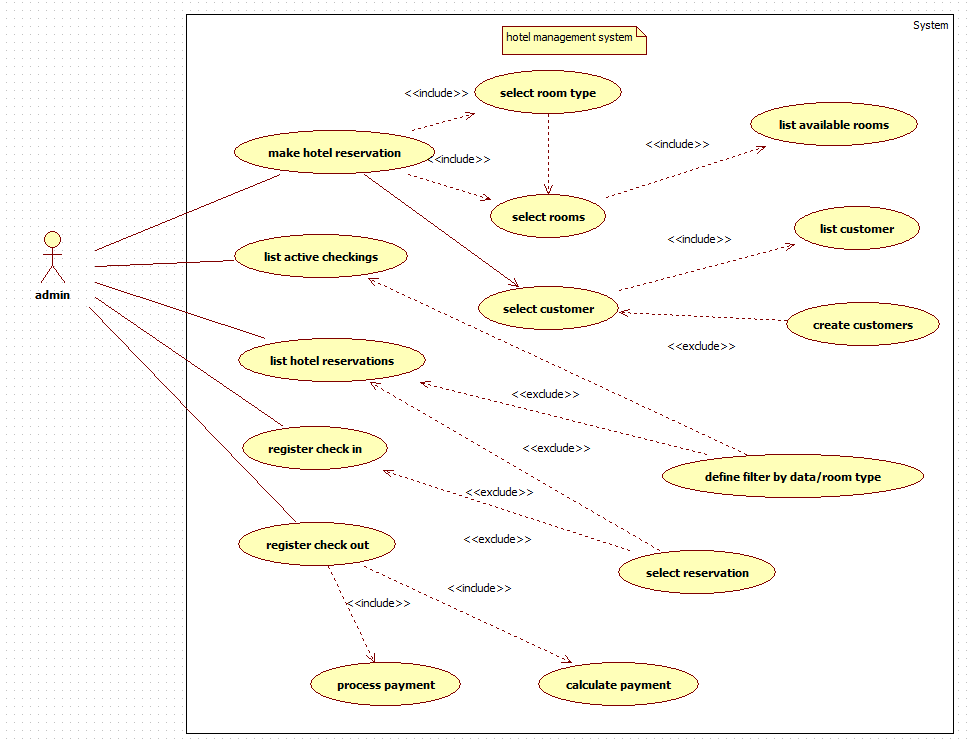


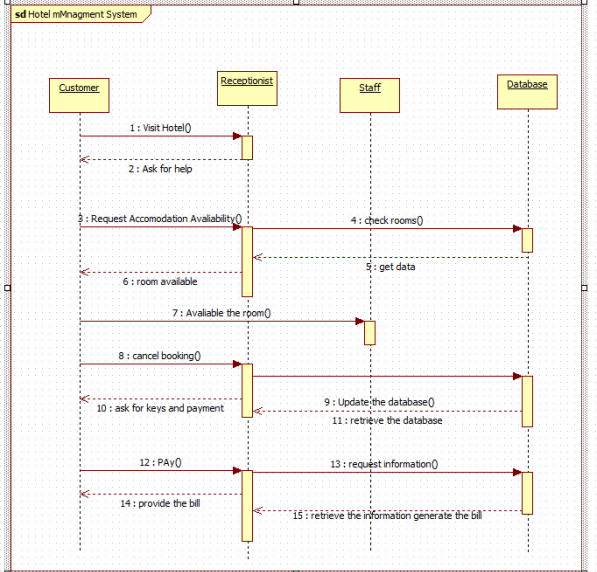
TABLE 1: Actors and Their Responsibilities

|  |  |
| --- | --- |
| **Actor** | **Description / Responsibilities** |
| Admin | The main user of the system. Responsible for managing reservations, customer records, check-ins, check-outs, viewing room availability, processing payments, and handling all booking-related operations. |

TABLE 2: Use Cases, Purpose, and Relationships

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **What It Does** | **Actor Involved** | **Relationship (Include / Extend)** | **Explanation of Relationship** |
| Make hotel reservation | Creates a new reservation for a customer | Admin | Includes → Select room type, Select rooms, Select customer | Reservation requires selecting room type → available rooms → customer. |
| Select room type | Filters rooms based on type (AC, Deluxe, etc.) | Admin | Included in Make reservation | Must be done before selecting specific rooms. |
| Select rooms | Chooses a room based on availability | Admin | Includes → List available rooms | Requires showing all available rooms. |
| List available rooms | Displays all free rooms that match date/type | Admin | Included | Automatically triggered during room selection. |
| Select customer | Picks a customer for the booking | Admin | Includes → List customers Extends → Create customers | If customer doesn’t exist, admin creates one. |
| List customer | Shows all existing customers | Admin | Included | Called when selecting a customer. |
| Create customers | Adds a new customer to the system | Admin | Extends Select customer | Optional step only when there’s no existing record. |
| List active check-ins | Shows all checked-in guests | Admin | — | Independent monitoring function. |
| List hotel reservations | Displays all reservations | Admin | Extends → Define filter | Admin can filter by room type/date. |
| Define filter | Filters reservations | Admin | Extends List hotel reservations | Optional when searching. |
| Register check-in | Marks guest as checked-in | Admin | Extends → Select reservation | Used to complete arrival process. |
| Register check-out | Completes guest stay | Admin | Includes → Process payment | Checkout always needs payment. |
| Process payment | Handles billing | Admin | Includes → Calculate payment | Requires cost computation. |
| Calculate payment | Computes booking charges | Admin | Included | Mandatory part of payment processing. |
| Select reservation | Selects reservation to manage | Admin | Extends other flows | Used in check-in, check-out, review. |

**Sequence diagram**

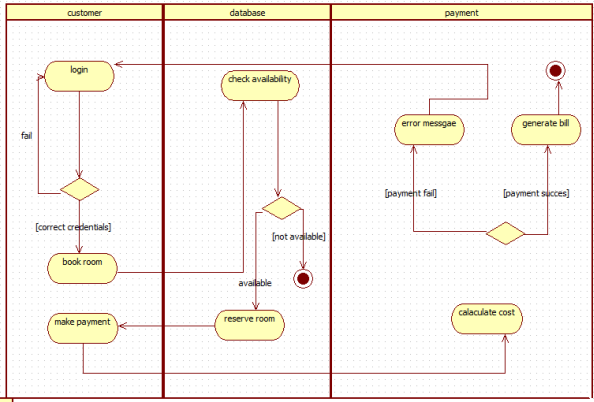


The sequence diagram shows the interaction between the customer, receptionist, staff, and

database during a hotel booking process. The customer visits the hotel, requests accommodation, and the receptionist checks room availability through the staff and database. Once availability is confirmed, the customer can proceed to book or cancel. The receptionist updates the database,

collects payment, and requests billing information. Finally, the receptionist provides the bill to the customer.

**Acitivity diagram**



The activity diagram shows the steps a customer follows to book a room in the hotel system. The customer logs in, and the database checks room availability. If a room is available, the customer proceeds to book and make a payment. The payment system calculates the cost and either

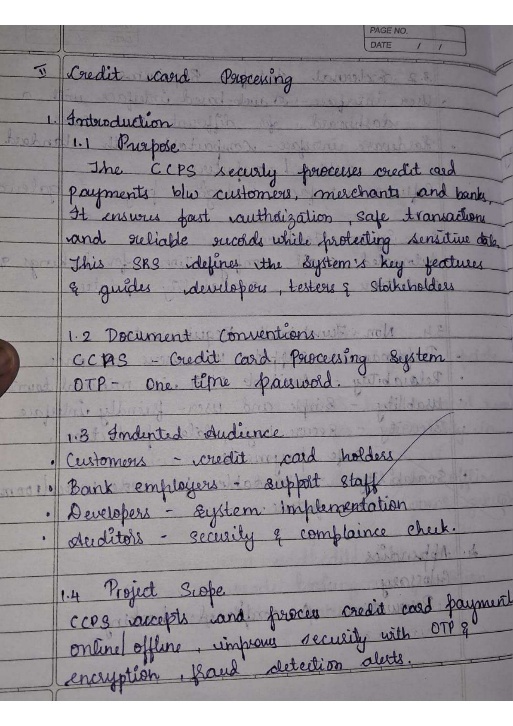
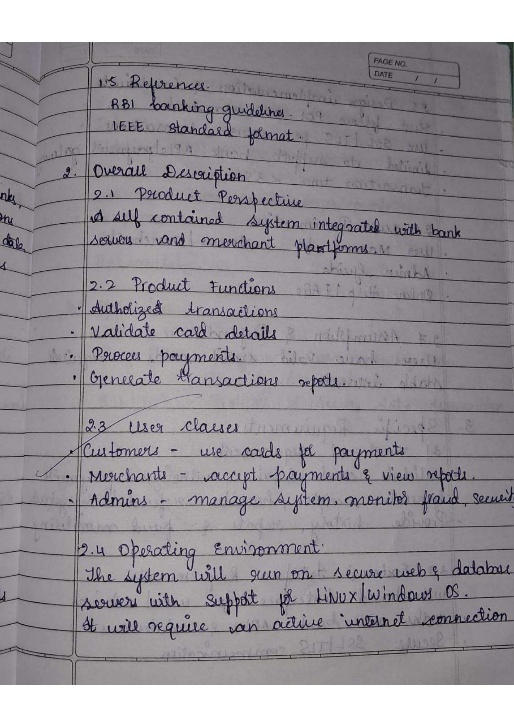
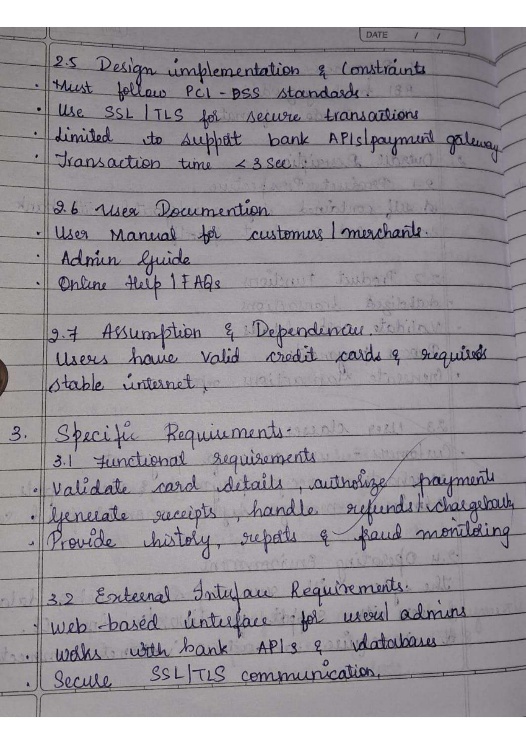
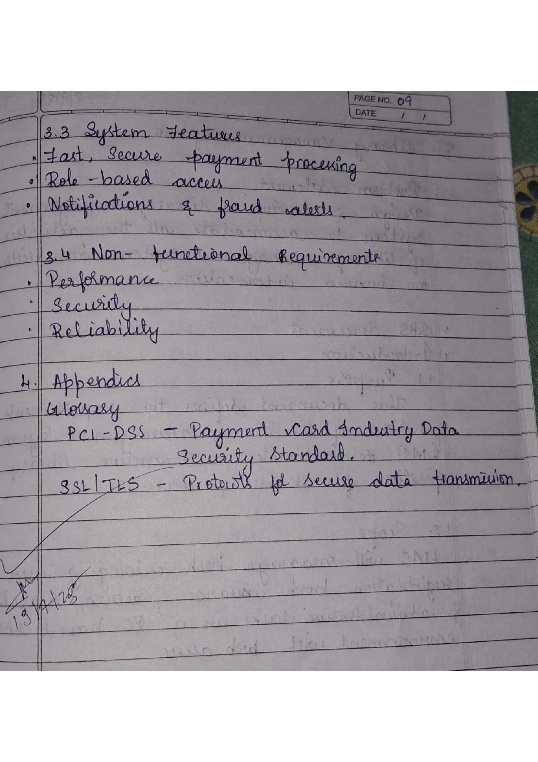
generates the bill for a successful payment or shows an error message if the payment fails.

1. **Credit Card Processing**

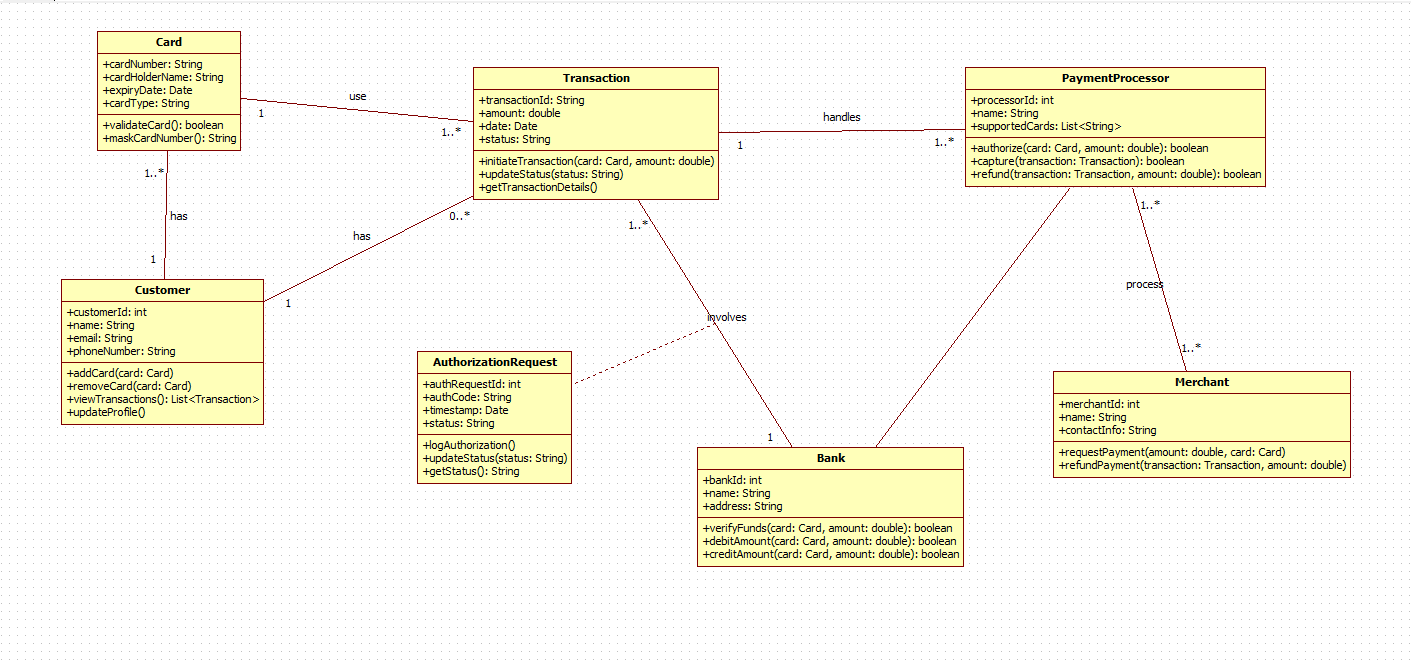
**Problem statement :**

The existing manual and semi-automated credit card processing methods often lead to delays, security risks, and inconsistencies in transaction approvals. Financial institutions and merchants face challenges such as fraud detection, slow authorization response, and lack of real-time monitoring. A secure and efficient Credit Card Processing System is required to automate card verification, authorize transactions instantly, detect fraudulent activities, and ensure compliance with financial security standards. The system should enable smooth communication between merchants, banks, and payment networks to ensure fast, reliable, and secure payment processing.

**SRS-Software Requirements Specification**

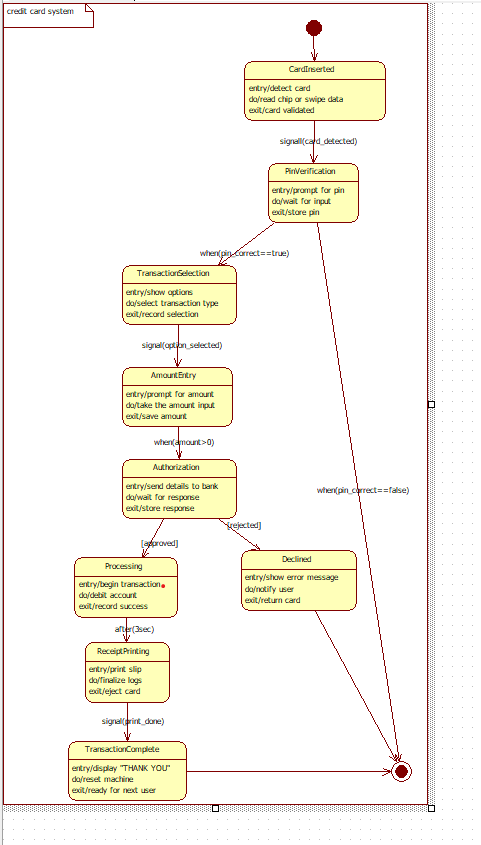
   

**Class diagram**



|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Purpose / What It Does** | **Relationships** | **Explanation** |
| Card | Stores the card number, expiry, CVV, and validity status. Used to perform transactions. | Card → Transaction (uses) Card → Customer (belongs to) | A card belongs to one customer but can be used in many transactions. |
| Customer | Stores customer details and linked cards. Can view transactions and update profile. | Customer → Card (1 to many) Customer → Transaction (makes) | One customer can have multiple cards and can make many transactions. |
| Transaction | Represents a payment made using a card. Stores amount, date, and status. | Transaction → Card Transaction → Bank Transaction → PaymentProcessor Transaction → Merchant | A transaction involves a card, a bank, a merchant, and is handled by a processor. |
| AuthorizationRequest | Sends verification request to the bank before approving a transaction. | AuthorizationRequest → Bank (sent to) AuthorizationRequest → Transaction (belongs to) | A transaction generates at most one authorization request. |
| Bank | Verifies card validity, approves transactions, debits or credits accounts. | Bank → AuthorizationRequest Bank → Transaction | The bank approves or rejects transaction requests. |
| PaymentProcessor | Processes payments between card, merchant, and bank. Handles authorizations, captures, and refunds. | PaymentProcessor → Transaction PaymentProcessor → Merchant | The processor manages the flow of each transaction to the bank and merchant. |
| Merchant | Represents the business receiving payment. Requests payment and processes refunds. | Merchant → PaymentProcessor Merchant → Transaction | A merchant initiates transaction processing through a payment processor. |

**State Diagram:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| CardInserted | Detects card and validates chip/magnetic data. | card\_detected | PinVerification | First step when card is inserted. |
| PinVerification | Prompts for PIN and validates it. | pin\_correct == true | TransactionSelection | User enters PIN. |
| TransactionSelection | Shows transaction options (withdrawal, balance, etc.). | option\_selected | AmountEntry | User selects operation. |
| AmountEntry | User enters amount. | amount > 0 | Authorization | System prepares for authorization. |
| Authorization | Sends details to bank for approval. | approved | Processing | Bank approves or rejects. |
| Processing | Debits account and confirms transaction. | after(3 sec) | ReceiptPrinting | Actual transaction execution. |
| ReceiptPrinting | Prints slip and finalizes logs. | print\_done | TransactionComplete | Final step before completion. |
| TransactionComplete | Displays thank you and resets machine. | — | End State | Process ends. |
| Declined | Shows error if PIN or bank check fails. | rejected | End | Alternative failure path. |

**Use-Case Diagram:**

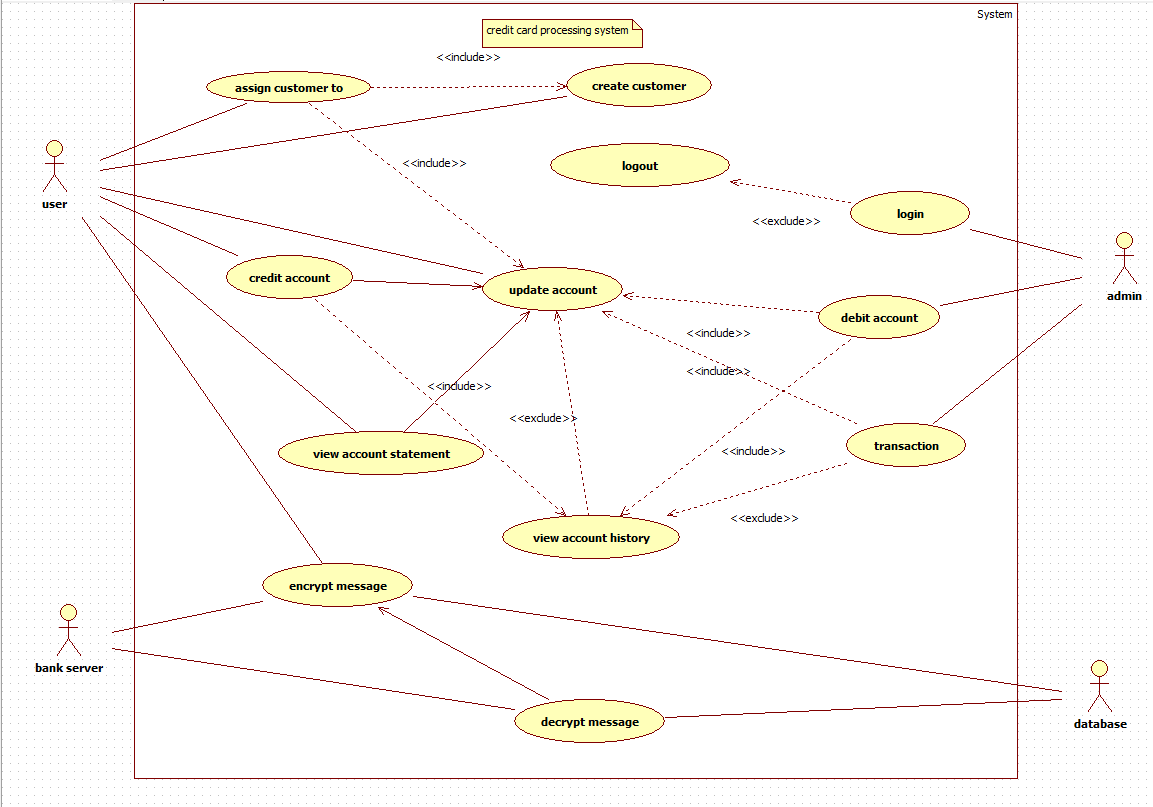


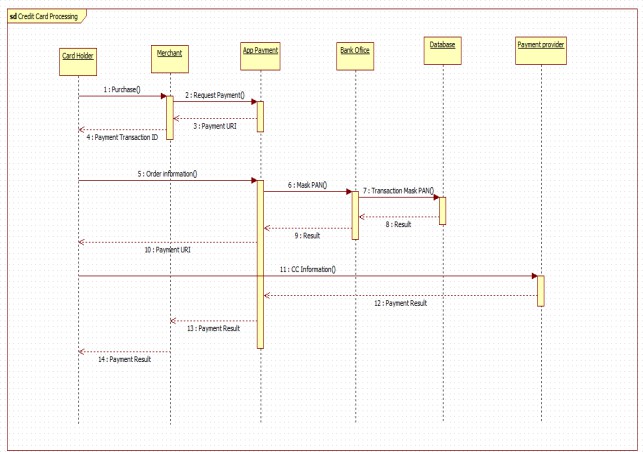
TABLE 1: Actors & Responsibilities

|  |  |
| --- | --- |
| **Actor** | **Responsibilities** |
| User (Customer) | Creates account, credits/debits account, views statements, encrypt/decrypt messages for secure transactions. |
| Admin | Manages login, creates customer accounts, updates/deletes account records, handles transactions. |
| Bank Server | Encrypts and decrypts sensitive messages, validates transactions. |
| Database | Stores account data, transaction history, login credentials. |

TABLE 2: Use Cases, Purpose, and Relationships

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Purpose** | **Actor** | **Relationship** | **Explanation** |
| Assign customer to | Links a customer to account | User | Includes → Create customer | Customer creation may be required first. |
| Create customer | Registers new user | Admin | — | Admin adds customer to database. |
| Login | Admin or user logs in | Admin | Extends → Logout | After login, logout becomes available. |
| Logout | Ends the session | Admin | — | Ends active session. |
| Credit account | Adds funds to account | User | Includes → Update account | Updates logged account. |
| Debit account | Deducts money from account | User | Includes → Update account | Requires account update. |
| Update account | Modifies account balance or details | Admin / User | — | Central use case for all transactions. |
| View account statement | Shows recent transactions | User | Includes → View account history | Needs history data. |
| View account history | Shows complete transaction record | User | Extends (optional) | Called if statement needs detail. |
| Encrypt message | Secures message before sending | Bank server | — | Used for transaction security. |
| Decrypt message | Converts encrypted message back | Database | — | Data reading from secure source. |
| Transaction | Handles payment transfer | Admin | Included by debit/credit | Triggered during money operations. |

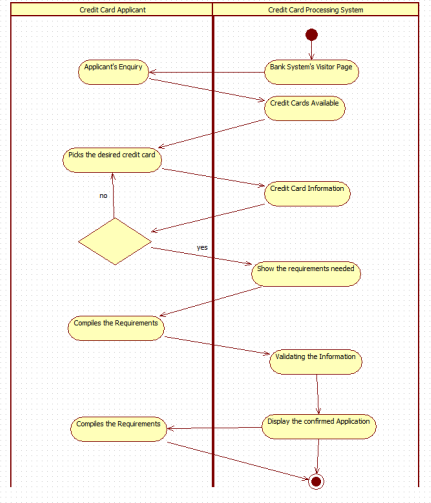
**Sequence diagram**



The sequence diagram shows the flow of a credit card payment from purchase to completion.

The Card Holder initiates a payment, the Merchant sends the request, and the App Payment system processes the order by masking card details and communicating with the Bank Office and Database for verification. After validation, the payment result is returned through the App Payment system to the Merchant, Payment Provider, and finally to the Card Holder.

**Acitivity diagram**



The activity diagram illustrates the process of a credit card applicant applying through the bank’s credit card processing system. The applicant sends an enquiry, views available credit cards, and selects one. The system displays the card information and required documents. After the applicant submits the necessary requirements, the system validates the information and finally

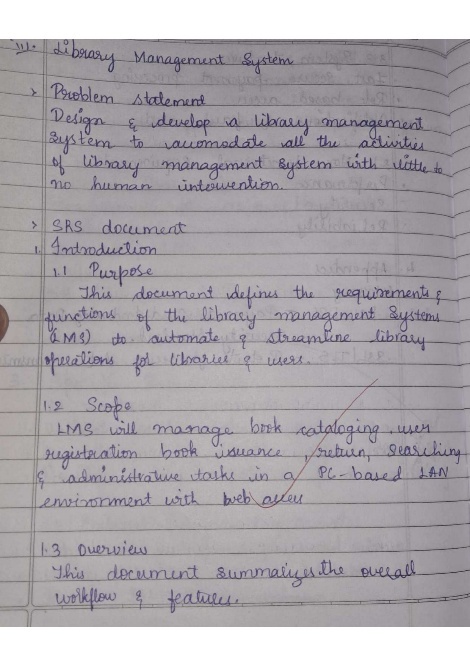
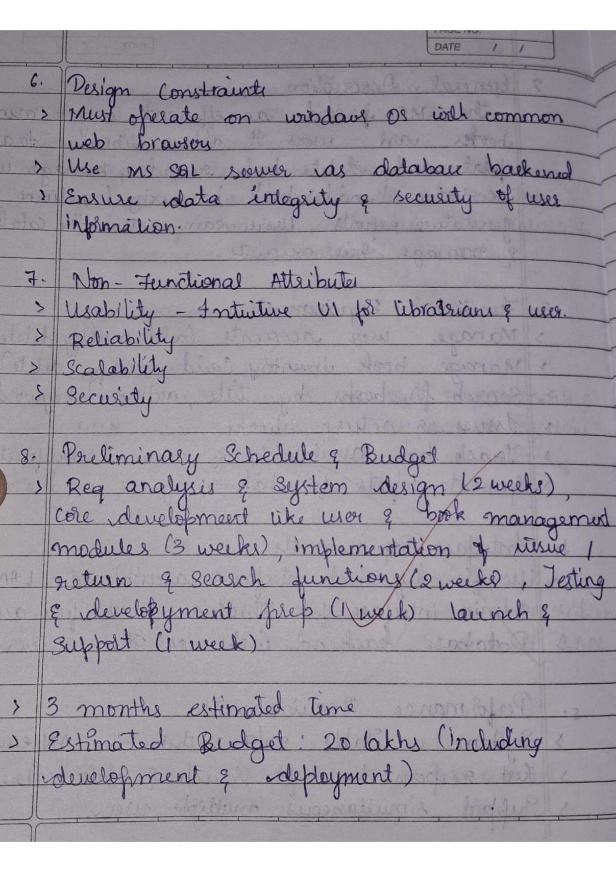
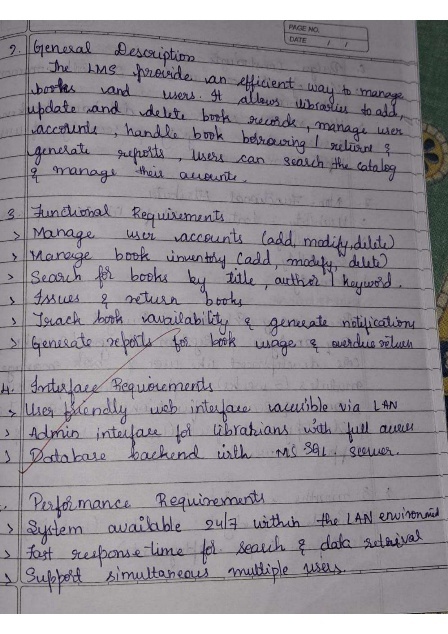
displays the confirmed application.

1. **. Library Management System**

**Problem statement :**

Traditional library management relies heavily on manual book issuance, record keeping, and fine calculation, which leads to errors, loss of data, and inefficiency during peak hours. Students and staff often face difficulty in finding books, checking availability, and tracking due dates. Therefore, an automated Library Management System is needed to streamline book cataloging, member registration, lending/return operations, and fine management. The system should provide quick search capabilities, maintain accurate inventory records, and improve overall library operations with reduced manual workload.

**SRS-Software Requirements Specification**

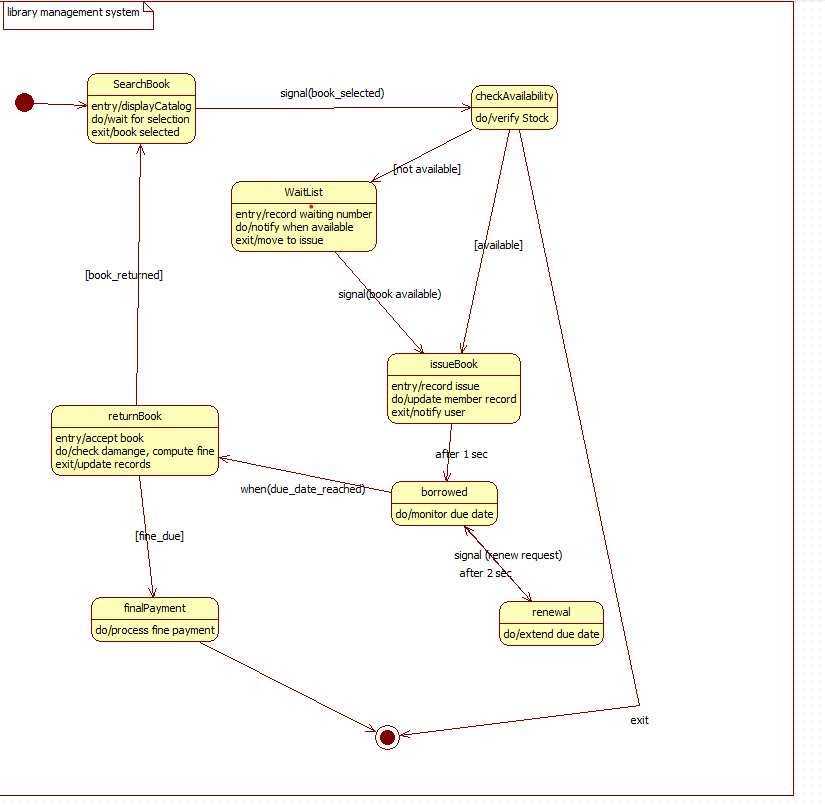
 

**Class diagram**



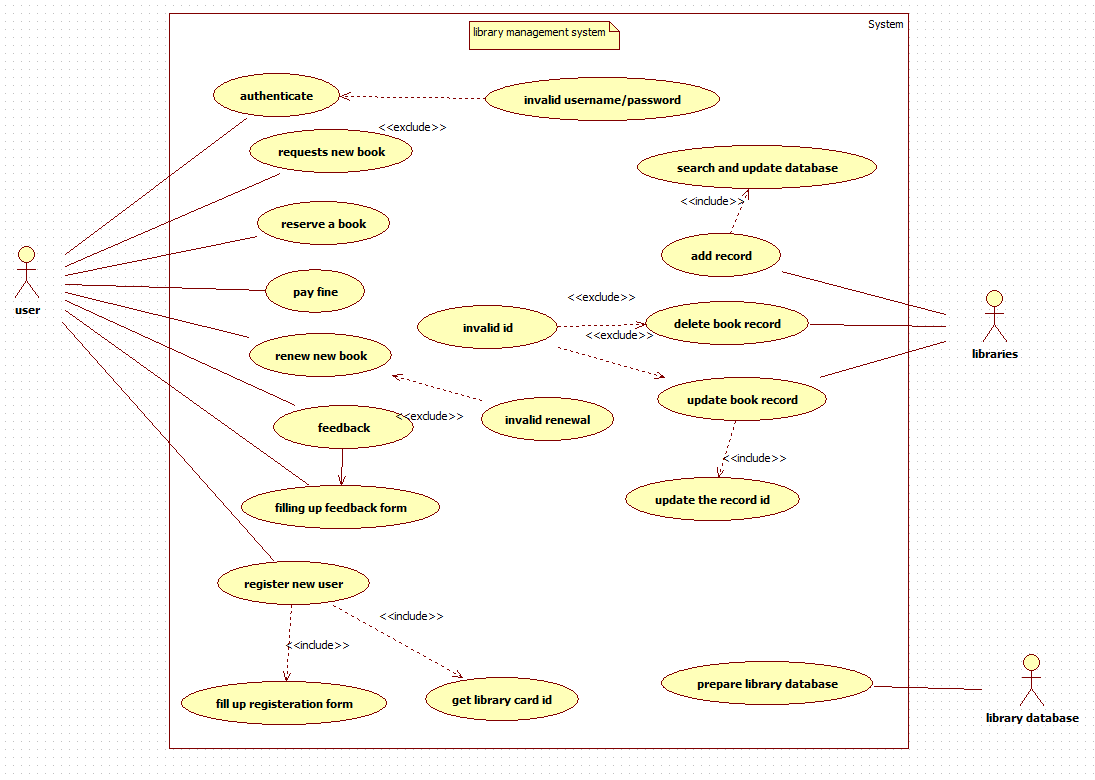
|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Purpose** | **Relationships** | **Explanation** |
| Library | Contains library information and manages book inventory. | Library → Book (has) | One library contains many books. |
| Staff | Manages library operations, adds books, and generates reports. | Staff → Library (works for) | Staff works in a specific library. |
| Book | Holds book details such as title, author, ISBN, and availability. | Book → Loan (lent to) Book → Reservation (reserved by) Book → Library | One book can be loaned many times or reserved by members. |
| Member | Represents library members and their borrowed/reserved books. | Member → Loan (borrows) Member → Reservation (reserves) | A member can borrow multiple books and place multiple reservations. |
| Loan | Represents the borrowing of a book. Stores due date, issue date, and fines. | Loan → Member Loan → Book | One loan is created for one member and one book. |
| Reservation | Represents a book reserved by a member. | Reservation → Member Reservation → Book | One reservation links one book to one member. |

**State Diagram:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| SearchBook | User searches and selects a book. | book\_selected | checkAvailability | First step of book issue. |
| checkAvailability | System checks if book is available. | available | issueBook | Availability determines next state. |
| WaitList | Records waiting number for unavailable books. | book\_available | issueBook | User waits until book is returned. |
| issueBook | Issues book and updates records. | after(1 sec) | borrowed | Book issued to member. |
| borrowed | Member currently holds the book. | due\_date\_reached | returnBook | Borrowing period active. |
| returnBook | Accepts returned book and computes fines. | fine\_due | finalPayment | Book returned. |
| finalPayment | Processes fine payments if needed. | — | Exit | Last step before exit. |
| renewal | Extends book borrowing time. | renew\_request | borrowed | Optional extension path. |

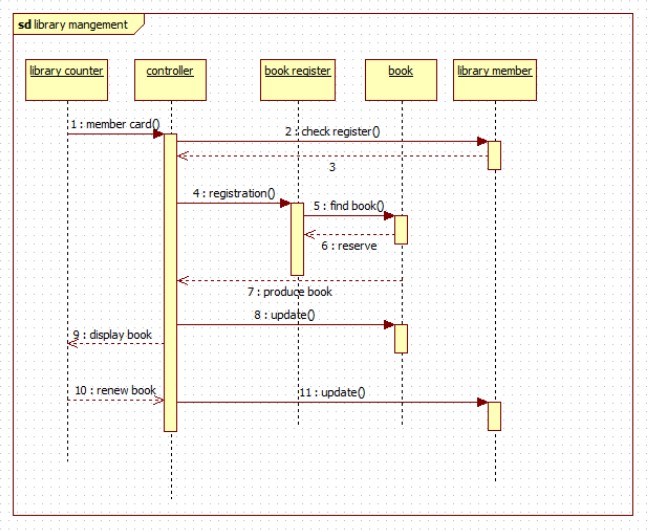
**Use-Case Diagram:**



|  |  |
| --- | --- |
| **Actor** | **Description / Responsibilities** |
| User (Student/Member) | Authenticates into the system, reserves books, requests new books, renews books, pays fines, registers as new user, and submits feedback. |
| Libraries (Librarian) | Manages book records, updates database, adds books, deletes books, updates book information. |
| Library Database | Stores and maintains all books, users, and transaction records (system actor). |

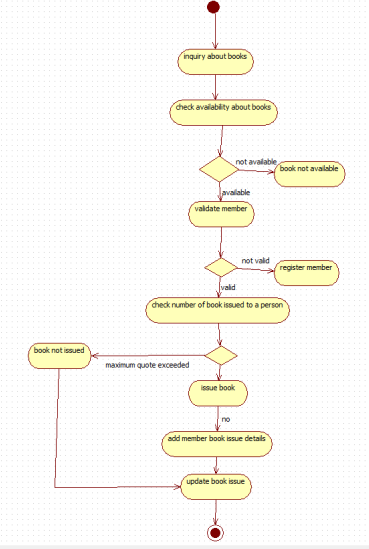
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Purpose / What It Does** | **Actor** | **Relationship** | **Explanation** |
| Authenticate | Verifies username and password | User | Extends → Invalid username/password | If authentication fails, invalid message displays. |
| Request new book | User requests a new book to be added | User | — | Sent to library staff for processing. |
| Reserve a book | Reserves an available book | User | — | Regular book reservation. |
| Pay fine | Pays overdue fine | User | — | Triggered after return processing. |
| Renew new book | Extends borrowing period | User | Extends → Invalid renewal | Renewal may fail if due date passed. |
| Feedback | Provides comments or suggestions | User | Includes → Fill feedback form | Feedback needs form filling. |
| Fill feedback form | Submits detailed feedback | User | Included | Always part of feedback. |
| Register new user | Creates a new member account | User | Includes → Fill registration form | Registration requires filling user details. |
| Fill registration form | Captures user details | User | Included | Required for new user creation. |
| Get library card ID | Generates card ID for new user | User | Included | Follows registration. |
| Invalid ID | Displays error if book ID incorrect | User | Extends Book update/delete | Shown only when ID mismatch. |
| Search and update database | Searches books and updates data | Libraries | Includes → Add record / Update book record | Required for modifying library records. |
| Add record | Adds new book to system | Libraries | — | Part of managing library. |
| Update book record | Updates existing book info | Libraries | Includes → Update record ID | Requires ID validation. |
| Delete book record | Deletes selected book | Libraries | Extends → Invalid ID | Deletion fails if ID invalid. |
| Prepare library data | Prepares master book file | Library database | — | Back-end system processing. |

**Sequence diagram**



The sequence diagram shows the process of managing book transactions in a library. A library member presents their card at the library counter, which triggers the controller to check the book register, register the member, and search for the requested book. Once the book is found and reserved, it is produced for the member. The system updates the book’s status, displays it to the member, and also handles book renewal requests, updating the records accordingly.

**Acitivity diagram**



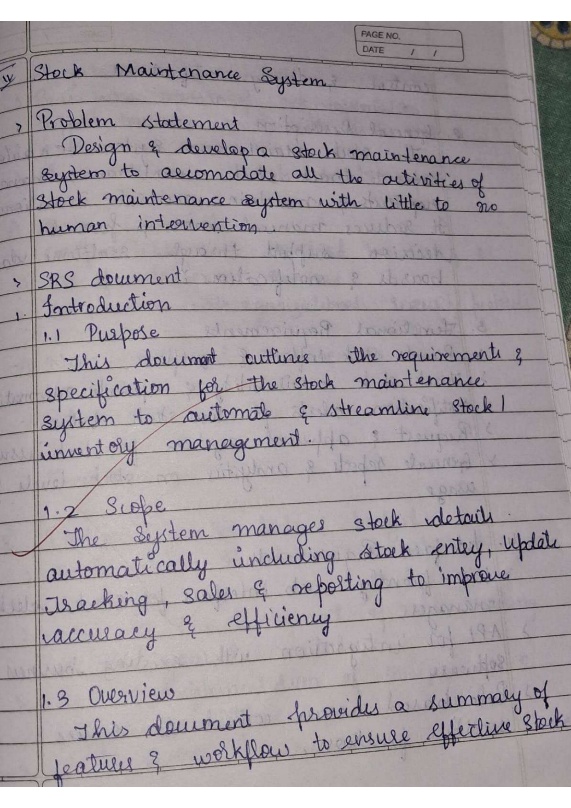
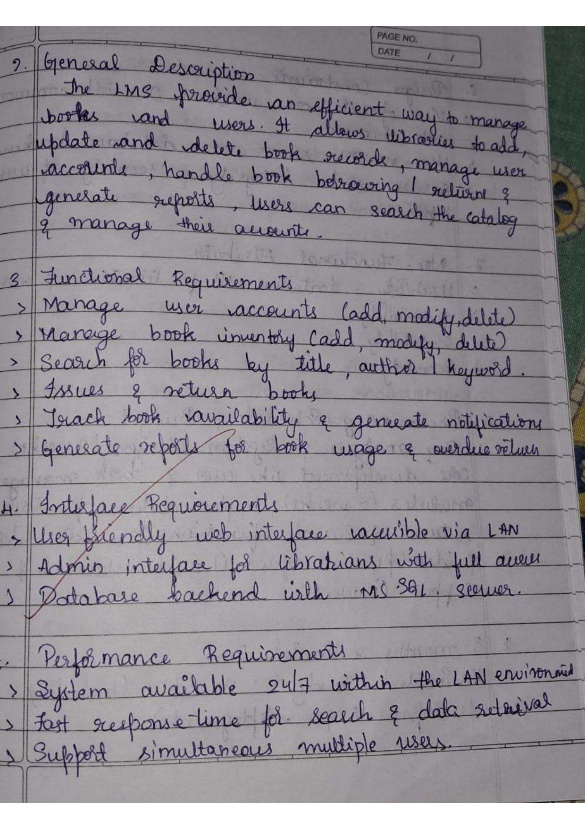
The activity diagram outlines the process of issuing a book in a library. It begins with a user inquiring about a book and checking its availability. If available, the system validates the member; if not registered, the member is asked to register. The system then checks whether the member has exceeded the maximum number of issued books. If eligible, the book is issued, the member’s issue details are added, and the book issue record is updated. Otherwise, the book is not issued.

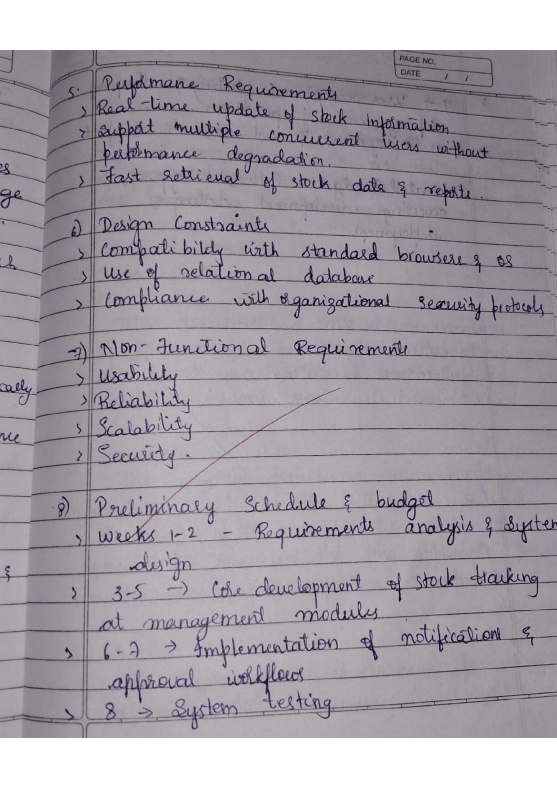
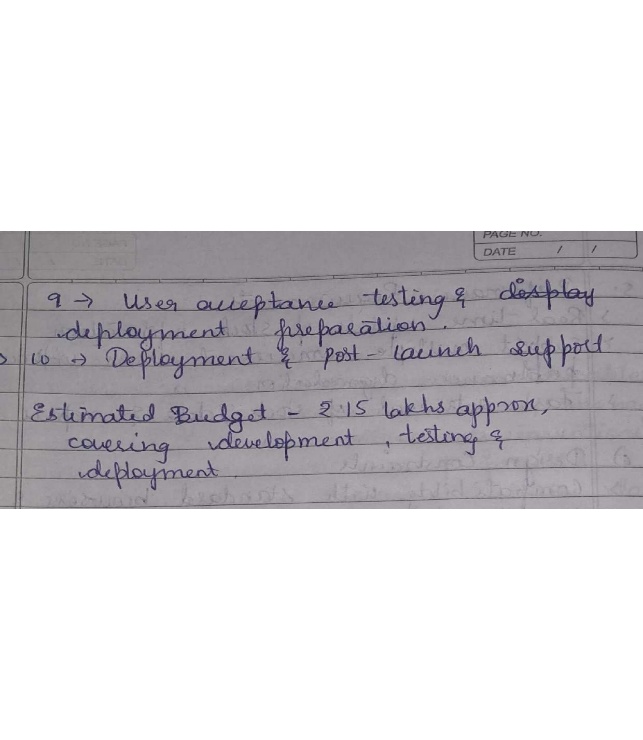
1. **. Stock Maintenance System**

**Problem statement :**

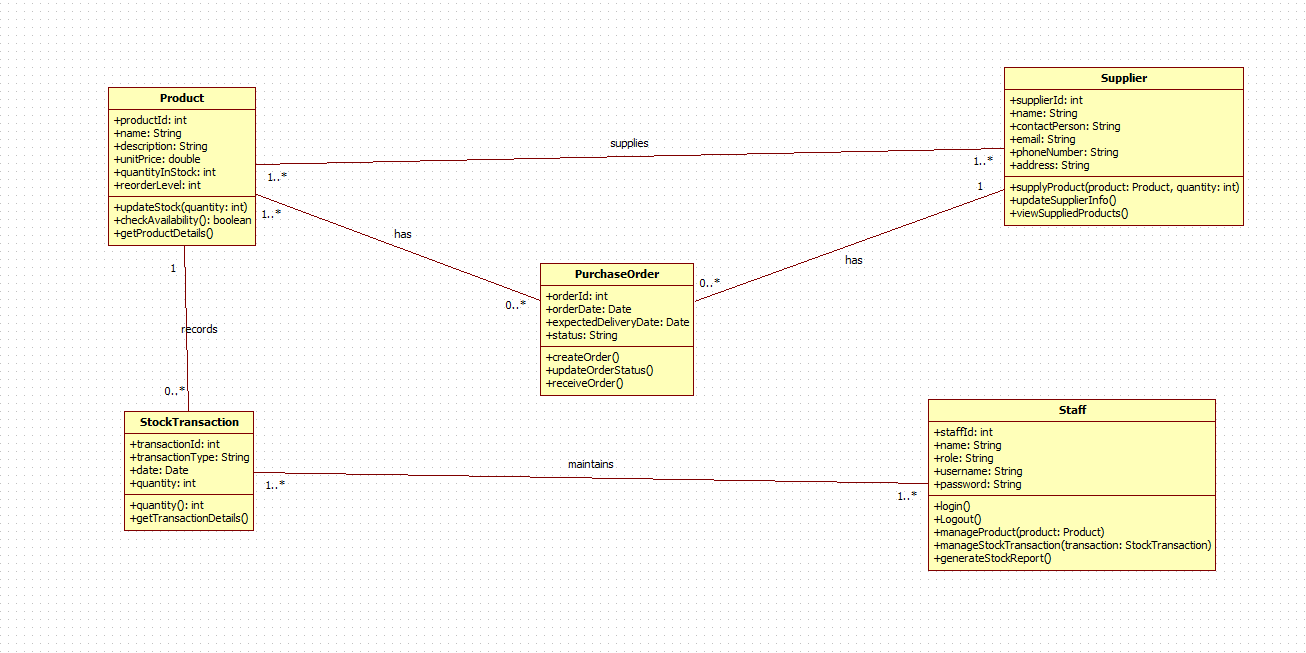
Businesses often struggle to maintain accurate stock details, record stock movements, track damaged goods, and ensure timely replenishment due to manual monitoring methods. This leads to inconsistencies, loss of inventory, and delays in decision-making. A Stock Maintenance System is required to efficiently record stock entries, monitor stock levels, track item conditions, update stock usage, and generate maintenance logs. The system should help staff maintain accurate inventory records, reduce manual errors, and support smooth operational flow.

**SRS-Software Requirements Specification**

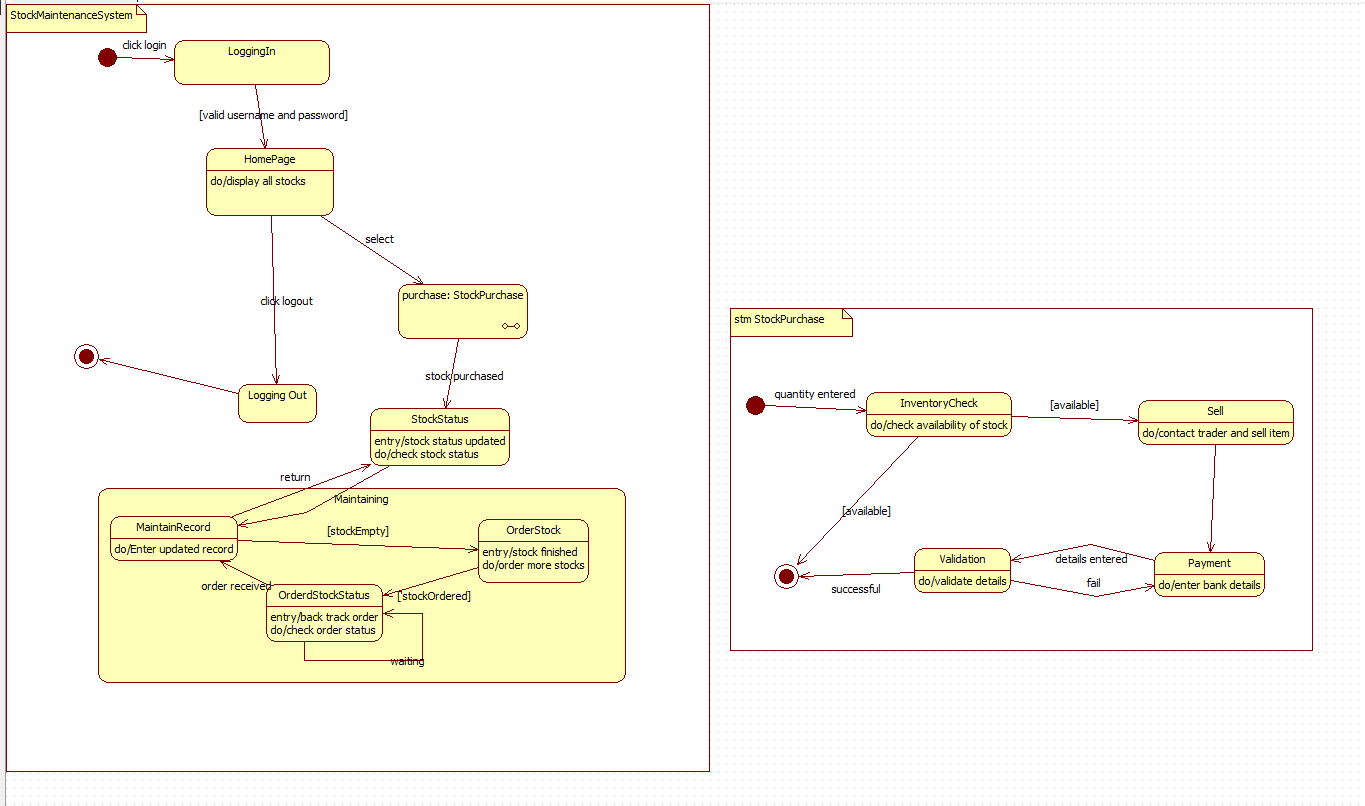
 

**Class dagram**



|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Purpose** | **Relationships** | **Explanation** |
| Product | Stores product details like name, price, quantity, and reorder level. | Product → PurchaseOrder (ordered via) Product → StockTransaction | Each product can be included in multiple orders and stock transactions. |
| Supplier | Holds supplier details and provides products. | Supplier → PurchaseOrder (supplies) | A supplier supplies many purchase orders. |
| PurchaseOrder | Represents product orders sent to suppliers. | PurchaseOrder → Product PurchaseOrder → Supplier | One order can contain multiple products and is assigned to one supplier. |
| StockTransaction | Records stock in/out movements. | StockTransaction → Product StockTransaction → Staff | A stock transaction updates product quantity and is recorded by staff. |
| Staff | Maintains stock records and generates reports. | Staff → StockTransaction Staff → PurchaseOrder | Staff manages order creation and stock transactions. |

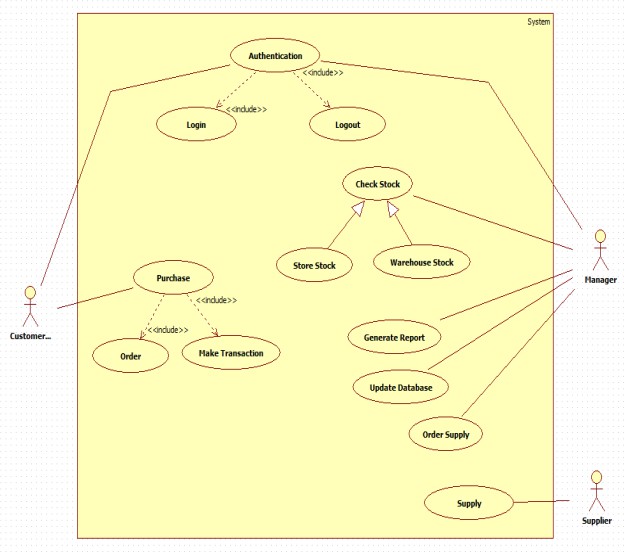
**State Diagram:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| LoggingIn | Validates credentials. | valid username/password | HomePage | User logs into the system. |
| HomePage | Displays all stock and menu options. | purchase: StockPurchase | StockPurchase | Main dashboard. |
| LoggingOut | Clears session and logs user out. | logout | End | User exits the system. |
| StockStatus | Updates stock information. | stockPurchased | MaintainRecord | System updates after purchase. |
| MaintainRecord | Records updated stock details. | stockEmpty | OrderStock | Handles stock records. |
| OrderStock | Places purchase order for low stock. | stockOrdered | OrderStockStatus | Stock order initiated. |
| OrderStockStatus | Tracks order status. | order\_received | MaintainRecord | Monitor arrival of ordered stock. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| InventoryCheck | Checks if stock available for sale. | available | Sell | First validation step. |
| Sell | Contacts trader and sells goods. | — | Payment | Item sold. |
| Payment | Enters payment/bank details. | details entered | Validation | Payment information processed. |
| Validation | Confirms transaction details. | available | Success | Final confirmation. |

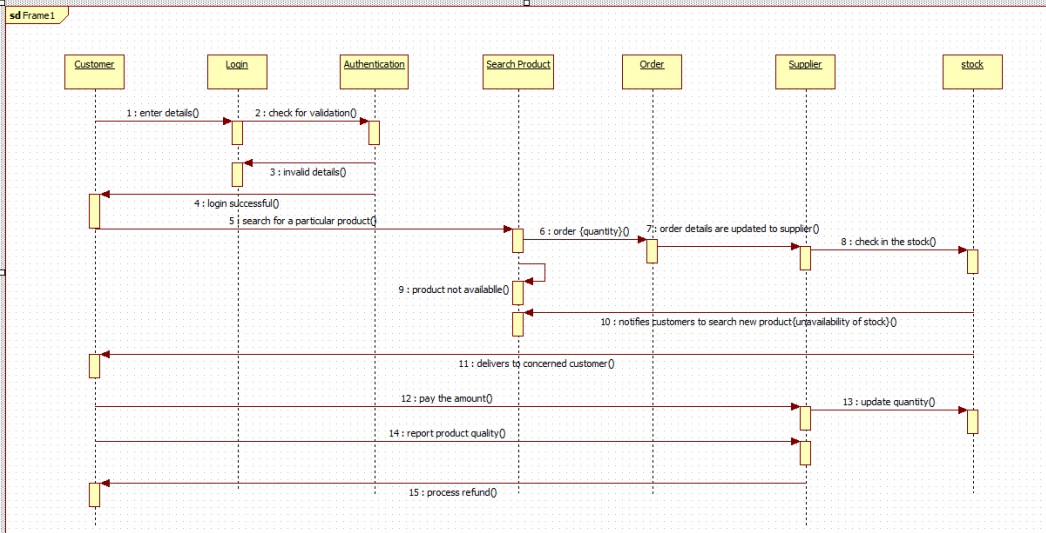
**Use-Case Diagram:**



|  |  |
| --- | --- |
| **Actor** | **Responsibilities** |
| Staff | Manages stock, updates records, purchases stock, checks stock status, logs into system. |
| Supplier | Receives purchase orders and delivers stock. |
| Trader (System Actor) | Used in the StockPurchase sub-process (for selling). |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Purpose** | **Actor** | **Relationship** | **Explanation** |
| Logging in | Authenticates staff | Staff | — | Entry point for system. |
| HomePage | Shows stock options | Staff | — | Main menu. |
| Logout | Ends user session | Staff | — | Exit action. |
| Purchase: StockPurchase | Starts stock buying workflow | Staff | — | Leads to sub-state machine. |
| StockStatus | Updates status of stock | Staff | — | Triggered after purchase. |
| MaintainRecord | Updates stock details | Staff | — | Manual corrections or updates. |
| OrderStock | Places new order | Staff | — | Triggered when stock empty. |
| OrderStockStatus | Tracks order state | Staff | Extends OrderStock | Optional repeated checking. |
| InventoryCheck | Checks if stock available | Staff | — | Part of StockPurchase flow. |
| Sell | Sells stock to trader | Staff | — | Sub-process step. |
| Payment | Processes payment for stock | Staff | — | Needed for transaction. |
| Validation | Confirms final details | Staff | — | Ends purchase workflow. |

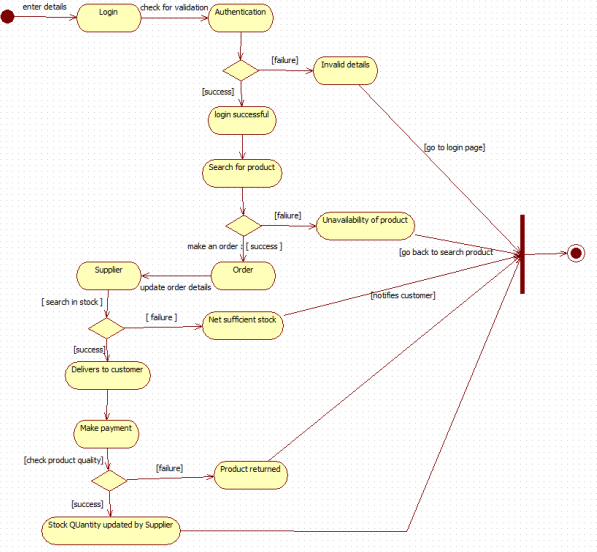
**Sequence diagram**



The sequence diagram illustrates the workflow of an online product ordering system. A customer logs in, searches for a product, and places an order. The system forwards order details to the supplier, who checks stock availability. If the product is unavailable, the system notifies the customer. If available, the supplier delivers the product, the customer makes the payment, and stock levels are updated. The customer may also report product quality, after which the system

processes a refund if required.

**Acitivity diagram**



The activity diagram shows the workflow of an online product purchase process. The customer logs in and is authenticated; if valid, they search for a product and place an order. The supplier

checks stock availability and updates order details. If stock is sufficient, the product is delivered, payment is made, and product quality is checked. If the product fails the quality check, it is returned; if successful, stock quantity is updated. Invalid login or unavailable products redirect

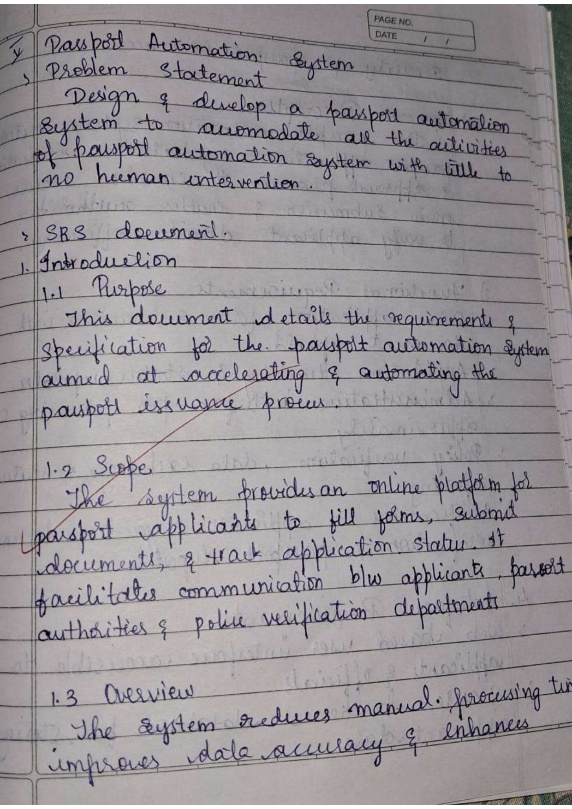
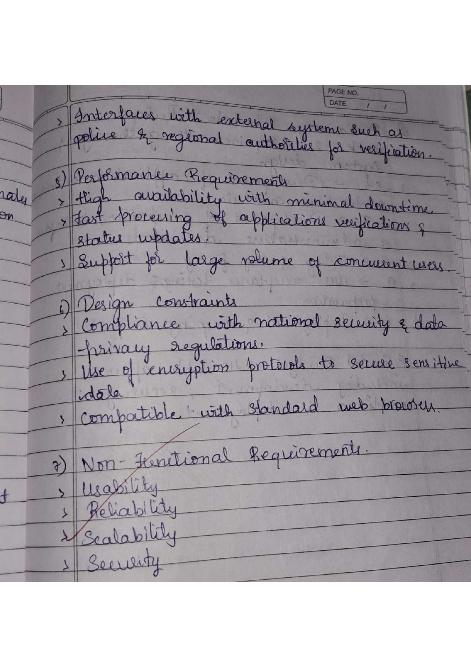
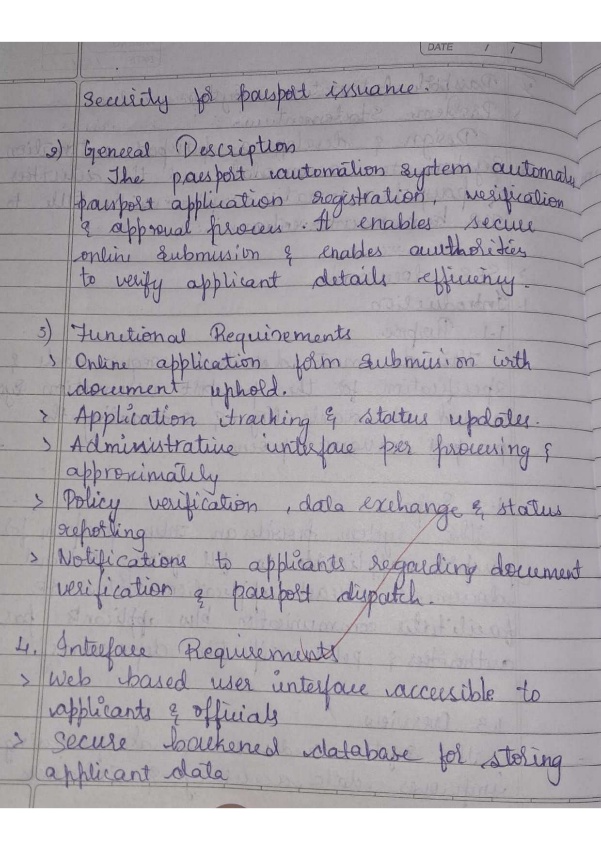
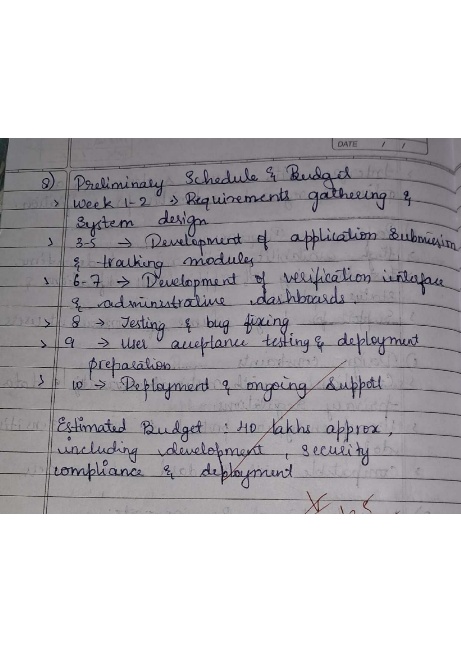
the user back to the appropriate steps.

1. **Passport Automation System**

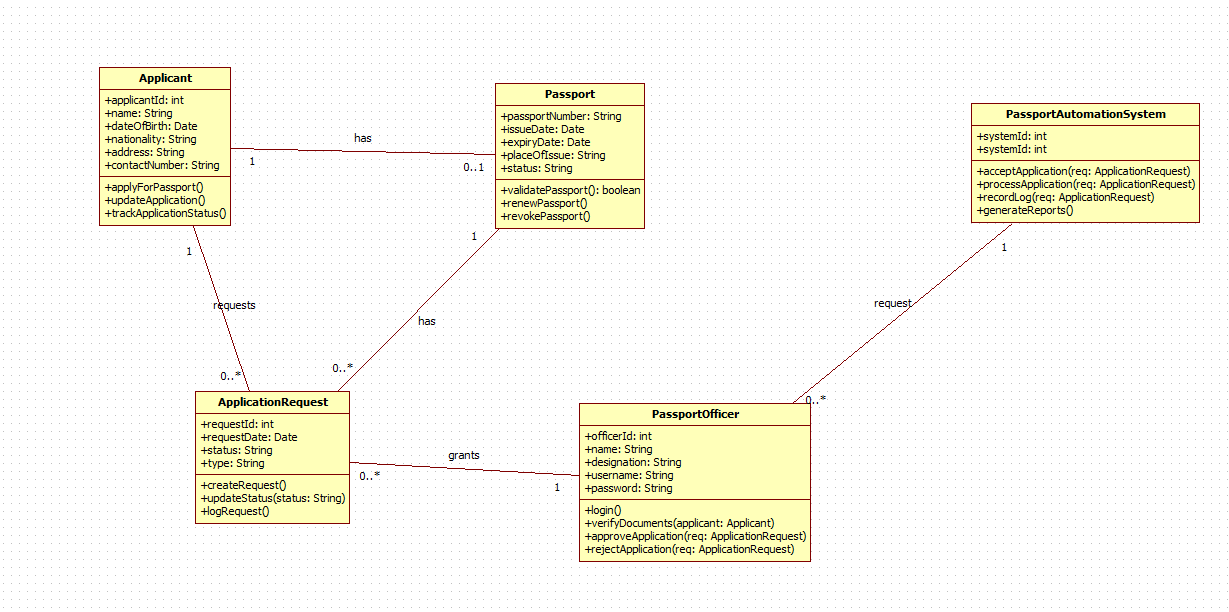
**Problem statement :**

Manual passport application and processing procedures are often time-consuming, prone to human errors, and require applicants to repeatedly visit passport offices for verification and documentation. This results in delays, inefficient document handling, and difficulty in tracking application status. Therefore, a Passport Automation System is required to streamline passport application submission, document verification, appointment scheduling, fee payment, and status tracking. The system should automate workflow steps, reduce manual intervention, improve accuracy, and provide applicants with a transparent and user-friendly experience.

**SRS-Software Requirements Specification**

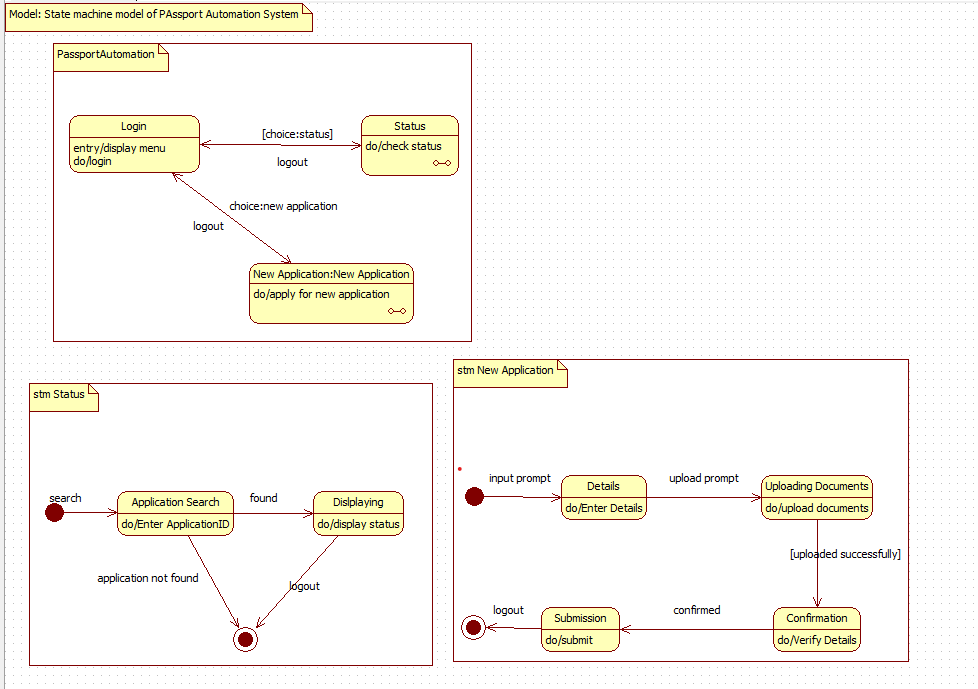
**  **

**Class diagram**



|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Purpose** | **Relationships** | **Explanation** |
| Applicant | Stores applicant information and manages passport applications. | Applicant → ApplicationRequest Applicant → Passport | One applicant can submit multiple requests and have a passport. |
| Passport | Stores passport details such as issue/expiry date and status. | Passport → Applicant Passport → ApplicationRequest | One passport belongs to an applicant and is created after request approval. |
| ApplicationRequest | Handles the request for a new passport or renewal. | ApplicationRequest → Applicant ApplicationRequest → PassportOfficer | One applicant can submit many requests; each request is handled by one officer. |
| PassportOfficer | Verifies documents, approves or rejects applications. | PassportOfficer → ApplicationRequest | Officers validate each application request. |
| PassportAutomationSystem | Coordinates application submission, verification, and reporting. | System → ApplicationRequest | The system receives and manages all application requests. |

**State Diagram:**

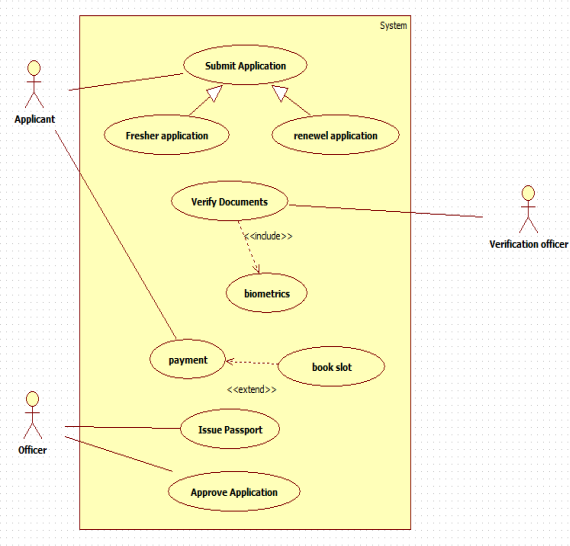


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| Login | User logs into system and views menu. | choice:status | Status | First interaction. |
| Status | Checks application status. | logout | Login | User views current passport progress. |
| New Application | User applies for new passport. | choice:new application | New Application Flow | Moves to application submission. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| Application Search | User enters application ID. | found | Displaying | First step for status check. |
| Displaying | Shows application status to user. | logout | End | Status displayed. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State** | **Description** | **Trigger** | **Next State** | **Explanation** |
| Details | User enters basic personal information. | upload prompt | Uploading Documents | First step of new passport. |
| Uploading Documents | User uploads required documents. | uploaded successfully | Confirmation | Document verification step. |
| Confirmation | System verifies data and documents. | confirmed | Submission | Validation phase. |
| Submission | Final submission and ID generation. | logout | End | Completes application. |

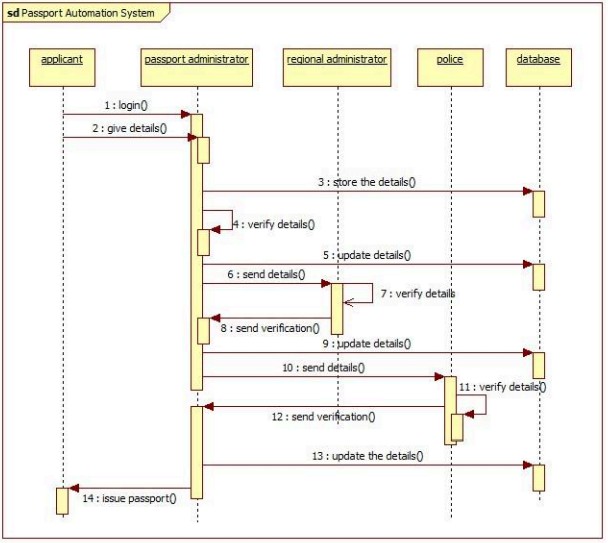
**Use-Case Diagram:**



|  |  |
| --- | --- |
| **Actor** | **Responsibilities** |
| Applicant | Logs in, checks status, applies for new passport, submits documents. |
| Passport Officer | Reviews applications, verifies documents, approves/rejects. |
| System | Stores details, processes applications, shows status. |

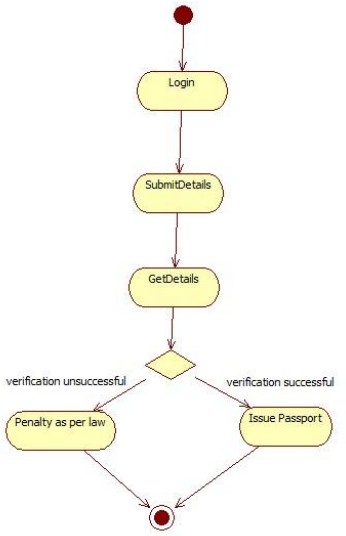
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Purpose** | **Actor** | **Relationship** | **Explanation** |
| Login | Applicant logs into system | Applicant | — | Starting point of use. |
| Status | Shows passport application status | Applicant | — | Called after login when checking progress. |
| New Application | Starts new passport request | Applicant | Includes → Details, Upload Documents, Confirmation, Submission | New application process has multiple mandatory steps. |
| Application Search | Searches application by ID | Applicant | Extends Status | Optional when user wants to see status. |
| Details | Enters personal information | Applicant | Included | Mandatory part of new application. |
| Upload Documents | Uploads ID proof, address proof | Applicant | Included | Required to complete application. |
| Confirmation | System verifies details | System | Included | Automatic verification stage. |
| Submission | Final submission | Applicant | Included | Final step in new application. |
| Logout | Ends session | Applicant / Officer | — | Optional action. |

**Sequence diagram**



The sequence diagram shows the workflow of a passport automation system. The applicant logs in and submits details, which the passport administrator stores in the database. The regional administrator and police department sequentially verify the submitted information and update the records. After receiving verification from both authorities, the passport administrator issues the passport to the applicant.

**Acitivity diagram**



The activity diagram shows a simple passport verification process. A user logs in, submits their details, and the system retrieves and checks the information. If verification is successful, a passport is issued; if verification fails, a legal penalty is applied. The process then ends.