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

Lab Record

Object Oriented Modelling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

SHREYAS RAO M

1BM22CS272

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
September-January 2025

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Modelling (23CS5PCOOM) laboratory has been carried out by SHREYAS RAO M (1BM22CS272) during the 5th Semester Sep 24- Jan2025.

Signature of the Faculty Incharge:

Radhika A. D
Asst. Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System

Problem Statement

Managing hotel operations efficiently is a complex task due to the diverse range of activities involved, such as room reservations, guest check-ins and check-outs, staff coordination, inventory management, and billing. Many hotels still rely on manual processes or outdated systems, which can lead to inefficiencies, human errors, and poor customer experiences.

The lack of a centralized, user-friendly system makes it challenging to manage guest bookings, track room availability, handle billing, and maintain seamless communication between departments. Additionally, there is often no integration with modern features like online bookings, digital payments, or data analytics, which are essential for staying competitive in the hospitality industry.

SRS – Software Requirements Specification

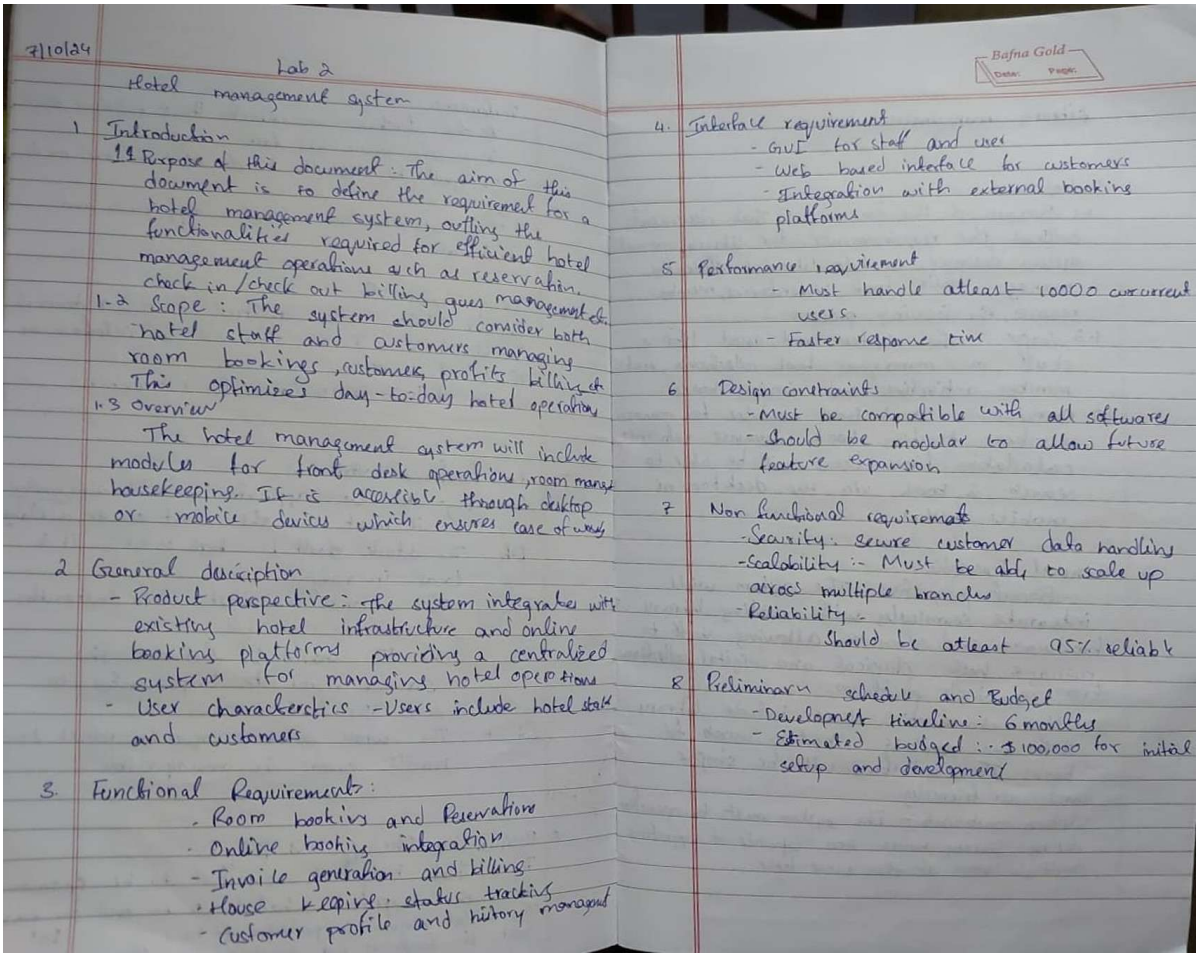


Figure 1.1: SRS Hotel Management

Class Diagram

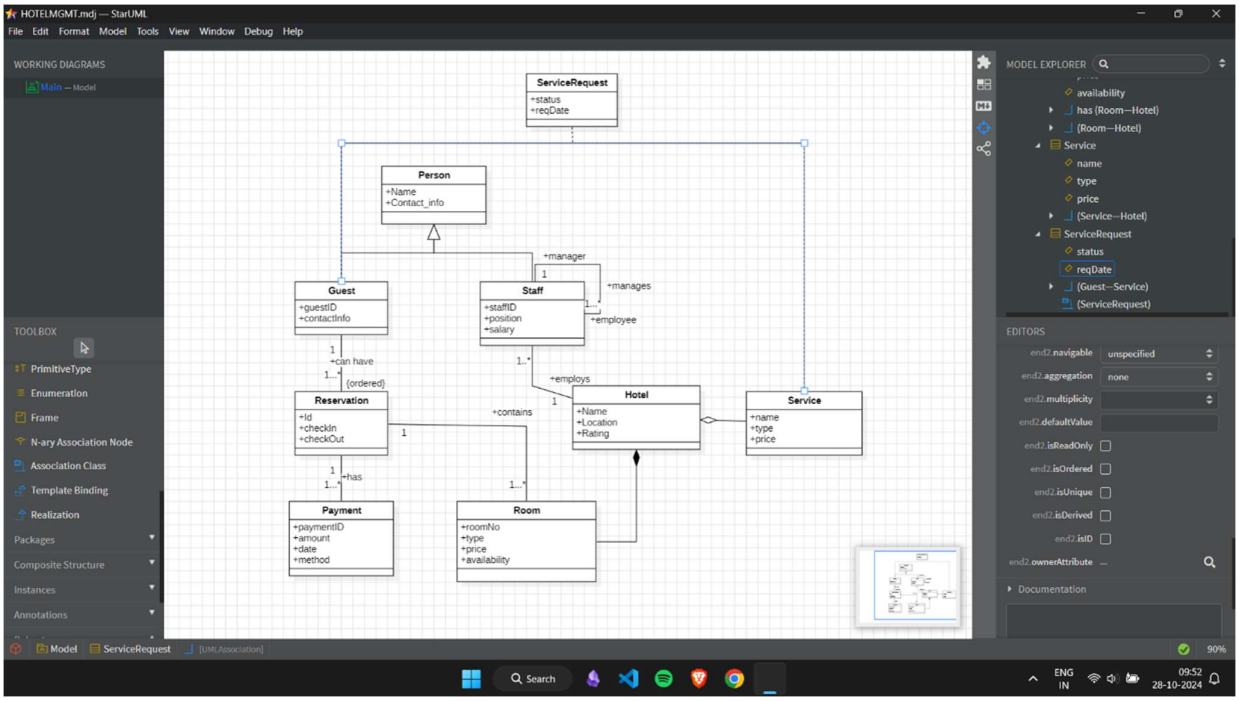


Figure 1.2: Class Diagram

1. **Person**:
    - Attributes: Name, Contact_info
    - Represents a general person in the system with basic details such as name and contact information.
2. **Guest**:
    - Attributes: guestID, contactInfo
    - A specialized class of Person representing guests of the hotel. It includes a unique guest ID and contact information.
3. **Staff**:
    - Attributes: staffID, position, salary
    - Another specialized class of Person representing staff members. This class includes attributes for staff ID, position, and salary.

4. **Reservation**:
    - Attributes: Id, checkIn, checkOut
    - Represents the reservation details including reservation ID, check-in, and check-out dates. It is linked to guests and rooms.
5. **Payment**:
    - Attributes: paymentID, amount, date, method
    - Manages the payment transactions with attributes such as payment ID, amount, date, and method. It is linked to guest transactions.
6. **Room**:
    - Attributes: roomNo, type, price, availability
    - Represents a hotel room with attributes for room number, type, price, and availability. It is associated with reservations.
7. **Hotel**:
    - Attributes: Name, Location, Rating
    - Captures the overall details of the hotel, including its name, location, and rating. It contains rooms and employs staff.
8. **Service**:
    - Attributes: name, type, price
    - Represents various services offered by the hotel, detailing the service name, type, and price.
9. **ServiceRequest**:
    - Attributes: status, reqDate
    - Manages service requests made by guests, including the request status and date. It is linked to services.

Relationships:

- **Person** can be a **Guest** or **Staff**.

- A **Guest** can have one or more **Reservations**.
- Each **Reservation** involves one **Payment**.
- A **Reservation** is for one **Room**.
- A **Hotel** contains multiple **Rooms**.
- A **Hotel** employs multiple **Staff** members.
- A **Staff** member manages one **Hotel**.
- A **Hotel** offers multiple **Services**.
- A **Guest** can request multiple **Services** via **ServiceRequest**.

State Diagram

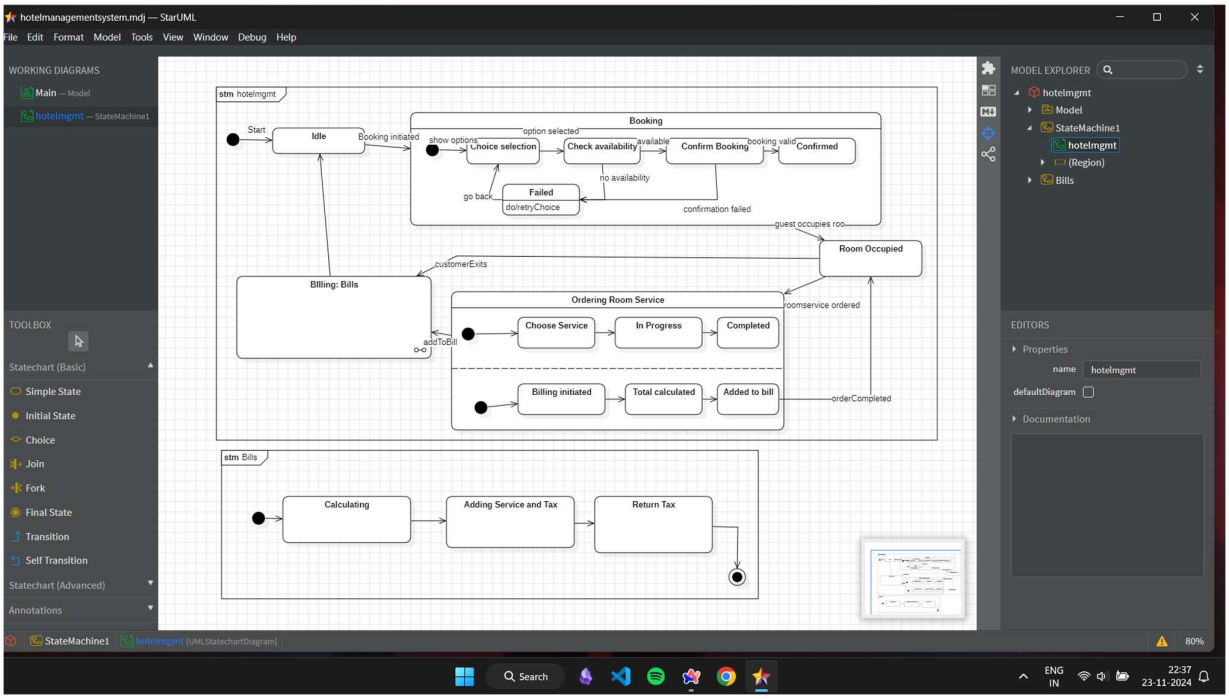


Figure 1.3: State Diagram

1. Booking:
    - States: Idle, Choice Selection, Check Availability, Confirm Booking, Confirmed, Failed.
    - Transitions: Booking starts from Idle, progresses through choice selection and availability checks, and ends in either Confirmed or Failed.
2. Room Occupied:
    - State: Represents the guest occupying the room after a successful booking.
    - Trigger: Booking is confirmed, and the guest checks in.
3. Ordering Room Service:
    - States: Choose Service, In Process, Completed.
    - Transitions: Customers choose a service, it gets processed, and then completed, with the cost added to the bill.
4. Billing:
    - States: Billing Initiated, Total Calculated, Added to Bill.
    - Transitions: Begins when a customer exits or orders additional services, calculates the total, and updates the bill.
5. Bill Calculation:

- States: Calculating, Adding Service and Tax, Return Tax.
- Transitions: Starts with calculating charges, adds applicable taxes or services, and adjusts for refunds if needed.

Relationships depict:

- Bookings transitioning through various states until confirmation or failure.
- Room occupancy dependent on booking confirmation.
- Room service ordered by guests affects the billing process.
- Billing integrates calculations for room charges, services, and taxes.

Use - Case Diagram

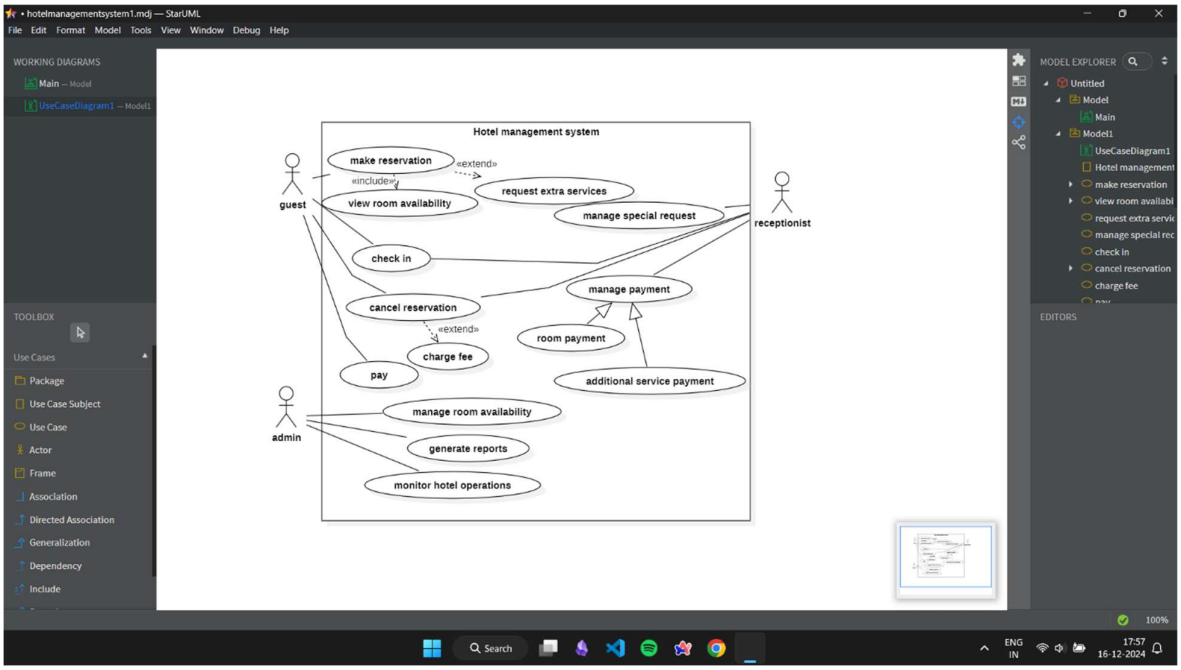


Figure 1.4: Use Case Diagram

• Guest:

Use Cases:

- make reservation: Guests can make reservations for hotel rooms.
- view room availability: Guests can check the availability of rooms.
- check in: Guests can check into their reserved rooms.
- cancel reservation: Guests can cancel their existing reservations.

- pay: Guests can make payments for their reservations and services.
- request extra services: Guests can request additional services during their stay.

- Receptionist:

  Use Cases:

  - manage room availability: Receptionists can manage the availability of rooms.
  - check in: Receptionists assist guests with the check-in process.
  - generate reports: Receptionists can generate various reports related to hotel operations.
  - manage special request: Receptionists handle any special requests made by guests.
  - manage payment: Receptionists process payments made by guests.

- Admin:

  Use Cases:

  - monitor hotel operations: Admins oversee and monitor overall hotel operations.
  - generate reports: Admins can also generate reports for analysis and decision-making.

Sequence Diagram

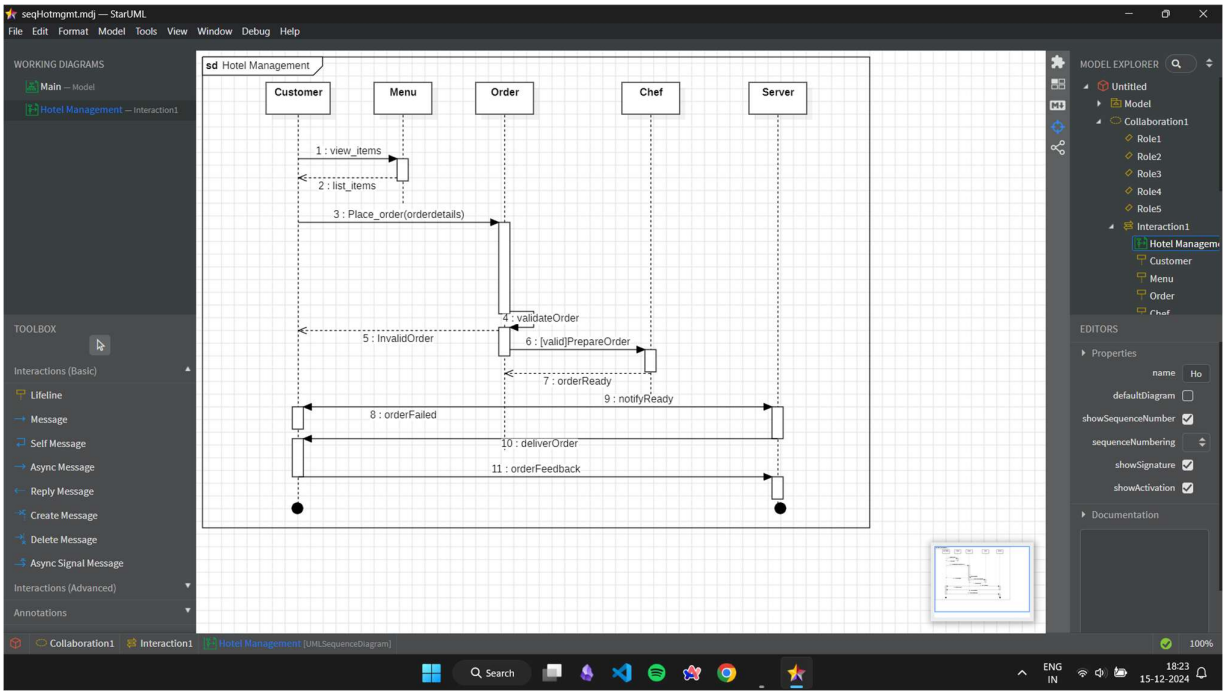


Figure 1.5: Sequence Diagram

Actors/Entities:

1. Customer: Initiates the process by interacting with the system to place an order.
2. Menu: Provides item details to the customer.
3. Order: Handles order validation and management.
4. Chef: Prepares the order after validation.
5. Server: Delivers the prepared order to the customer and manages notifications.

Interactions and Processes:

1. Customer views items (view_items):
    - The customer interacts with the menu to view available items.
    - Menu responds with the item_list.
2. Customer places an order (placeOrder):
    - The customer sends order details to the Order system.

3. Order validation (validateOrder):
    - The Order system validates the order details.
    - If invalid, the invalidOrder method is triggered, and the customer is notified of a failure (orderFailedNotify).
    - If valid, the system proceeds to preparation.
4. Order preparation (prepareOrder):
    - The validated order is sent to the Chef.
    - The preparation takes 15 to 20 minutes (indicated with a timing note).
5. Order readiness (orderReady):
    - The Chef notifies the Order system when the preparation is complete.
6. Notification to Server (notifyReady):
    - The Order system notifies the Server about the order's readiness.
7. Order delivery (deliverOrder):
    - The Server delivers the order to the customer.
8. Customer feedback (orderFeedback):
    - The customer provides feedback on the order.

Activity Diagram

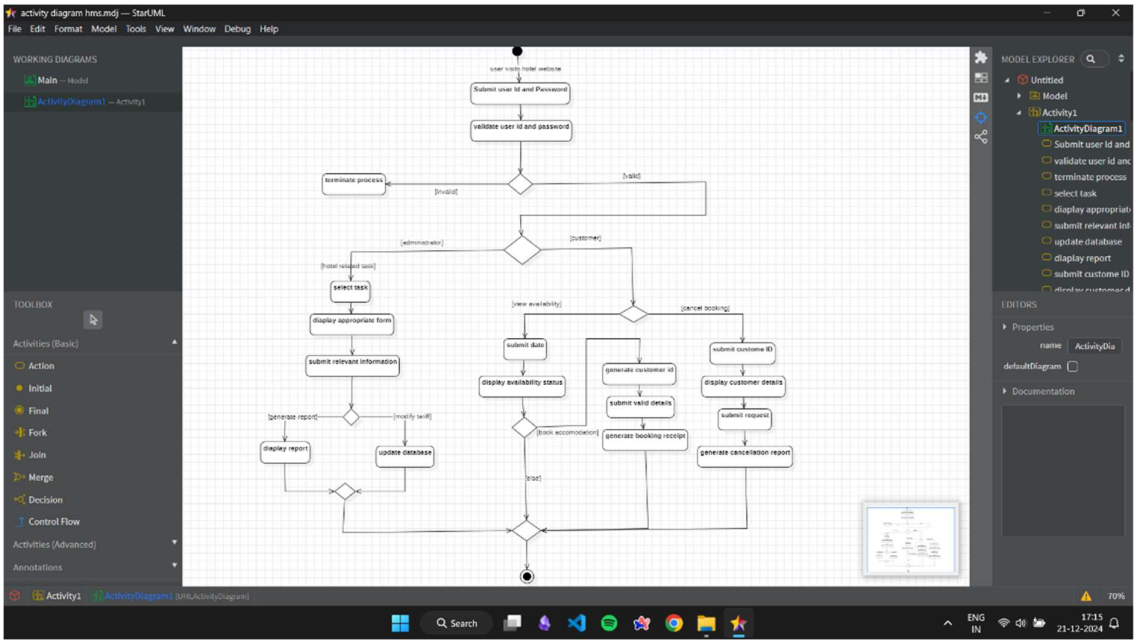


Figure 1.6: Activity Diagram

1. View Items:
    - The Customer interacts with the Menu to view available items (view_items()).
    - The Menu responds with the list of items (item_list()).
2. Place Order:
    - The Customer places an order by providing details (placeOrder(order details)).
3. Validate Order:
    - The Order object validates the received order (validateOrder()).
    - If the order is invalid, an error is triggered (invalidOrder()), and the customer is notified (orderFailedNotify()).
    - If the order is valid, the system proceeds to the next step.
4. Prepare Order:
    - The Order sends the request to the Chef to prepare the order (prepareOrder()).
    - The preparation process takes between 15–20 minutes.

5. Order Ready:
    - Once the preparation is complete, the Chef marks the order as ready (orderReady()).
    - The Order notifies the Server (notifyReady()).
6. Deliver Order:
    - The Server delivers the order to the Customer (deliverOrder()).
7. Feedback:
    - After receiving the order, the Customer provides feedback (orderFeedback()).

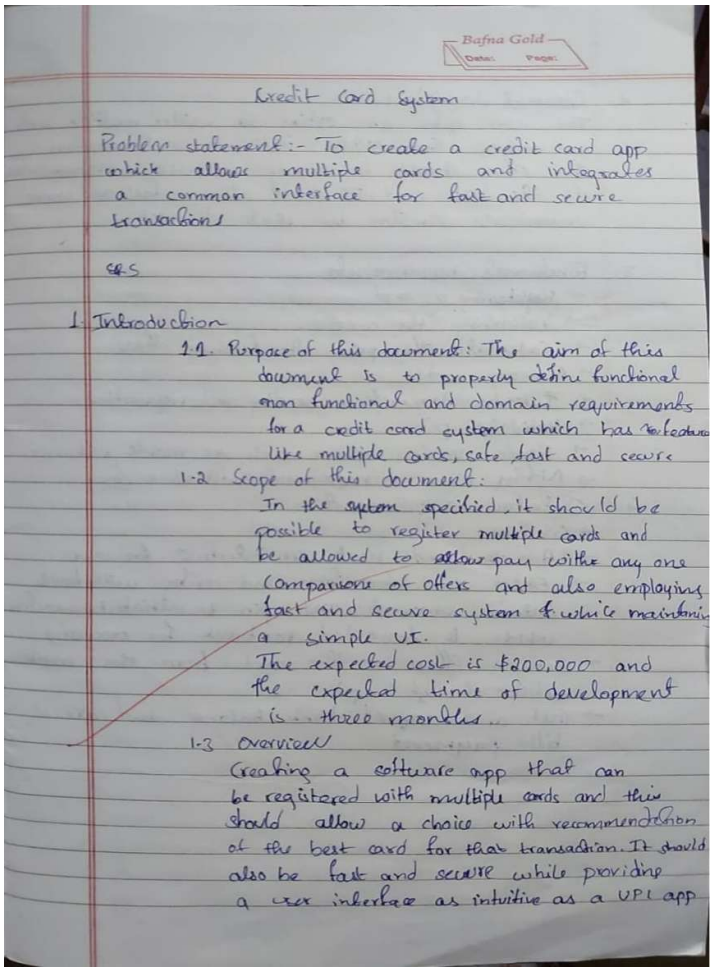
2. Credit Card Processing

Problem Statement

The growing reliance on credit cards for transactions demands a robust, secure, and efficient system for processing payments. Traditional credit card processing systems often face challenges such as slow transaction speeds, security vulnerabilities, limited support for multiple payment methods, and difficulty in handling disputes or errors. These issues lead to customer dissatisfaction, financial losses, and non-compliance with industry standards like PCI DSS (Payment Card Industry Data Security Standard).

Additionally, businesses need real-time insights into their transactions, seamless integration with existing accounting systems, and support for modern payment technologies like mobile wallets and contactless payments.

SRS – Software Requirements Specification



Bafna Gold

Credit Card System

Problem statement:- To create a credit card app which allows multiple cards and integrates a common interface for fast and secure transactions

SRS

1. Introduction

1.1. Purpose of this document: The aim of this document is to properly define functional non functional and domain requirements for a credit card system which has features like multiple cards, safe, fast and secure

1.2 Scope of this document:
In the system specified, it should be possible to register multiple cards and be allowed to select pay with any one comparisons of offers and also employing fast and secure system & while maintaining a simple UI.
The expected cost is ₹200,000 and the expected time of development is three months.

1.3 Overview
Creating a software app that can be registered with multiple cards and this should allow a choice with recommendation of the best card for that transaction. It should also be fast and secure while providing a user interface as intuitive as a UPI app

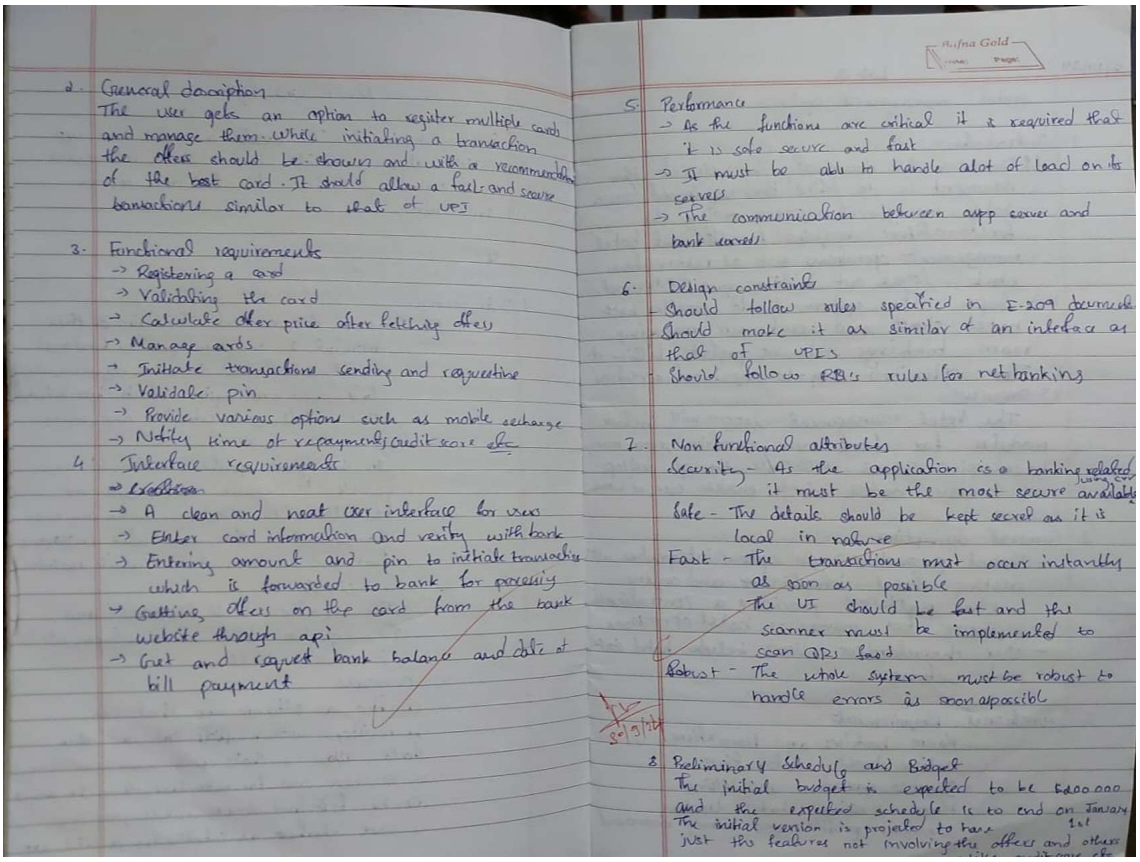


Figure 2.1: SRS Credit Card Processing

Class Diagram

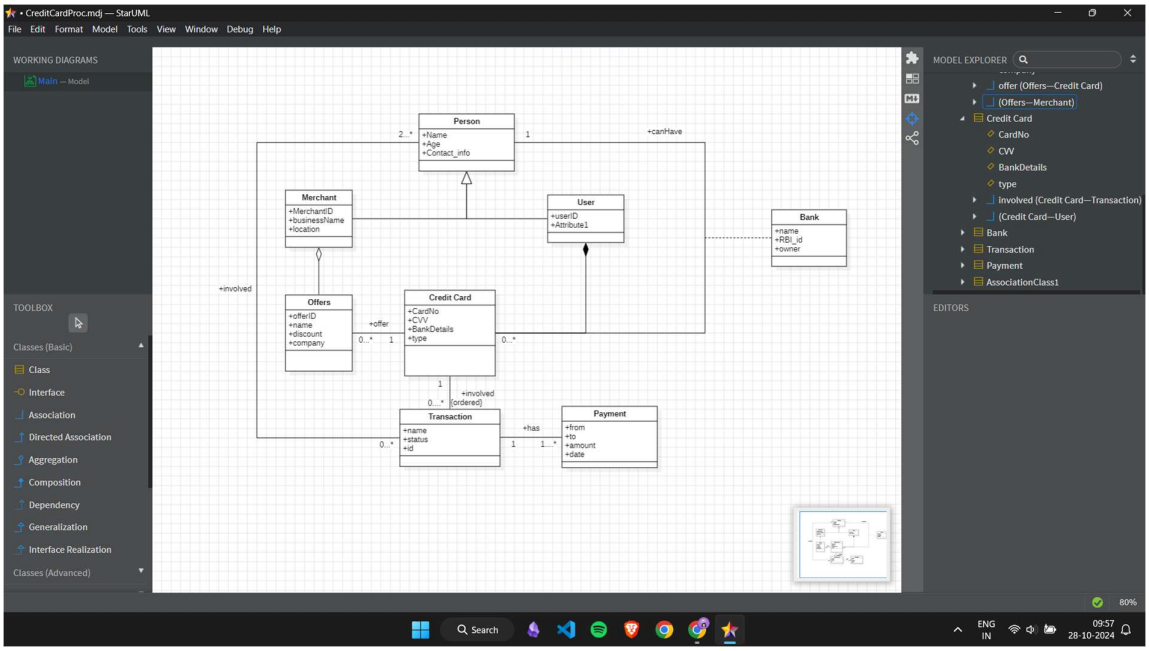


Figure 2.2: Class Diagram

1. **Person**:

   The Person class represents a general entity within the credit card processing system, containing attributes such as Name, Age, and Contact_info. This class serves as a base class for more specific entities.

2. **User**:

   The User class specializes the Person class to represent individuals who use the credit card system. It includes attributes like userID and other specific details, inheriting general attributes from Person.

3. **Merchant**:

   The Merchant class represents businesses that accept credit card payments. It includes attributes such as MerchantID, businessName, and location, enabling the system to track and manage different merchants.

4. **Offers**:

   The Offers class represents promotional deals available through the credit card system. Attributes like offerID, name, discount, and company detail various offers available to users and merchants.

5. **Credit Card**:

   The Credit Card class encapsulates the details of the credit cards used in transactions. Attributes such as CardNo, CVV, BankDetails, and type provide essential information for processing credit card transactions.

6. **Transaction**:

   The Transaction class records individual credit card transactions. It includes attributes like name, status, and id, tracking the details and state of each transaction.

7. **Payment**:

   The Payment class details the flow of money in transactions. Attributes such as from, to, amount, and date track who made the payment, who received it, the transaction amount, and the date of the transaction.

8. **Bank**:

   The Bank class represents the banks involved in the credit card processing system. It includes attributes like name, RBI_id, and owner, capturing the essential details about the banks that issue and manage credit cards.

Associations:

- A Person can have one User.
- A User can possess zero to many Credit Cards.
- A Credit Card can be involved in zero to many Transactions.
- Each Transaction has one Payment.
- A Merchant can have zero to many Offers.
- A Credit Card can have zero to one Offer.
- Bank is associated with Credit Card.

State Diagram

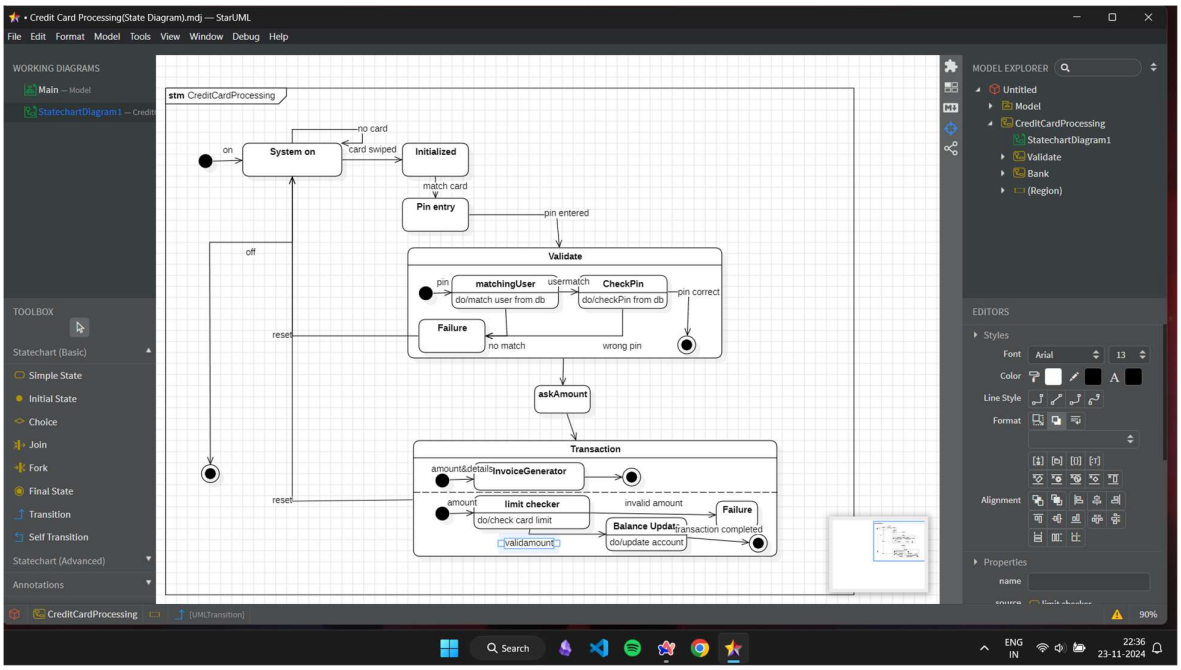


Figure 2.3: State Diagram

States:

- Initialized: The system starts here.
- Validate: Card and PIN are checked.
- Transaction: Amount is entered and processed.
- Failure: Errors occur during processing.

Transitions:

- The system moves from Initialized to Validate upon card insertion.
- If validation succeeds, it transitions to Transaction.
- If validation fails, it goes to Failure.
- Transaction can also lead to Failure if issues arise.
- The system can be reset from any state.

Actions:

- Card matching, PIN verification, amount input, and balance updates are performed during the process.

Use - Case Diagram

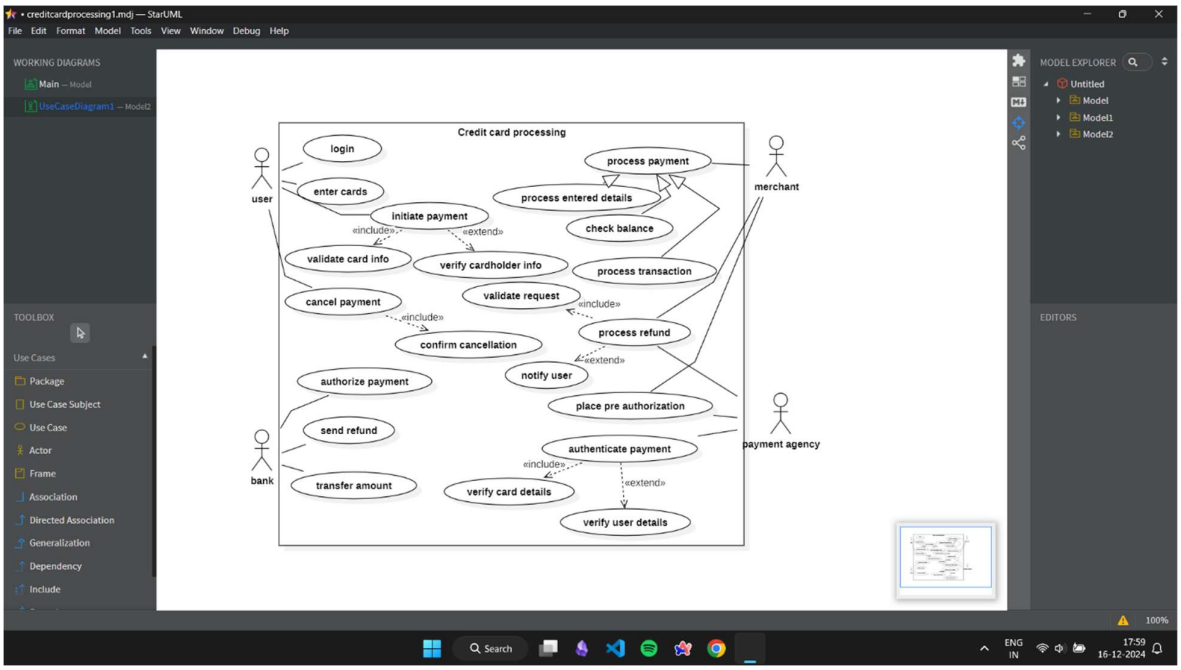


Figure 2.4: Use Case Diagram

Use Cases:

Login: Users log into the system.

Enter cards: Users enter their card details into the system.

Initiate payment: Users start the payment process.

Validate card info: The system checks the validity of the card information.

Cancel payment: Users cancel an ongoing payment.

Authorize payment: The system approves the payment.

Send refund: The system processes refunds for users.

Transfer amount: The system transfers the payment amount.

Process payment: Merchants handle the payment transaction.

Process entered details: The system processes the card details entered by users.

Check balance: Merchants check the account balance.

Verify cardholder info: The system verifies the information of the cardholder.

Validate request: The system validates the transaction request.

Process transaction: The system handles the payment transaction.

Process refund: The system processes the refund request.

Notify user: Users are notified about the status of their transactions.

Place pre-authorization: The payment agency places a pre-authorization hold.

Authenticate payment: The payment agency verifies the payment.

Verify user details: The payment agency verifies the user's details.

Confirm cancellation: The system confirms the cancellation of a payment.

Verify card details: The system checks the details of the credit card.

Sequence Diagram

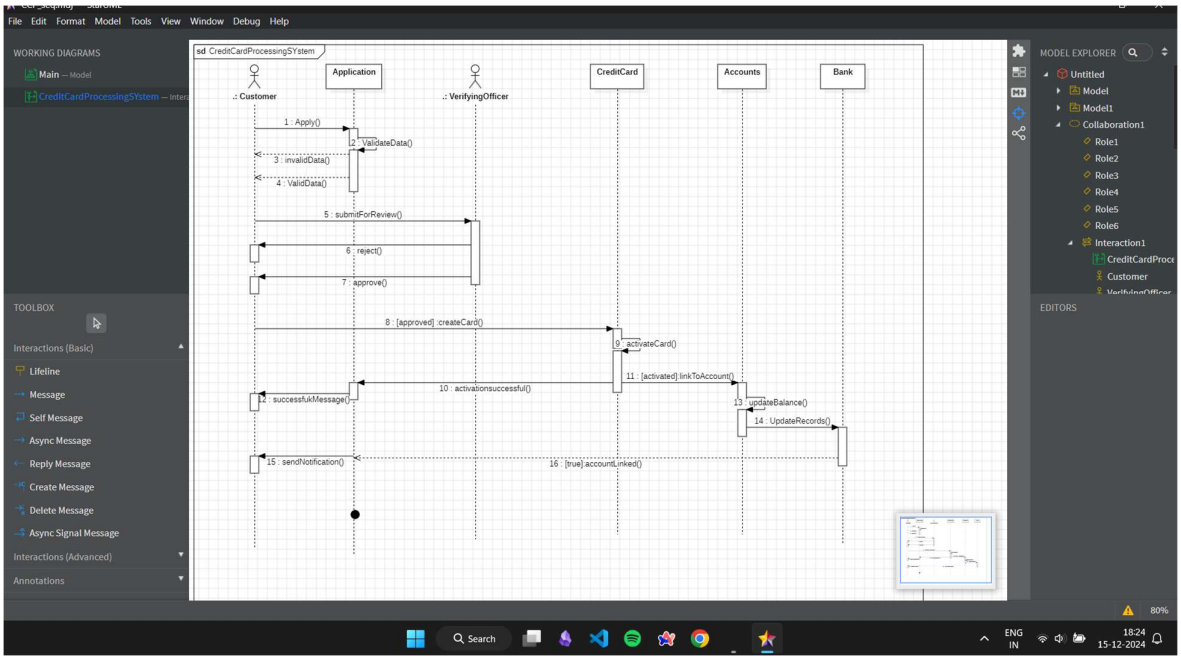


Figure 2.5: Sequence Diagram

1. Customer applies(): The customer initiates the process by applying for a credit card.
2. validateData(): The application object validates the data provided by the customer.
3. invalidData(): If the data is invalid, an error message is sent to the customer.
4. validData(): If the data is valid, the application is submitted for review.
5. submitForReview(): The application object submits the application to the verifying officer.
6. reject(): If the verifying officer rejects the application, the customer is notified.
7. approve(): If the verifying officer approves the application, a credit card is created.
8. createCard(): The credit card object is created.
9. activateCard(): The credit card is activated.
10. activationSuccessful(): The customer is notified of the successful activation.
11. SuccessfulMessage(): A success message is displayed to the customer.
12. linkToAccount(): The credit card is linked to the customer's account.
13. updateBalance(): The customer's account balance is updated.
14. updateRecords(): The bank's records are updated.
15. sendNotification(): A notification is sent to the customer.
16. accountLinked(): The system confirms that the credit card has been linked to the account.

Activity Diagram

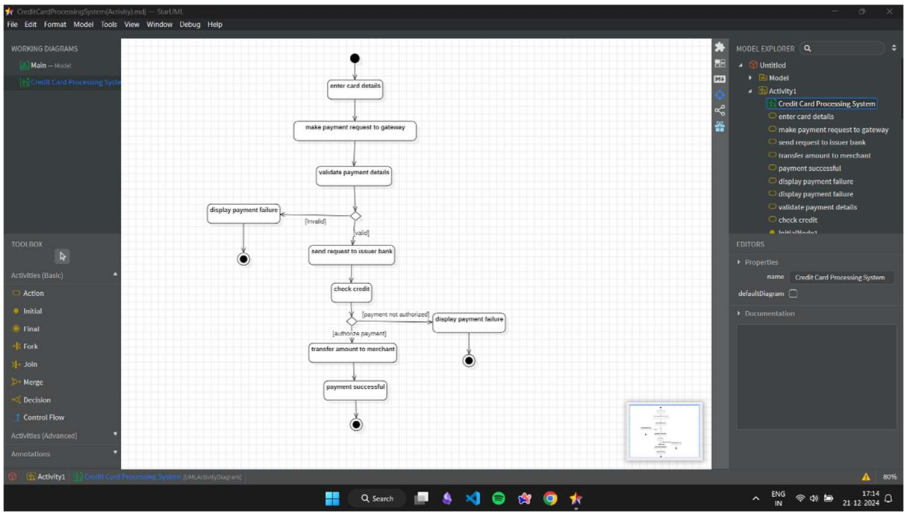


Figure 2.6: Activity Diagram

1. Enter Card Details: The user enters their card details, such as card number, expiration date, and CVV.
2. Make Payment Request to Gateway: The system sends a payment request to the payment gateway.
3. Validate Payment Details: The payment gateway validates the entered card details.
    - Invalid: If the details are invalid, the process ends with a "payment failure" message.
    - Valid: If the details are valid, the process proceeds to the next step.
4. Send Request to Issuer Bank: The payment gateway sends a request to the issuer bank to authorize the transaction.
5. Check Credit: The issuer bank checks the user's credit limit and account balance.
    - Payment Not Authorized: If the payment is not authorized (e.g., insufficient funds), the process ends with a "payment failure" message.
    - Authorize Payment: If the payment is authorized, the process proceeds to the next step.
6. Transfer Amount to Merchant: The issuer bank transfers the payment amount to the merchant's account.
7. Payment Successful: The payment process is completed successfully.

3. Library Management System

Problem Statement

Traditional library management systems often rely on manual processes, leading to inefficiencies in tracking books, managing member records, and handling transactions such as book borrowing and returns. These methods result in time-consuming tasks, data inaccuracies, limited accessibility for users, and challenges in generating meaningful insights for library staff. A lack of integration with digital technologies further hampers the library's ability to meet modern user expectations for convenience and efficiency.

SRS – Software Requirements Specification

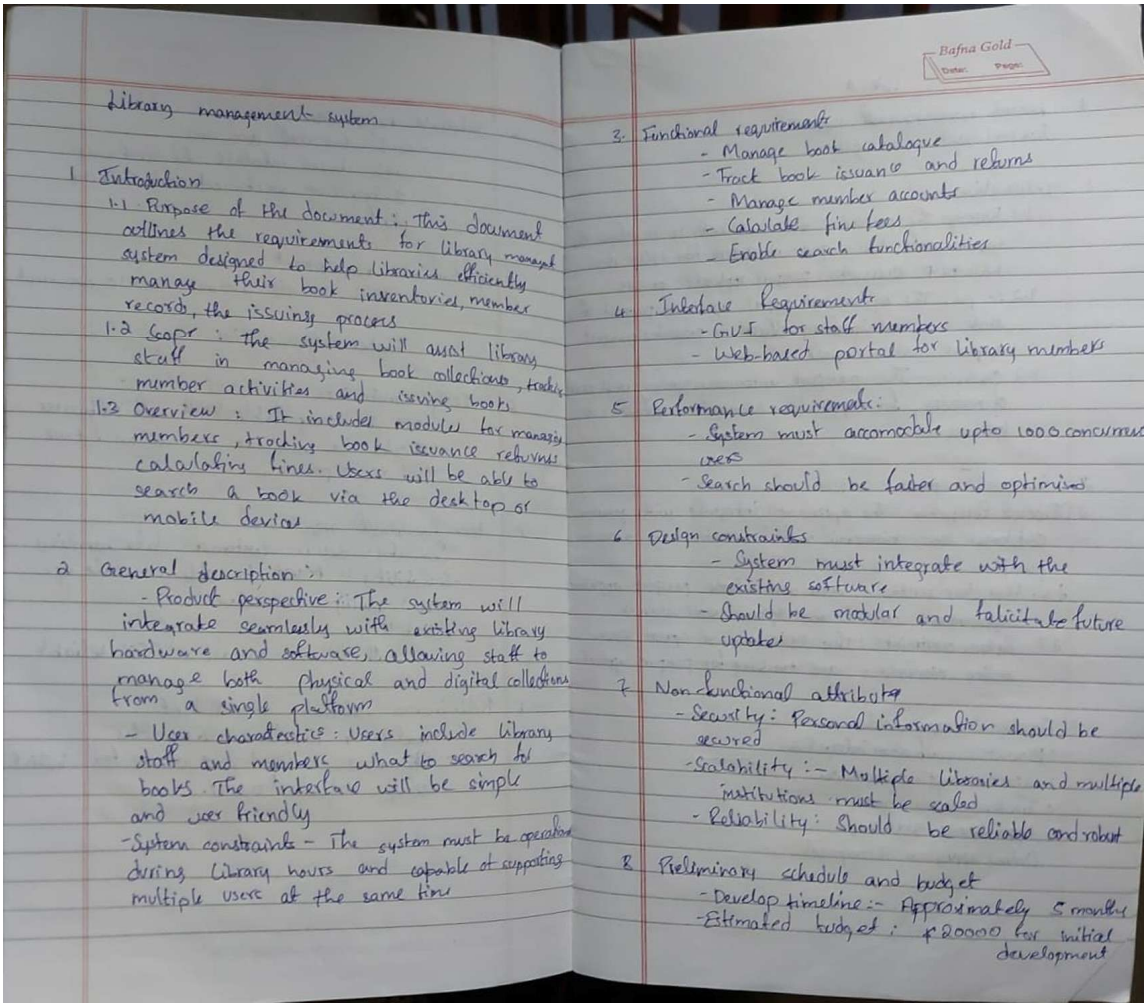


Figure 3.1: SRS Library Management System

Class Diagram

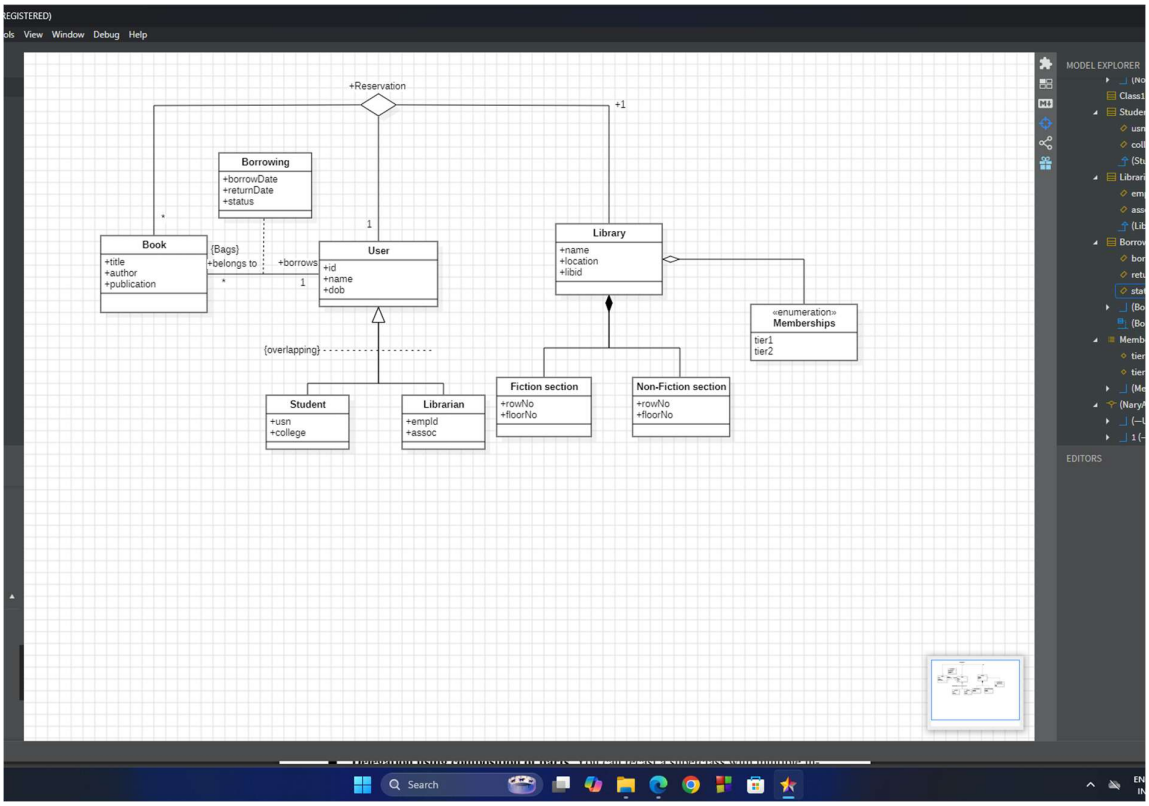


Figure 3.2: Class Diagram

1. **Book**:
    - **Attributes**: title, author, publication.
    - Represents a book in the library with details like the title, author, and publication information.
2. **User**:
    - **Attributes**: id, name, dob.
    - Represents a general user of the library, with attributes for user identification, name, and date of birth.
3. **Library**:
    - **Attributes**: name, location, libid.
    - Represents the library itself, including its name, location, and a unique library ID.
4. **Student**:
    - **Attributes**: usn, college.

- Inherits from the User class, adding attributes specific to student users such as the university serial number (USN) and college.

5. **Librarian**:
   - **Attributes**: empId, assoc.
   - Inherits from the User class, with attributes specific to librarians like employee ID and association.
6. **Fiction section**:
   - **Attributes**: rowNo, floorNo.
   - Represents a section within the library dedicated to fiction books, with attributes for row and floor numbers.
7. **Non-Fiction section**:
   - **Attributes**: rowNo, floorNo.
   - Represents a section within the library for non-fiction books, with similar attributes for row and floor numbers as the fiction section.
8. **Borrowing**:
   - **Attributes**: borrowDate, returnDate, status.
   - Manages the borrowing process of books by users, capturing the borrowing and return dates, and the status of the borrowing process.
9. **Memberships**:
   - **Enumeration Values**: tier1, tier2.
   - Represents different membership tiers within the library system.

Relationships:

- A Book can be borrowed by multiple Users.
- A User can borrow multiple Books.
- A Library has multiple sections (Fiction section and Non-Fiction section).
- A User can make a reservation at a Library.
- Memberships (Memberships) are associated with Libraries

State Diagram

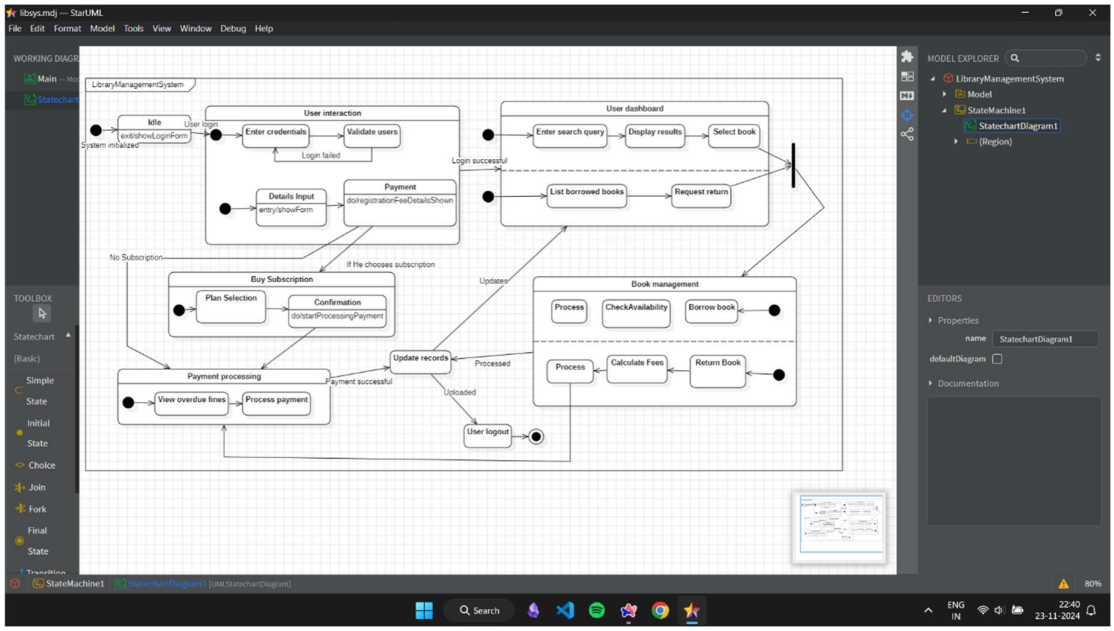


Figure 3.3: State Diagram

- User Interaction:
  - User Login: The user enters their credentials to log into the system.
  - Validate User: The system validates the user's credentials.
  - Enter Search Query: The user enters a search query to find books.
  - Display Results: The system displays the search results to the user.
  - Select Book: The user selects a book from the results.
  - List Borrowed Books: The system displays a list of books borrowed by the user.
  - Request Return: The user requests to return a borrowed book.
  - User Logout: The user logs out of the system.
- Book Management:
  - Borrow Book: The system processes the borrowing of a book by a user.
  - Return Book: The system processes the return of a book by a user.
- Payment Processing:
  - View Overdue Fines: The system displays the overdue fines for the user.
  - Process Payment: The system processes the payment of overdue fines.

Relationships:

- The diagram shows a sequential flow of activities, with some activities branching based on conditions (e.g., successful login, valid payment).
- The "Action Completed" state represents the end of a successful transaction.

Use - Case Diagram

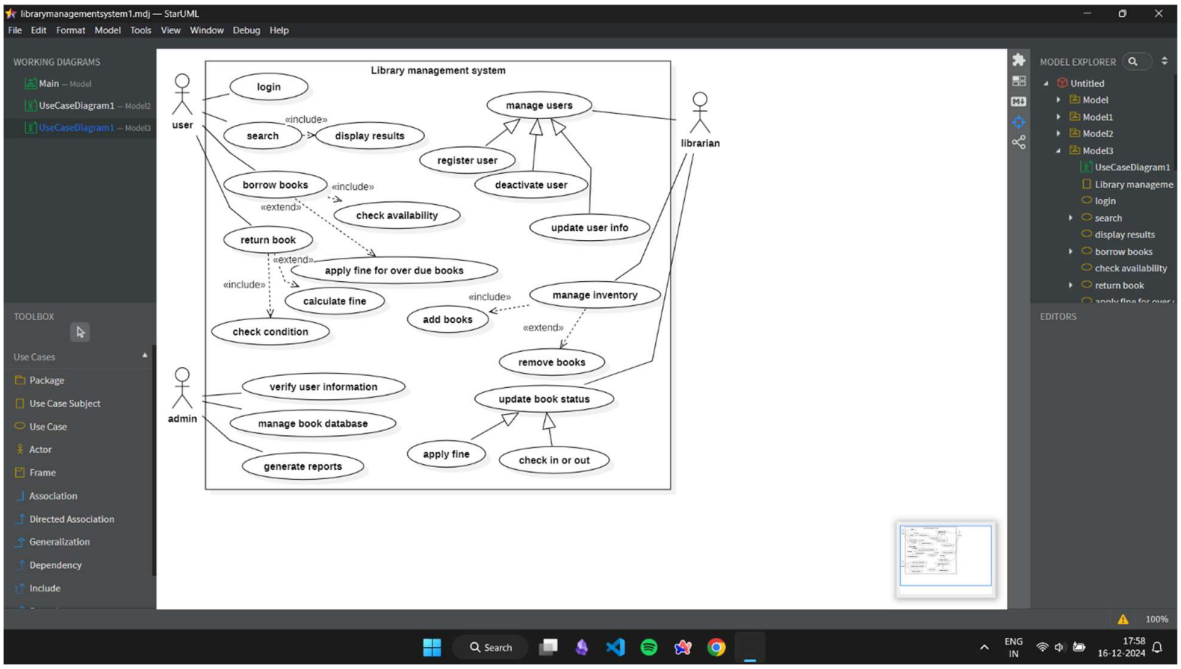


Figure 3.4: Use Case Diagram

Use Cases:

- Login: The process of authenticating a user to access the system.
- Search: Enables users to search for books in the library catalog.
- Display Results: Presents the search results to the user.
- Borrow Books: The process of issuing books to users.
- Return Book: The process of returning books to the library.

- Check Availability: Verifies the availability of a book before issuing.
- Apply Fine for Overdue Books: Calculates and applies fines for overdue books.
- Calculate Fine: Determines the amount of fine based on the overdue period.
- Check Condition: Checks the condition of returned books.
- Register User: Creates new user accounts in the system.
- Deactivate User: Deactivates user accounts.
- Update User Info: Updates user information, such as contact details.
- Manage Inventory: Manages the library's book collection.
- Add Books: Adds new books to the library catalog.
- Remove Books: Removes books from the library catalog.
- Update Book Status: Updates the status of books (e.g., available, checked out, reserved).
- Apply Fine: Applies fines to users for overdue books.
- Check In/Out: Processes book check-in and check-out operations.
- Verify User Information: Verifies the accuracy of user information.
- Manage Book Database: Performs administrative tasks related to the book database.
- Generate Reports: Generates various reports, such as usage statistics and overdue book reports.

Relationships:

- Include: Indicates that one use case includes another use case as a sub-step. For example, "Search" includes "Display Results."
- Extend: Indicates that a use case can be extended with optional behavior. For example, "Borrow Books" can be extended with "Check Availability."

Sequence Diagram

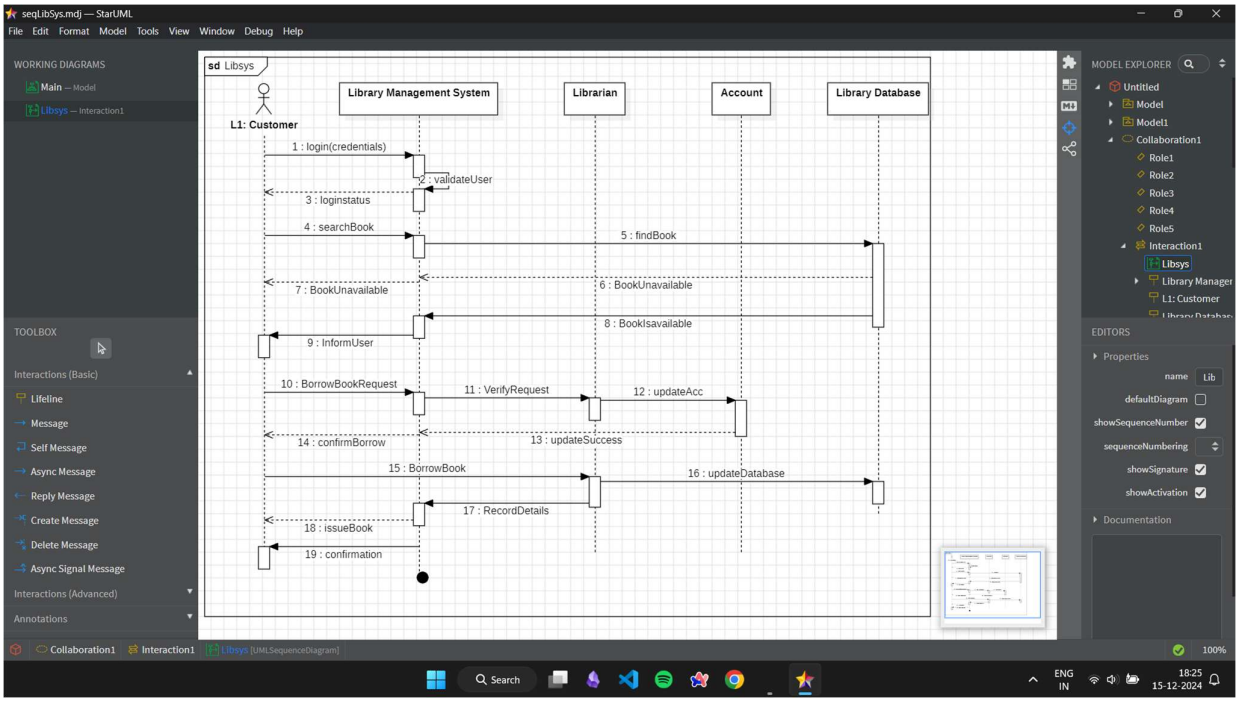


Figure 3.5: Sequence Diagram

1. **login(credentials):** The customer initiates the process by logging into the system with their credentials.
2. **validateUser():** The Library Management System validates the customer's credentials.
3. **loginFailed():** If the credentials are invalid, an error message is sent to the customer.
4. **loginSuccess():** If the credentials are valid, the customer is successfully logged in.
5. **searchBook():** The customer searches for a book in the system.
6. **findBook():** The Library Management System searches for the requested book in the Library Database.
7. **bookUnavailableNotification():** If the book is not available, a notification is sent to the customer.
8. **bookIsAvailable():** If the book is available, the system proceeds to the next step.
9. **bookAvailableNotification():** A notification is sent to the customer indicating that the book is available.
10. **borrowBookRequest():** The customer requests to borrow the available book.
11. **verifyRequest():** The Library Management System verifies the borrowing request.
12. **updateAccount():** The system updates the customer's account to reflect the borrowed book.

13. **borrowConfirmation():** A confirmation message is sent to the customer.
14. **borrowBook():** The Library Management System records the book as borrowed.
15. **updateIssuedBook():** The Library Database updates the status of the book to "issued."
16. **bookRecorded():** The system records the borrowing transaction in the Library Database.
17. **confirmationNotification():** A confirmation notification is sent to the customer.

Activity Diagram

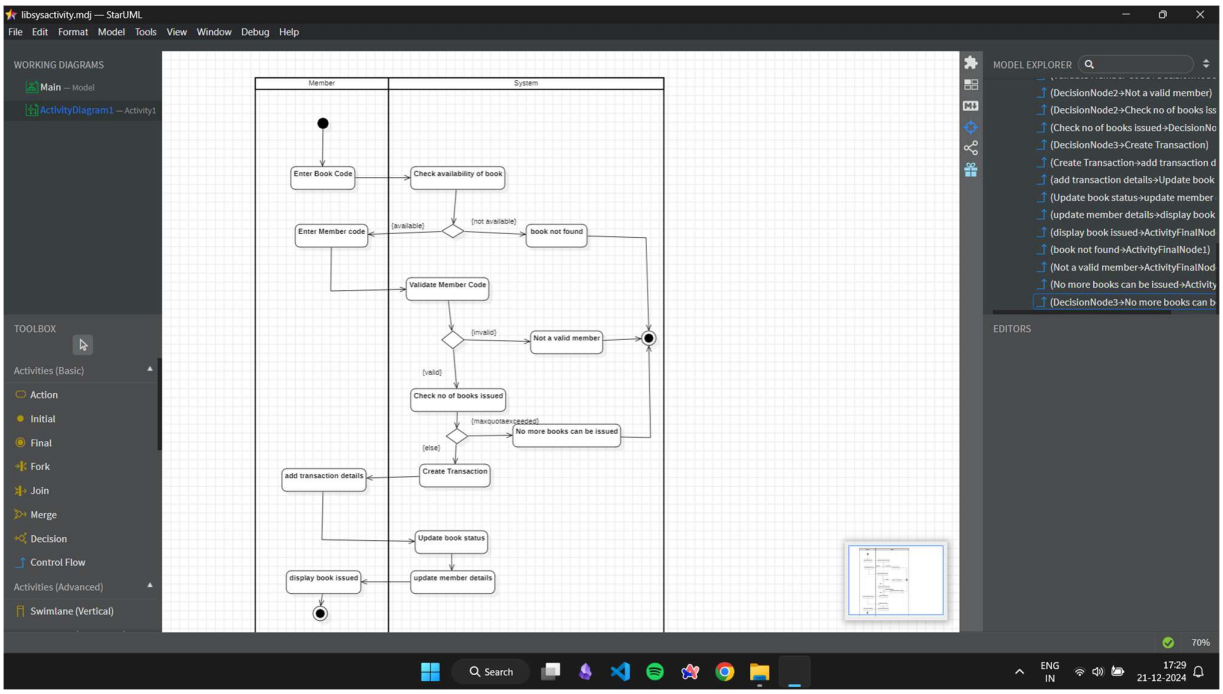


Figure 3.6: Activity Diagram

1. Enter Book Code: The user enters the code of the book they wish to borrow.
2. Check Availability of Book: The system checks if the book is available for borrowing.
    - Book Available: If the book is available, the process proceeds to the next step.
    - Book Not Found: If the book is not found in the system, an error message is displayed.
3. Enter Member Code: The user enters their member code.
4. Validate Member Code: The system validates the entered member code.
    - Valid Number: If the member code is valid, the process proceeds to the next step.
    - Not a Valid Number: If the member code is invalid, an error message is displayed.

5. Check No of Book Issued to Member: The system checks if the member has reached the maximum number of books allowed to borrow.
    - Max Quote Exceeded: If the member has exceeded the borrowing limit, an error message is displayed.
    - Within Limit: If the member is within the borrowing limit, the process proceeds to the next step.
6. Create Transaction: A new transaction is created in the system to record the book borrowing.
7. Add Transaction Details: Details of the transaction, such as book code, member code, and issue date, are added to the transaction record.
8. Update Book Status: The status of the book is updated in the system to "issued."
9. Update Member Details: The member's borrowing history is updated in the system to reflect the new book.
10. Display "Book Issued": A success message is displayed to the user, indicating that the book has been successfully issued.

4. Stock Maintenance System

Problem Statement

Effective stock management is critical for businesses to ensure optimal inventory levels, reduce wastage, and meet customer demands. However, traditional manual or semi-automated methods of stock maintenance often result in challenges such as inaccurate inventory tracking, overstocking or understocking, delayed replenishments, and inefficient reporting. These issues can lead to increased operational costs, missed sales opportunities, and poor decision-making due to a lack of real-time data and analytics.

SRS – Software Requirements Specification

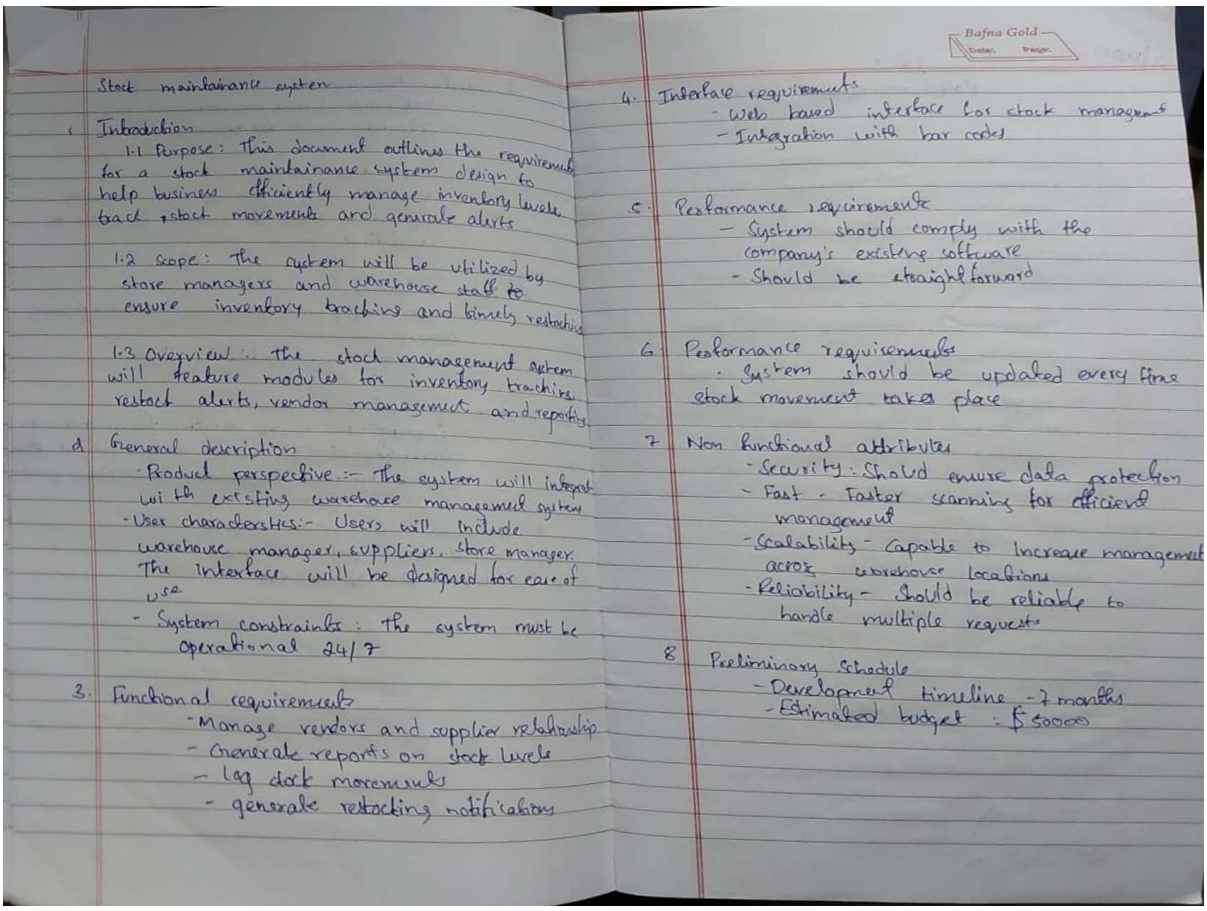


Figure 4.1: SRS Stock Maintenance System

Class Diagram

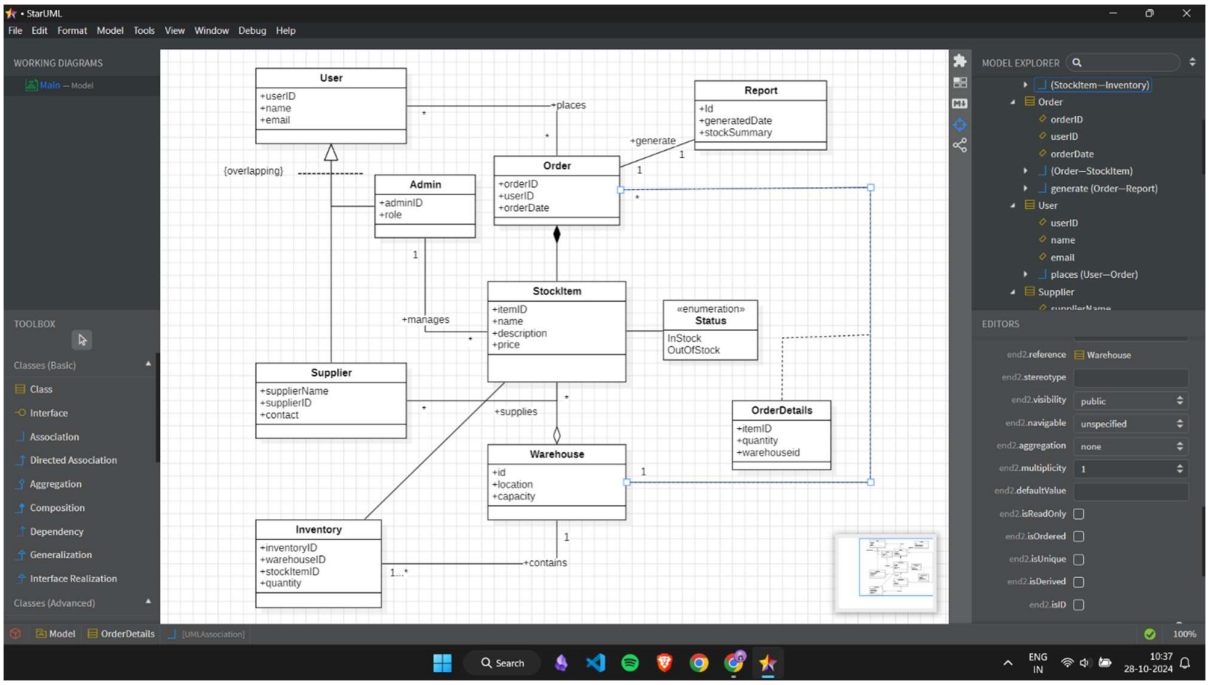


Figure 4.2: Class Diagram

1. **User**:
    - **Attributes**: userID, name, contactDetails.
    - Represents a general user of the system, with attributes for identification and contact information.
2. **Admin**:
    - **Attributes**: adminID, role.
    - Inherits from the User class, adding attributes specific to administrative users such as adminID and role in the system.
3. **Supplier**:
    - **Attributes**: supplierID, companyName, contactInfo.
    - Represents suppliers providing inventory, with unique identifiers and contact information for each supplier.
4. **Inventory**:

- **Attributes**: inventoryID, description, quantity.
- Represents the overall inventory, capturing details like descriptions and quantities of items.

5. **StockItem**:
    - **Attributes**: stockItemID, name, category, status.
    - Represents individual stock items within the inventory, including their unique ID, name, category, and status (e.g., InStock or OutOfStock).
6. **Warehouse**:
    - **Attributes**: warehouseID, location, capacity.
    - Represents different warehouses storing inventory, with details about location and storage capacity.
7. **Order**:
    - **Attributes**: orderID, orderDate, totalAmount.
    - Represents orders placed within the system, capturing essential details like order ID, date, and total amount.
8. **OrderDetails**:
    - **Attributes**: orderDetailID, quantity, price.
    - Represents details of each order, including quantities and prices of ordered items, linking back to Order.
9. **Report**:
    - **Attributes**: reportID, generatedDate, content.
    - Represents reports generated within the system, capturing details about the report's ID, generation date, and content.

Enumeration:

- **Status**:
    - Values: InStock, OutOfStock.

- Represents the status of stock items.

Relationships:

- A User can be an Admin.
- A Supplier supplies StockItems.
- StockItems are part of the Inventory.
- Warehouses store StockItems.
- An Order consists of multiple OrderDetails.
- Reports are generated to analyze inventory, orders, and other system components.

State Diagram

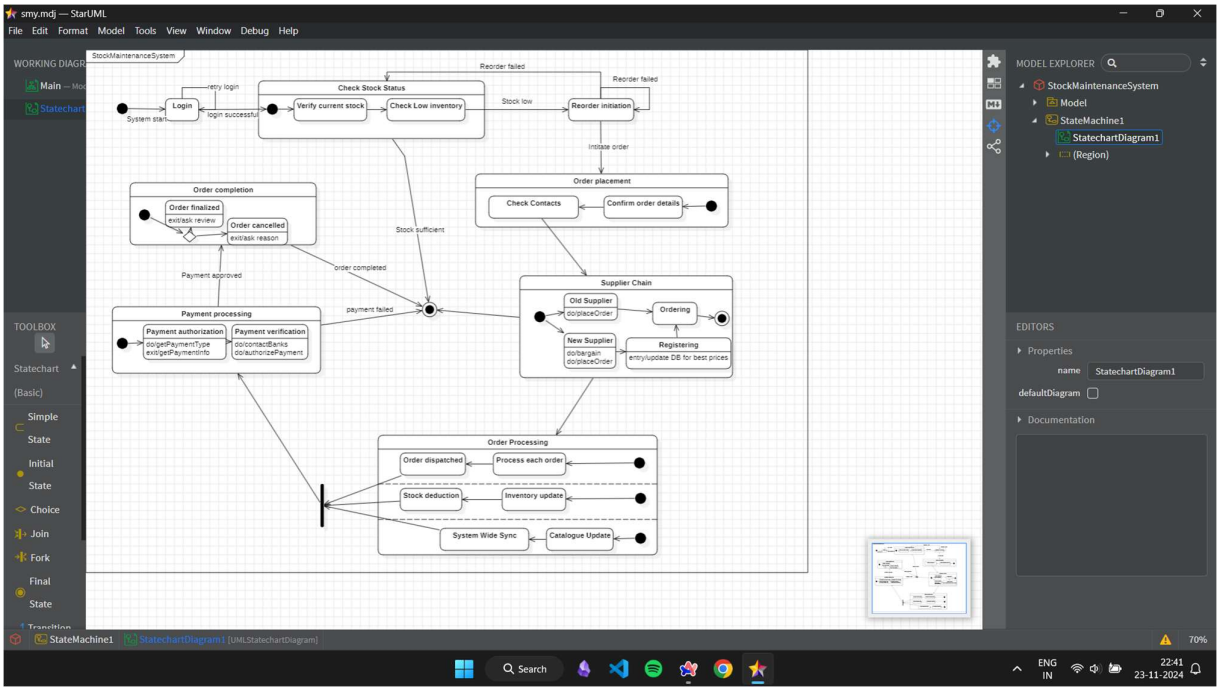


Figure 4.3: State Diagram

1. **System Start:** The initial state of the system.
2. **Login:** The state where the user attempts to log in to the system.
3. **Login Success:** The state reached after successful user authentication.
4. **Check Stock Status:** The state where the system verifies the current stock levels.
5. **Stock Sufficient:** The state indicating that stock levels are sufficient.
6. **Stock Low:** The state indicating that stock levels are below the reorder point.
7. **Reorder Initiation:** The state where the system initiates a reorder request.
8. **Order Placement:** The state where the order is being placed with the supplier.
9. **Order Confirmed:** The state indicating that the order has been confirmed by the supplier.
10. **Order Processing:** The state where the supplier is processing the order.
11. **Order Dispatched:** The state indicating that the order has been dispatched by the supplier.
12. **Stock Deduction:** The state where the system deducts the ordered quantity from the stock.
13. **Inventory Update:** The state where the inventory levels are updated to reflect the received order.
14. **Order Completed:** The final state indicating that the entire order process has been completed successfully.
15. **Order Cancelled:** The state indicating that the order has been canceled.

16. **Payment Processing:** The state where the payment for the order is being processed.
17. **Payment Authorization:** The state where the payment for the order is being authorized.
18. **Payment Verification:** The state where the payment for the order is being verified.
19. **Payment Error:** The state indicating that a payment error has occurred.
20. **Reorder Failed:** The state indicating that the reorder process has failed.

Use - Case Diagram

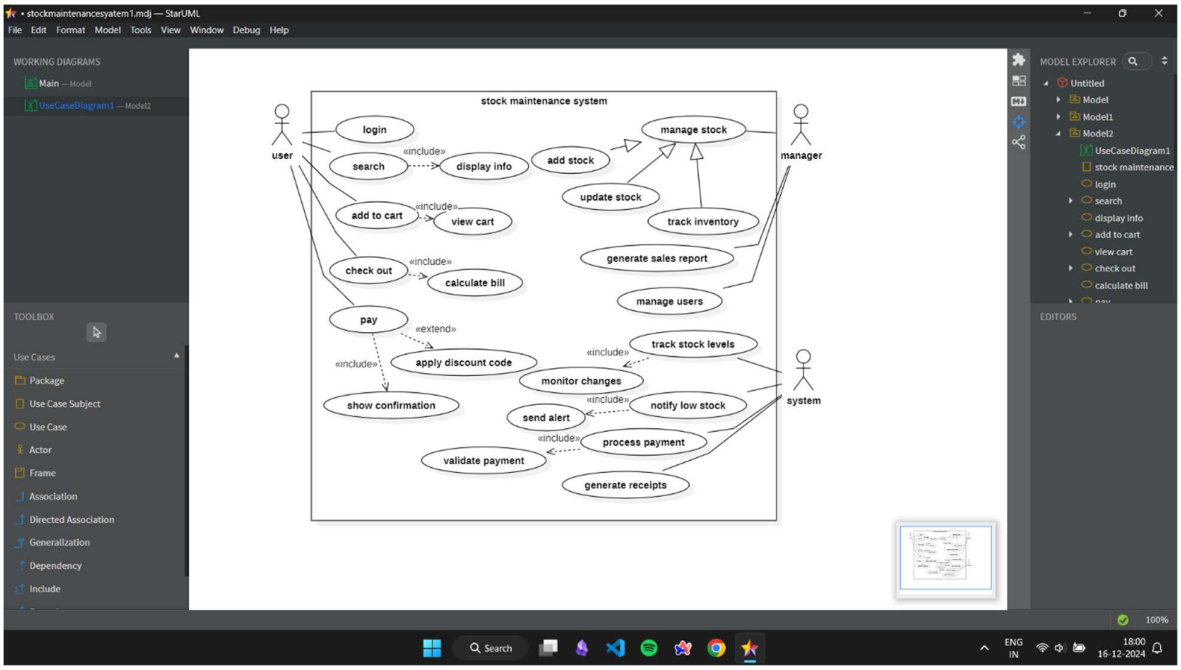


Figure 4.4: Use Case Diagram

Use Cases:

- **Login:** The process of authenticating a user to access the system.
- **Search:** Enables users to search for products in the inventory.
- **Display Info:** Displays detailed information about a product.
- **Add to Cart:** Allows users to add products to their shopping cart.
- **View Cart:** Displays the contents of the user's shopping cart.
- **Check Out:** The process of initiating the checkout process.
- **Calculate Bill:** Calculates the total cost of items in the shopping cart.

- **Pay:** The process of making a payment for the order.
- **Apply Discount Code:** Allows users to apply discount codes to their orders.
- **Show Confirmation:** Displays a confirmation message after a successful order.
- **Manage Stock:** Includes use cases for adding, updating, and tracking stock levels.
- **Add Stock:** Allows managers to add new products to the inventory.
- **Update Stock:** Allows managers to update product information and stock levels.
- **Track Inventory:** Monitors stock levels and generates reports.
- **Generate Sales Report:** Generates reports on sales data.
- **Manage Users:** Allows managers to manage user accounts (e.g., create, edit, delete).
- **Track Stock Levels:** Monitors stock levels and notifies relevant personnel when stock is low.
- **Notify Low Stock:** Sends alerts when stock levels fall below a certain threshold.
- **Monitor Changes:** Monitors changes in stock levels and other relevant data.
- **Process Payment:** Processes payment transactions.
- **Validate Payment:** Validates payment information.
- **Generate Receipts:** Generates receipts for completed orders.

Sequence Diagram

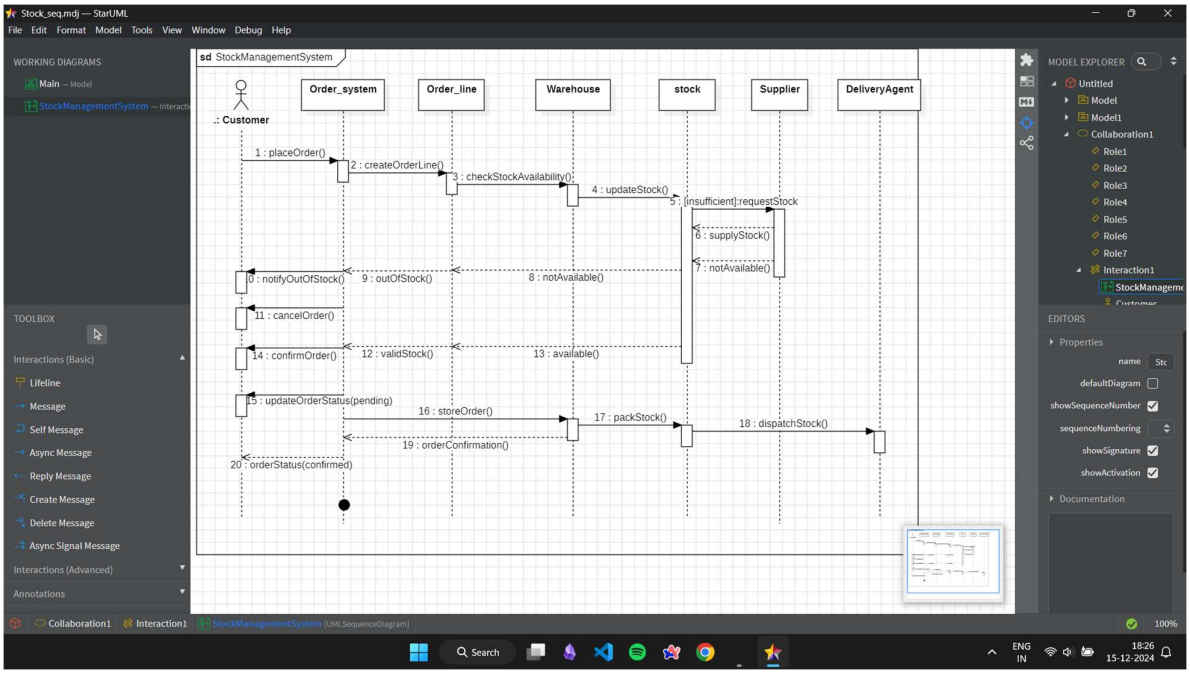


Figure 4.5: Sequence Diagram

1. placeOrder(): The customer initiates the process by placing an order.
2. createOrderLine(): The Order System creates an order line for each item in the order.
3. checkStockAvailability(): The Order Line checks the stock availability for each item in the warehouse.
4. updateStock(): The Warehouse updates the stock quantity based on the order line.
5. insufficient, requestStock(): If the stock is insufficient, the Warehouse requests more stock from the Supplier.
6. supplyStock(): The Supplier supplies the requested stock to the Warehouse.
7. notAvailable(): If the stock is not available, the system notifies the Customer.
8. notifyOutOfStock(): The Order System notifies the Customer about the out-of-stock items.
9. outOfStock(): The Order Line updates the status of the out-of-stock items.
10. notAvailable(): The Customer is notified about the unavailable items.
11. cancelOrder(): The Customer cancels the order.
12. confirmOrder(): The Order System confirms the order.
13. validStock(): The Order Line verifies that the stock is available for all items.
14. available(): The Warehouse confirms that the stock is available.
15. updateOrderStatus(pending): The Order System updates the order status to "pending."

16. storeOrder(): The Warehouse stores the order for processing.
17. packStock(): The Warehouse packs the ordered items for shipment.
18. sendOrderConfirmation(): The Order System sends an order confirmation to the Customer.
19. dispatchStock(): The Delivery Agent dispatches the order.
20. orderStatus(Confirmed): The Order System updates the order status to "Confirmed."

Activity Diagram

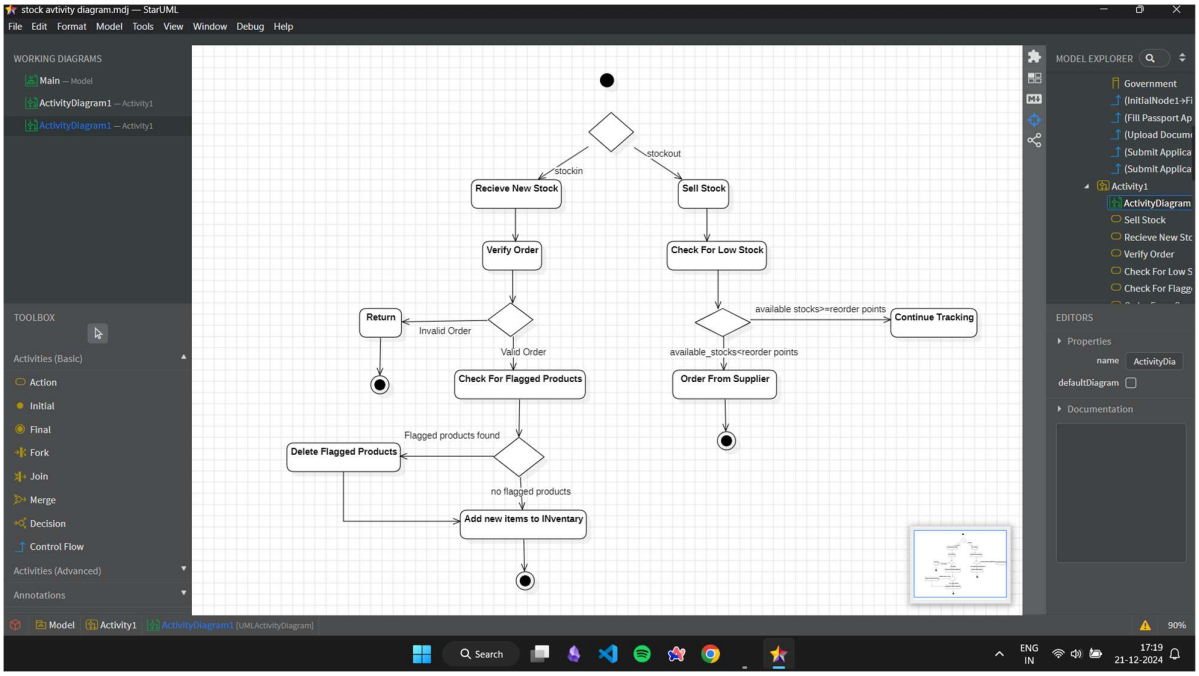


Figure 4.6: Activity Diagram

1. **Stock In/Out:** This is the initial state where the system can either receive new stock or sell stock.
2. **Receive New Stock:** This activity represents the process of receiving new stock into the inventory.
3. **Verify Order:** This activity checks the validity of an incoming order or a sales order.
4. **Check For Low Stock:** This activity checks if the stock levels of any products have fallen below the reorder point.
5. **Order From Supplier:** If stock levels are below the reorder point, this activity triggers an order to be placed with the supplier to replenish stock.

6. **Check For Flagged Products:** This activity checks if any products in the inventory have been flagged for removal or other actions.
7. **Delete Flagged Products:** If any flagged products are found, this activity removes them from the inventory.
8. **Add New Items to Inventory:** This activity adds new items to the inventory after receiving new stock or after removing flagged products.
9. **Sell Stock:** This activity represents the process of selling stock from the inventory.
10. **Continue Tracking:** This activity represents the ongoing monitoring and management of inventory levels.

11. 5. Passport Automation System

Problem Statement

The manual processing of passport applications is often time-consuming, prone to errors, and inefficient, leading to delays in issuance and renewal. Applicants face challenges such as lengthy queues, incomplete information, and lack of real-time updates on application status. Additionally, managing large volumes of data manually can result in errors, security concerns, and difficulty in tracking and retrieving records. These inefficiencies hinder the overall process and compromise user satisfaction and operational effectiveness.

SRS – Software Requirements Specification

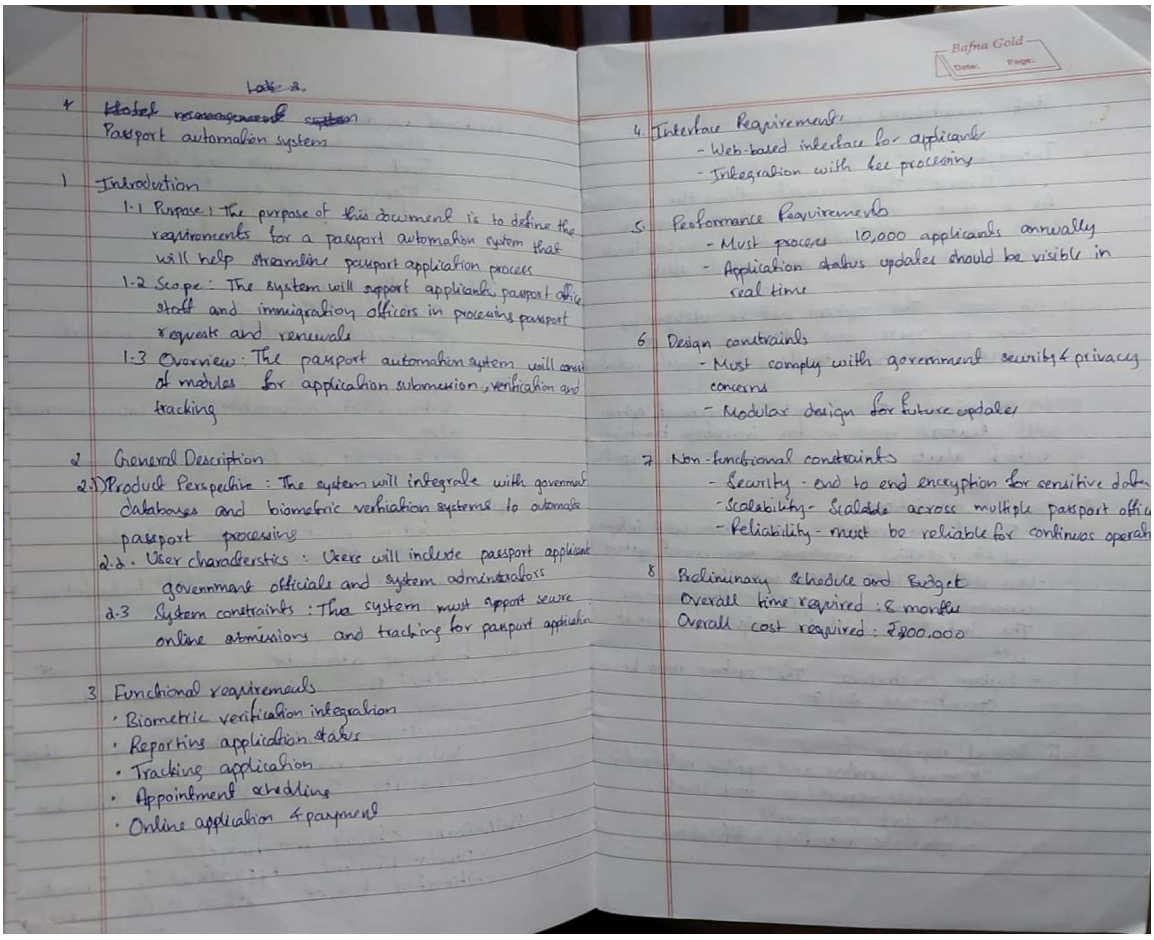


Figure 5.1: SRS Passport Automation

Class Diagram

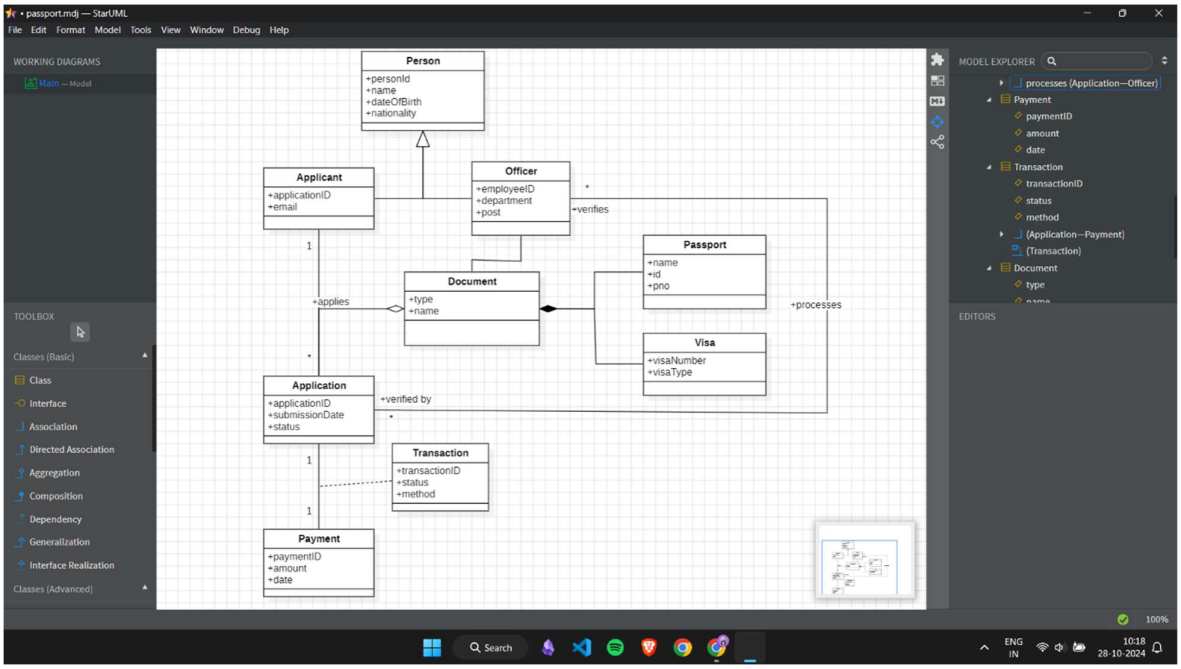


Figure 5.2: Class Diagram

1. **Person**:
    - **Attributes**: personId, name, dateOfBirth, nationality.
    - Represents a general person within the system, capturing essential details such as ID, name, date of birth, and nationality.
2. **Applicant**:
    - **Attributes**: applicationID, email.
    - Inherits from the Person class and includes additional attributes specific to applicants, like their application ID and email.
3. **Officer**:
    - **Attributes**: employeeID, department, post.
    - Represents officials within the system, including attributes like employee ID, department, and post. Officers verify applications and process documents.
4. **Document**:

- **Attributes**: type, name.
- Represents documents processed within the system, such as passports and visas, with attributes detailing the type and name of the document.

5. **Application**:
   - **Attributes**: applicationID, submissionDate, status.
   - Represents the applications submitted by applicants, capturing details such as application ID, submission date, and current status.
6. **Transaction**:
   - **Attributes**: transactionID, status, method.
   - Tracks financial transactions related to applications, including attributes like transaction ID, status, and payment method.
7. **Payment**:
   - **Attributes**: paymentID, amount, date.
   - Represents payments made within the system, capturing details such as payment ID, amount, and date.
8. **Passport**:
   - **Attributes**: name, id, pno.
   - A type of document representing passports, with attributes like name, ID, and passport number (pno).
9. **Visa**:
   - **Attributes**: visaNumber, visaType.
   - A type of document representing visas, including attributes such as visa number and visa type.

Relationships:

- An Applicant applies for an Application.
- An Application is verified by an Officer.

- An Application involves multiple Transactions.
- A Transaction involves a Payment.
- A Document is processed by an Officer.
- A Document can be a Passport or a Visa.

State Diagram

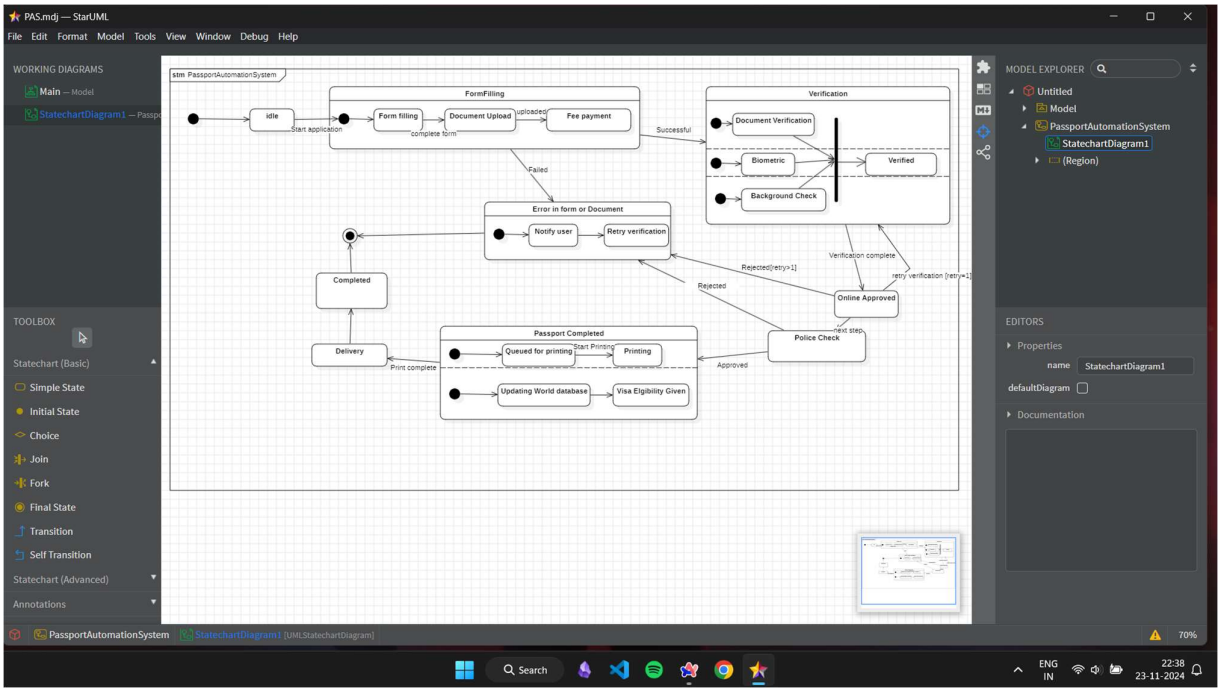


Figure 5.3: State Diagram

States:

- **Idle:** The initial state of the system.
- **Start Application:** The state where the application process begins.
- **Form Filling:** The state where the applicant is filling out the application form.
- **Document Upload:** The state where the applicant uploads supporting documents.
- **Fee Payment:** The state where the applicant pays the application fee.
- **Successful:** The state reached after successful submission of the application.

- **Document Verification:** The state where the submitted documents are being verified.
- **Biometric:** The state where biometric data (fingerprints, etc.) is being captured.
- **Background Check:** The state where the background check is being conducted.
- **Verification Complete:** The state reached after the verification process is complete.
- **Rejected:** The state indicating that the application has been rejected.
- **Approved:** The state indicating that the application has been approved.
- **Approval Decision:** The state where the decision to approve or reject the application is made.
- **Queued for Printing:** The state where the passport is queued for printing.
- **Start Printing:** The state where the passport printing process begins.
- **Printing:** The state where the passport is being printed.
- **Print Complete:** The state reached after the passport is printed.
- **Delivery:** The state where the passport is being delivered to the applicant.
- **Completed:** The final state indicating that the entire passport application process is complete.
- **Error Handling:** The state where error conditions are handled, such as failed payments or incomplete applications.

Use - Case Diagram

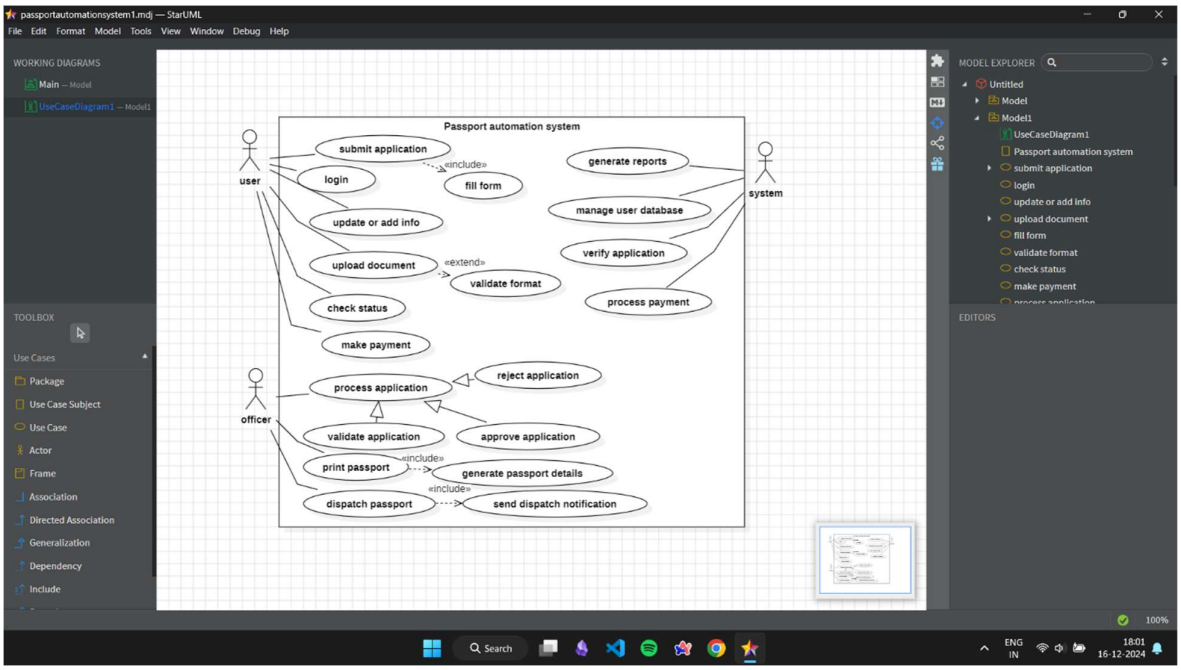


Figure 5.4: Use Case Diagram

Use Cases:

- Login: The process of authenticating a user to access the system.
- Submit Application: The process of submitting a passport application.
- Fill Form: The process of filling out the application form.
- Upload Document: The process of uploading supporting documents for the application.
- Check Status: Allows users to check the status of their application.
- Make Payment: The process of making the required payment for the application.
- Process Application: The process of reviewing and processing the application.
- Validate Application: The process of verifying the information and documents provided in the application.
- Approve Application: The process of approving an application.
- Reject Application: The process of rejecting an application.
- Print Passport: The process of printing the passport.

- Generate Passport Details: The process of generating the passport details based on the approved application.
- Dispatch Passport: The process of dispatching the passport to the applicant.
- Send Dispatch Notification: The process of sending a notification to the applicant regarding passport dispatch.
- Manage User Database: Includes tasks like adding, updating, and deleting user accounts.
- Generate Reports: Generates reports on various aspects of the passport application process.
- Verify Application: The process of verifying the authenticity of the application and supporting documents.

Sequence Diagram

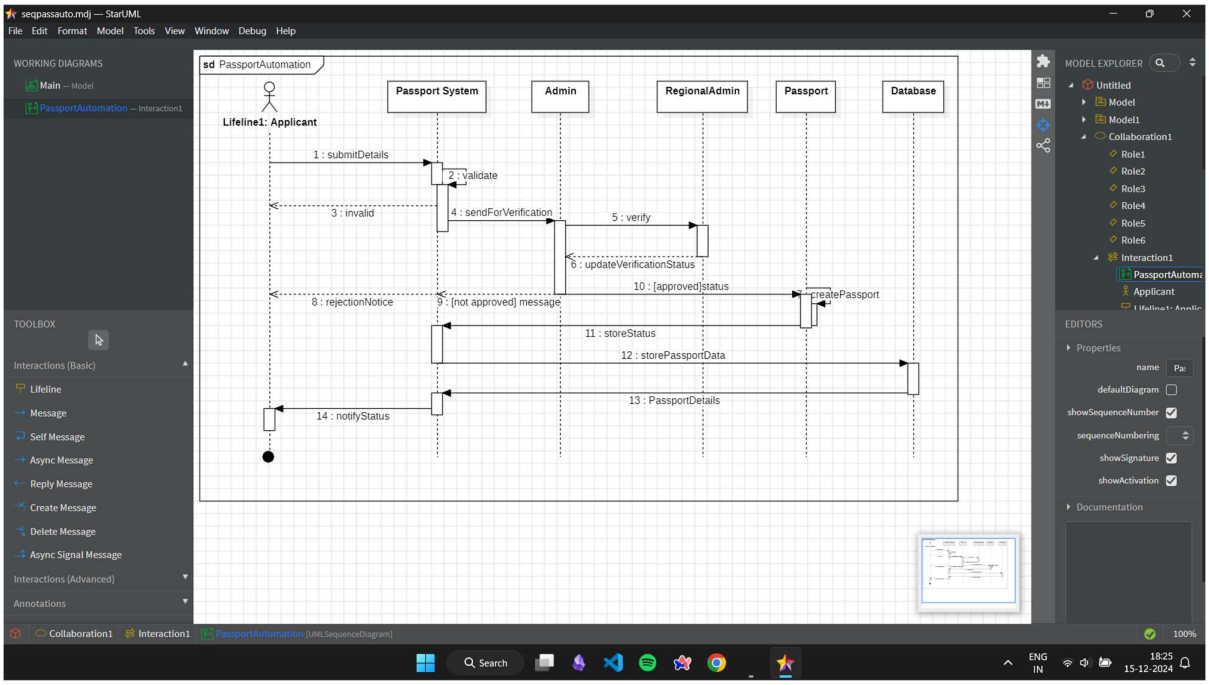


Figure 5.5: Sequence Diagram

1. submitDetails(): The Applicant initiates the process by submitting their application details to the PassportSystem.
2. validateDocs(): The PassportSystem validates the submitted documents.

3. invalidDocs(): If the documents are invalid, the PassportSystem sends an error message to the Applicant.
4. sendToVerification(): If the documents are valid, the PassportSystem sends the application to the PassportAdministrator for verification.
5. verifyApplication(): The PassportAdministrator verifies the application details.
6. updateVerificationStatus(): The PassportSystem updates the verification status of the application.
7. rejectApplication(): If the application is rejected, the PassportSystem sends a rejection notification to the Applicant.
8. approveApplication(): If the application is approved, the PassportSystem sends it to the RegionalAdministrator for further processing.
9. createPassport(): The PassportSystem creates a new Passport object.
10. storePassportData(): The PassportSystem stores the passport data in the Database.
11. storeApplication(): The PassportSystem stores the application details in the Database.
12. notifyApplicationStatus(): The PassportSystem notifies the Applicant about the status of their application.

Activity Diagram

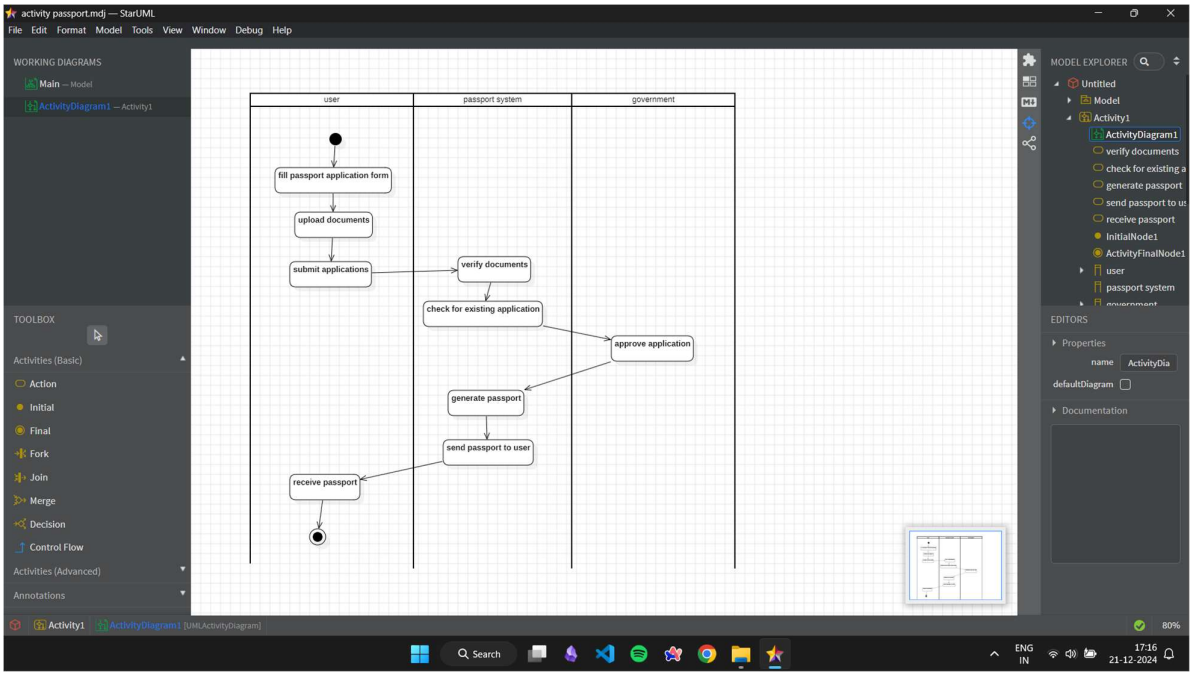


Figure 5.6: Activity Diagram

1. **Fill Passport Application:** The user starts by filling out the online passport application form.
2. **Upload Documents:** The user uploads the required supporting documents, such as proof of identity and address.
3. **Submit Application:** The user submits the completed application to the Passport System.
4. **Verify Document:** The Passport System verifies the authenticity of the uploaded documents.
5. **Check for Existing Application:** The system checks if the user already has an existing passport application.
6. **Approve Application:** If the documents are valid and no existing application is found, the Government authorities approve the application.
7. **Generate Passport:** The Passport System generates the passport.
8. **Send Passport to User:** The passport is sent to the user.
9. **Receive Passport:** The user receives the passport.