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

Lab Record

# Object-Oriented Modeling

*Submitted in partial fulfillment for the 5th Semester Laboratory*

Bachelor of Engineering
in
Computer Science and Engineering

*Submitted by:*

**SIDDHARTH H G**

1BM22CS276

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

# B.M.S. COLLEGE OF ENGINEERING

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


## *CERTIFICATE*

This is to certify that the Object Oriented Modelling (23CS5PCOOM) laboratory has been carried out by SIDDHARTH H G (1BM22CS276) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

Name of the Incharge: **RADHIKA A D**
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

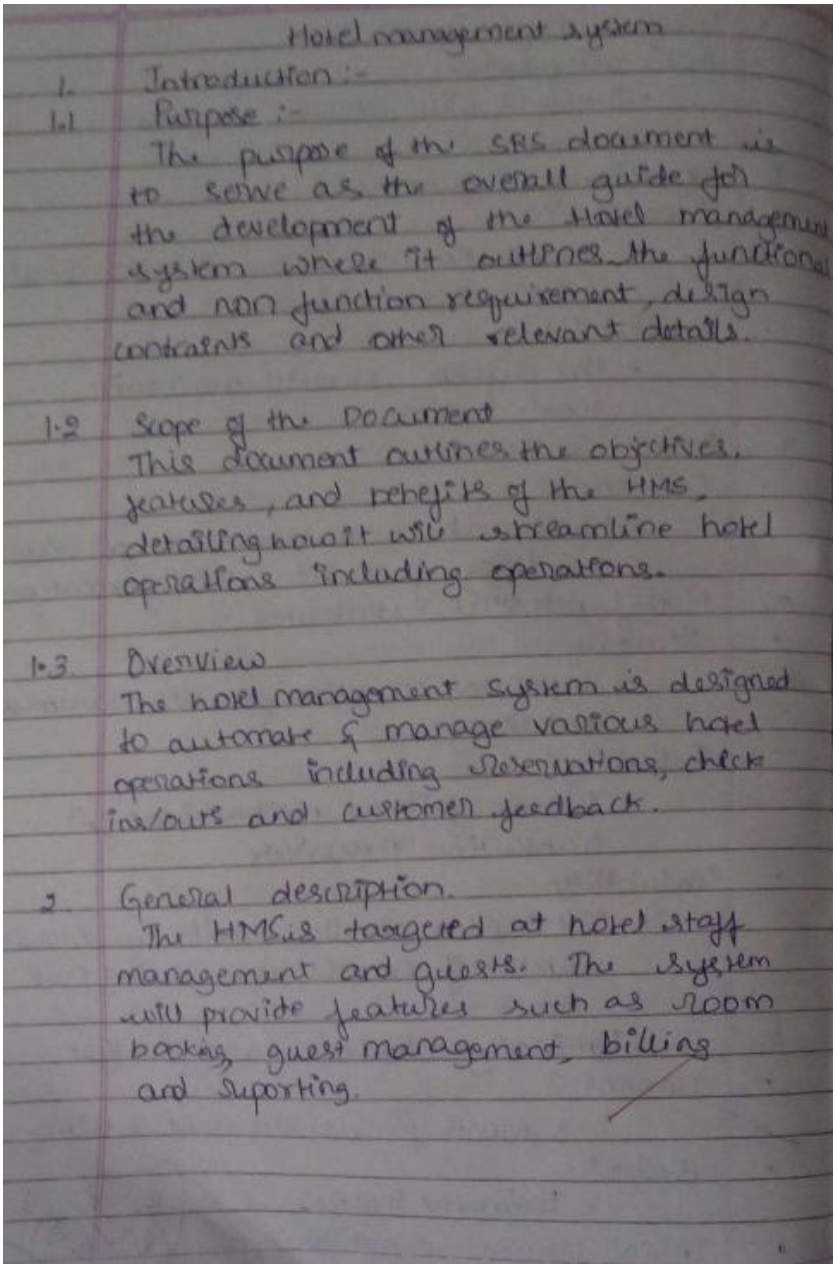
# 1. Hotel Management System

## 1.1. Problem Statement: Hotel Management System

## The Challenge:

Traditional hotel management often relies on manual processes and separate systems, leading to inefficiencies, errors, and a poor guest experience.

## 1.2. SRS Document:



Hotel management system

1. Introduction :-

1.1 Purpose :-
The purpose of the SRS document is to serve as the overall guide for the development of the Hotel management system where it outlines the functional and non function requirement, design constraints and other relevant details.

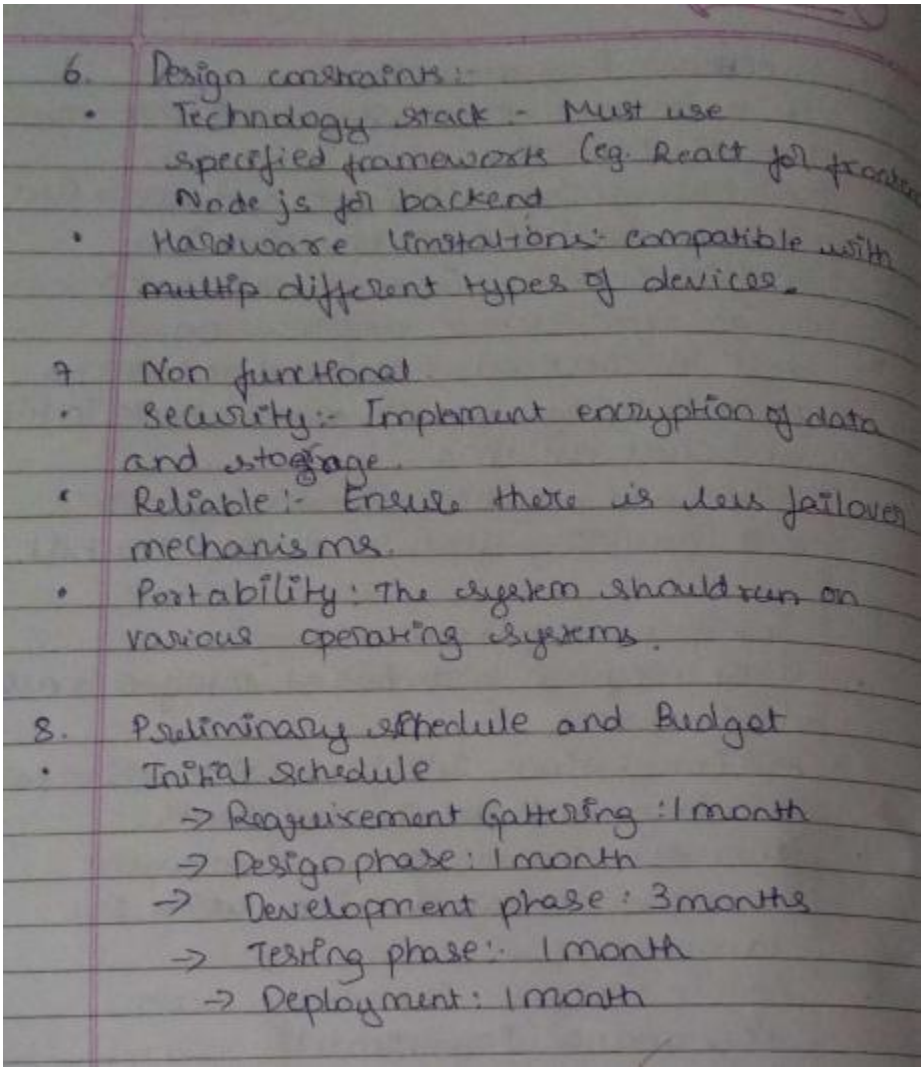
1.2 Scope of the Document
This document outlines the objectives, features, and benefits of the HMS, detailing how it will streamline hotel operations including operations.

1.3 Overview
The hotel management system is designed to automate & manage various hotel operations including reservations, check ins/outs and customer feedback.

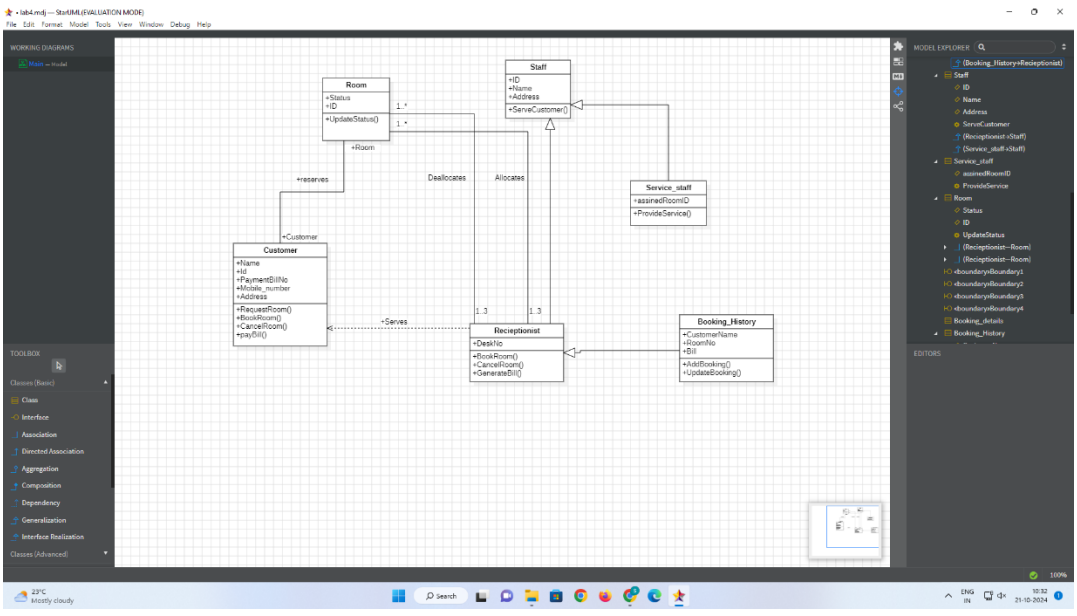
2. General description.
The HMS is targeted at hotel staff management and guests. The system will provide features such as room booking, guest management, billing and suporting.

3. Functional Requirements
1. User Authentication:- Staff and Admin must log in securely
2. Room management:- Ability to add, update and remove details (CRUD operations)
3. Booking System:- Guests can book rooms online and recieve confirmation.
4. check-in/check-out :- Streamlined process for managing guests, ~~room~~ check in time and check out time
5. Billing and payment:- Generating invoice and providing feature to pay the bill.

4. Interface requirements.
- User Interface :- Web based interface accessible via browsers
- API Integration:- Interfaces with third party gateways and secure payments.
- Database communication:- Implement Database communication using SQL languages.
- Admin interface :- To manage room
5. Performance Requirements.
- Response Time :- The system should respond quickly (eg. less than 5 seconds)
- Concurrent users:- Support thousands of simultaneous users without performance degradation.
- Data storage :- Handle tons of data of guests and booking.




## 1.3. Class Diagram

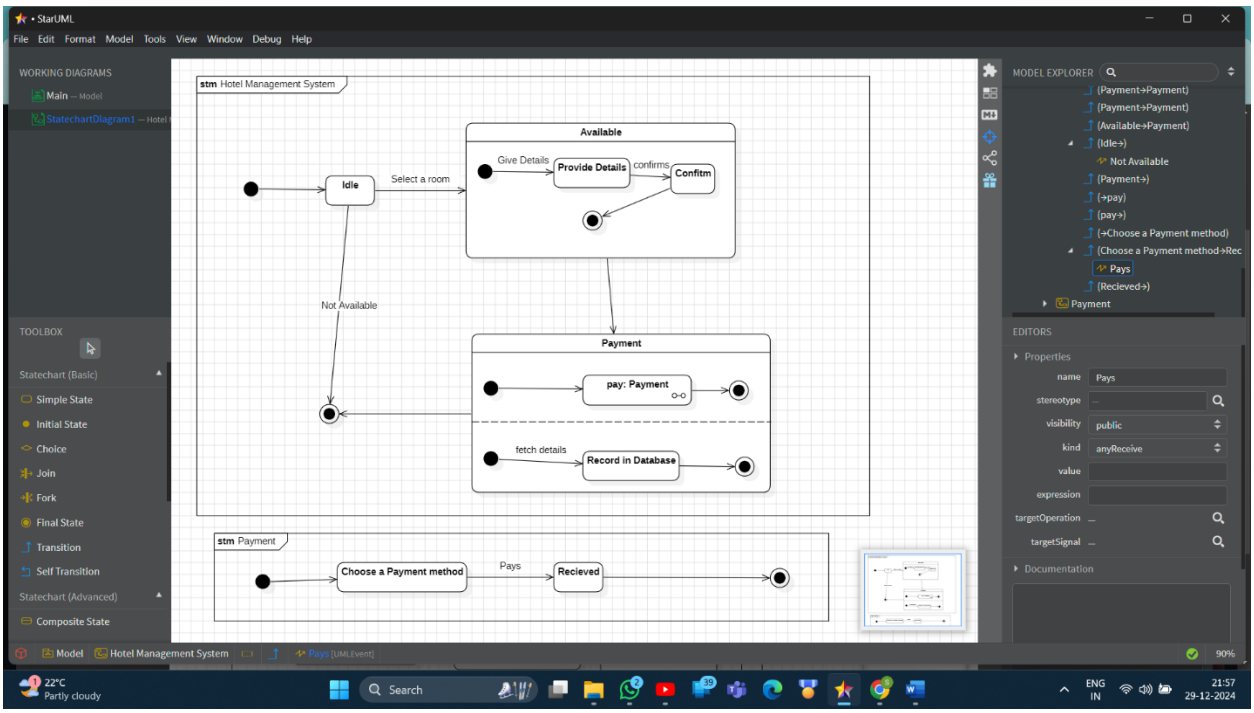


**The diagram shows the different types of things (classes) involved in a hotel management system and how they are related to each other.**

- **Customers** can make **bookings** for **rooms**.
- **Staff**, including **receptionists** and **service staff**, are involved in **managing** the rooms and **bookings**.
- **Receptionists** can **allocate** rooms to customers and **update** the status of rooms.
- **Service staff** can **provide services** to rooms.

**In essence, the diagram outlines the key players (customers, staff) and their interactions with rooms and bookings in a hotel.**

## 1.4. State diagram:




1. **Start (Idle State):**
    - The system starts in an **Idle** state where no action is performed.
    - If a user **selects a room**, the system checks its availability.

---

2. **Room Availability:**

- **If Available:** The user enters booking details, confirms the booking, and proceeds to payment.
- **If Not Available:** The system goes back to the idle state.

---

3. **Payment Process:**
   - The user chooses a payment method (e.g., card or wallet).
   - Payment details are processed and saved in the system's database.

---

4. **Completion:**
   - Once payment is successful, the booking is completed, and the system ends the process.

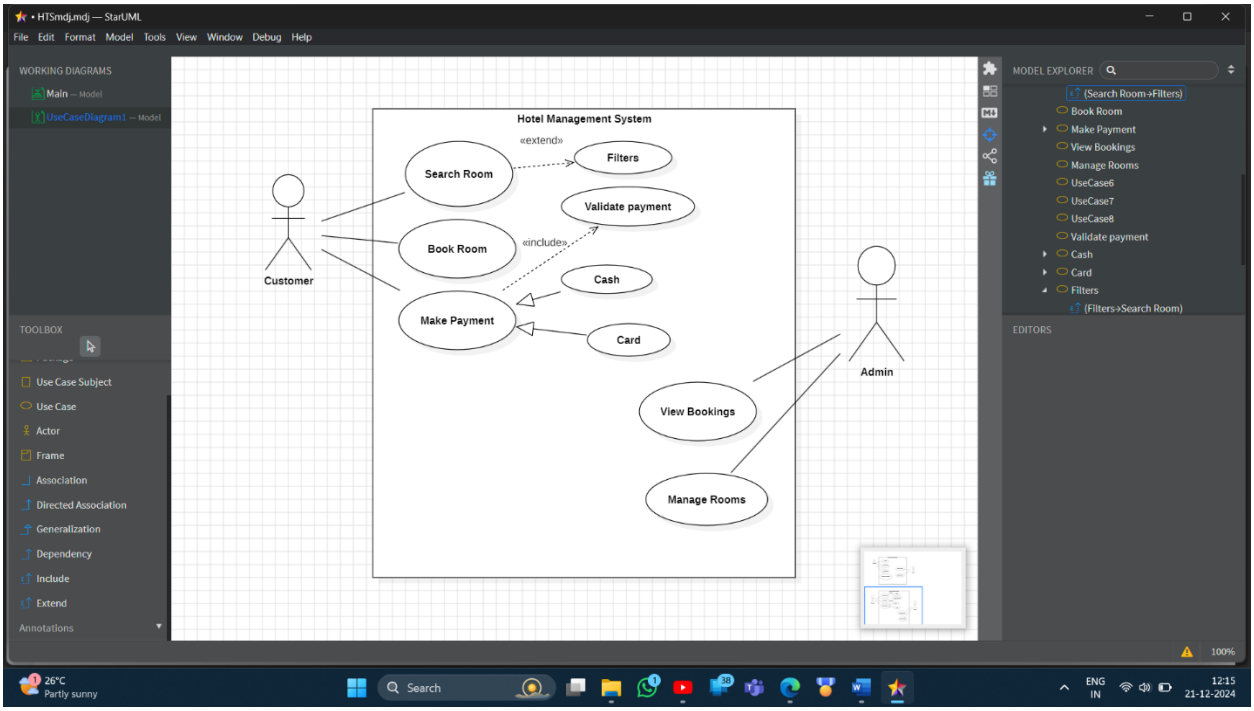
---

**Key Flow:**

1. **Idle → Room Check → Details → Payment → Completion.**
2. If no room is available, it loops back to Idle.

This diagram shows how the system moves from checking room availability to booking and payment in a simple, step-by-step manner.

1.5 Use Case Diagram:



**Actors:**

- **Customer:** Represents a user who interacts with the system to book rooms.
- **Admin:** Represents a user with administrative privileges, responsible for managing rooms and bookings.

**Use Cases:**

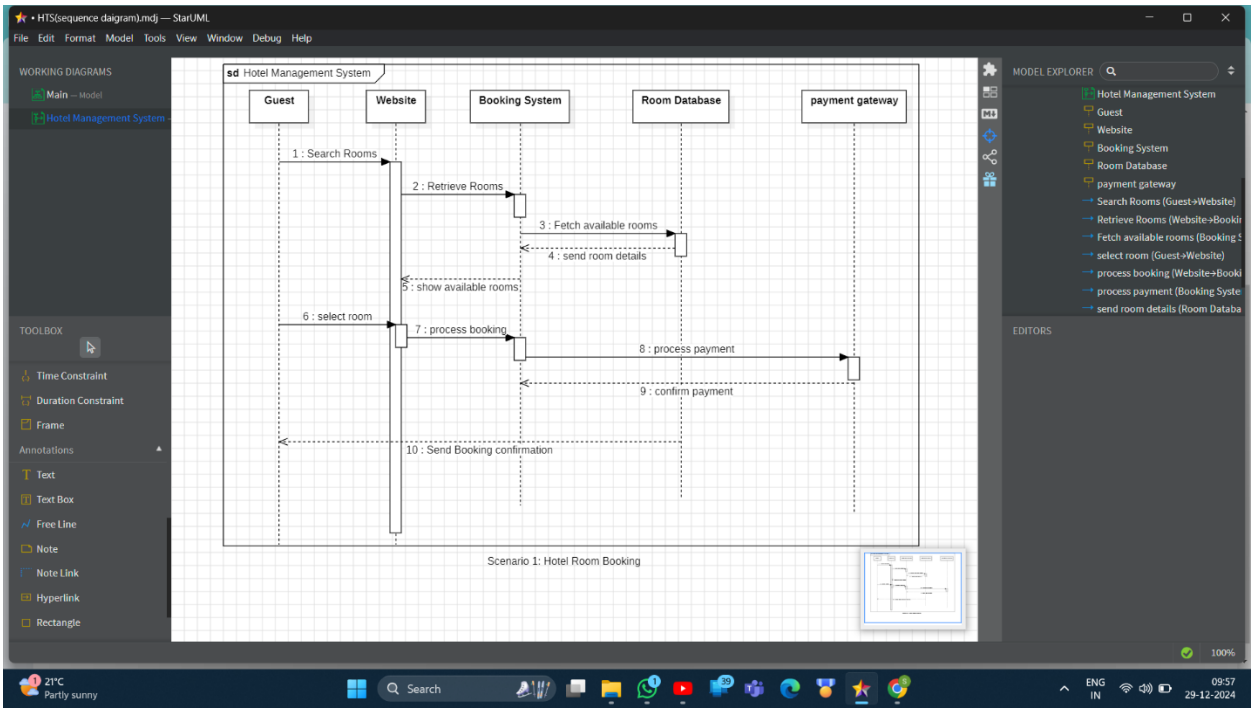
- **Search Room:** The customer can search for available rooms based on certain criteria (filters).
- **Book Room:** The customer can book a selected room.
- **Make Payment:** The customer can make payment for the booking.
- **Validate Payment:** The system validates the payment made by the customer.
- **View Bookings:** The customer can view their past and upcoming bookings.
- **Manage Rooms:** The admin can manage the available rooms, including adding, updating, and deleting rooms.

**Relationships:**

- **Association:** Represents the interactions between actors and use cases. For example, the customer is associated with the "Search Room," "Book Room," "Make Payment," and "View Bookings" use cases.
- **Include:** Represents a use case that is included within another use case. For example, the "Validate Payment" use case is included within the "Make Payment" use case.
- **Extend:** Represents an optional behavior that can be added to a use case. For example, the "Make Payment" use case can be extended to include different payment methods (cash or card).

**Overall, this use case diagram provides a high-level overview of the functional requirements of the hotel management system.** It shows the different actions that users can perform and how they interact with the system.

## 1.6 Sequence Diagram:



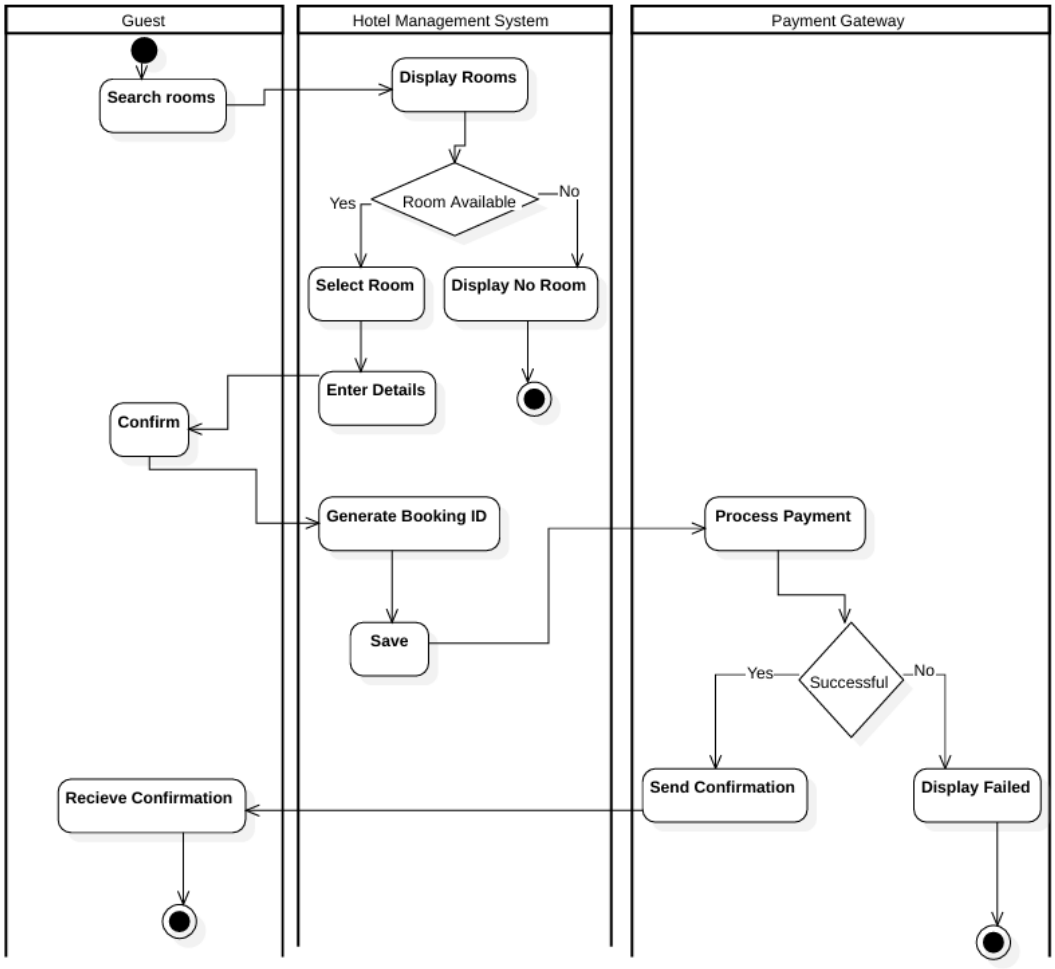

**In this case, the diagram shows how a guest books a hotel room.**

**Here's the step-by-step breakdown:**

1. **The Guest starts the process by searching for rooms.**
2. **The Guest sends the search request to the Website.**
3. **The Website retrieves available rooms from the Room Database.**
4. **The Website sends the available room details back to the Guest.**
5. **The Guest selects a room.**
6. **The Website processes the booking.**
7. **The Website sends a payment request to the Payment Gateway.**
8. **The Payment Gateway processes the payment.**
9. **The Payment Gateway confirms the payment to the Website.**
10. **The Website sends a booking confirmation to the Guest.**

**In essence, the diagram illustrates the flow of communication and actions between the Guest, the Website, the Room Database, and the Payment Gateway during the process of booking a hotel room.**

## 1.7 Activity Diagram:




**This diagram shows the steps involved in booking a hotel room.**

**Here's the step-by-step breakdown:**

1. **The Guest starts by searching for rooms.**
2. **The Hotel Management System displays the available rooms.**
3. **If a room is available, the Guest selects a room and enters their details.**
4. **The system then generates a Booking ID and processes the payment.**
5. **If the payment is successful, the system saves the booking and sends a confirmation to the Guest.**
6. **If the payment fails, the system displays an error message.**

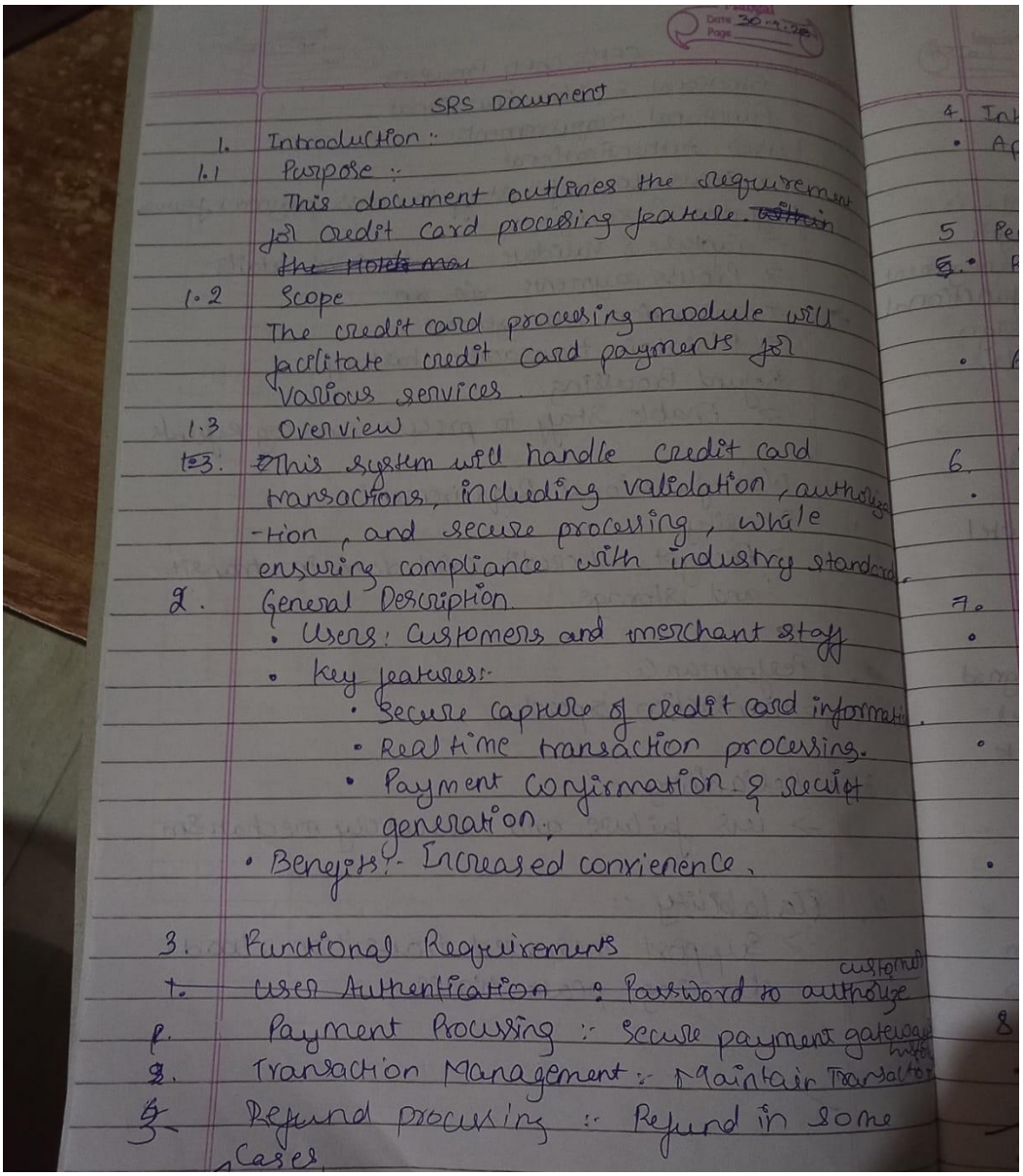
**In essence, this diagram illustrates the flow of events that occur when a guest books a hotel room.**

# 2. Credit Card Processing:

## 2.1 Problem Statement:

Businesses need a secure and efficient way to accept credit card payments from customers. Traditional methods can be slow, error-prone, and costly.

## 2.2 SRS Document:



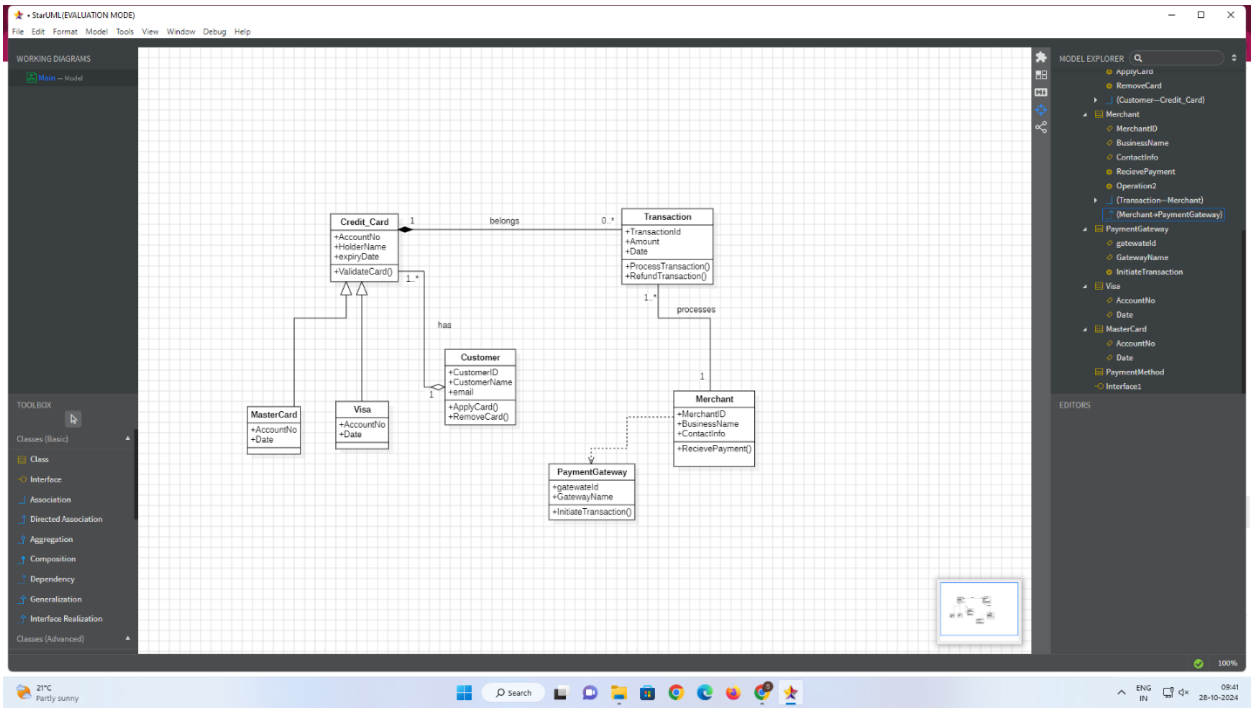
SRS Document

1. Introduction :
1.1 Purpose :
This document outlines the requirement for credit card processing feature ~~within the Hotels mgt~~
1.2 Scope
The credit card processing module will facilitate credit card payments for various services.
1.3 Overview
~~1.3.~~ This system will handle credit card transactions, including validation, authorization, and secure processing, while ensuring compliance with industry standard.
2. General Description
- Users: Customers and merchant staff
- Key features:
  - Secure capture of credit card information.
  - Real time transaction processing.
  - Payment confirmation & receipt generation.
- Benefits: Increased convienence.

3. Functional Requirements
1. ~~User Authentication~~ : Password to authorize customer
2. Payment Processing : Secure payment gateway integration
3. Transaction Management : Maintain Transaction
4. Refund processing : Refund in some Cases.

4. Interface Requirements
   - Api Integration
     - Interface with secure payment gateways.
5. Performance Requirements.
   - Response time
     - Transaction processing should complete within 5 sec @ fast.
   - Availability
     - The system should maintain 100% uptime.
6. Design Constraints
   - Compliance
     - Must adhere to standards for secure handling of credit Card Information.
7. Non-functional Attributes.
   - Security
     - Implement encryption for data transmission and storage.
   - Reliability:
     - Ensure failover mechanisms for transaction processing.
   - Scalability:-
     - The system should handle an increasing number of transactions during peak times.

## 2.3. Class Diagram:

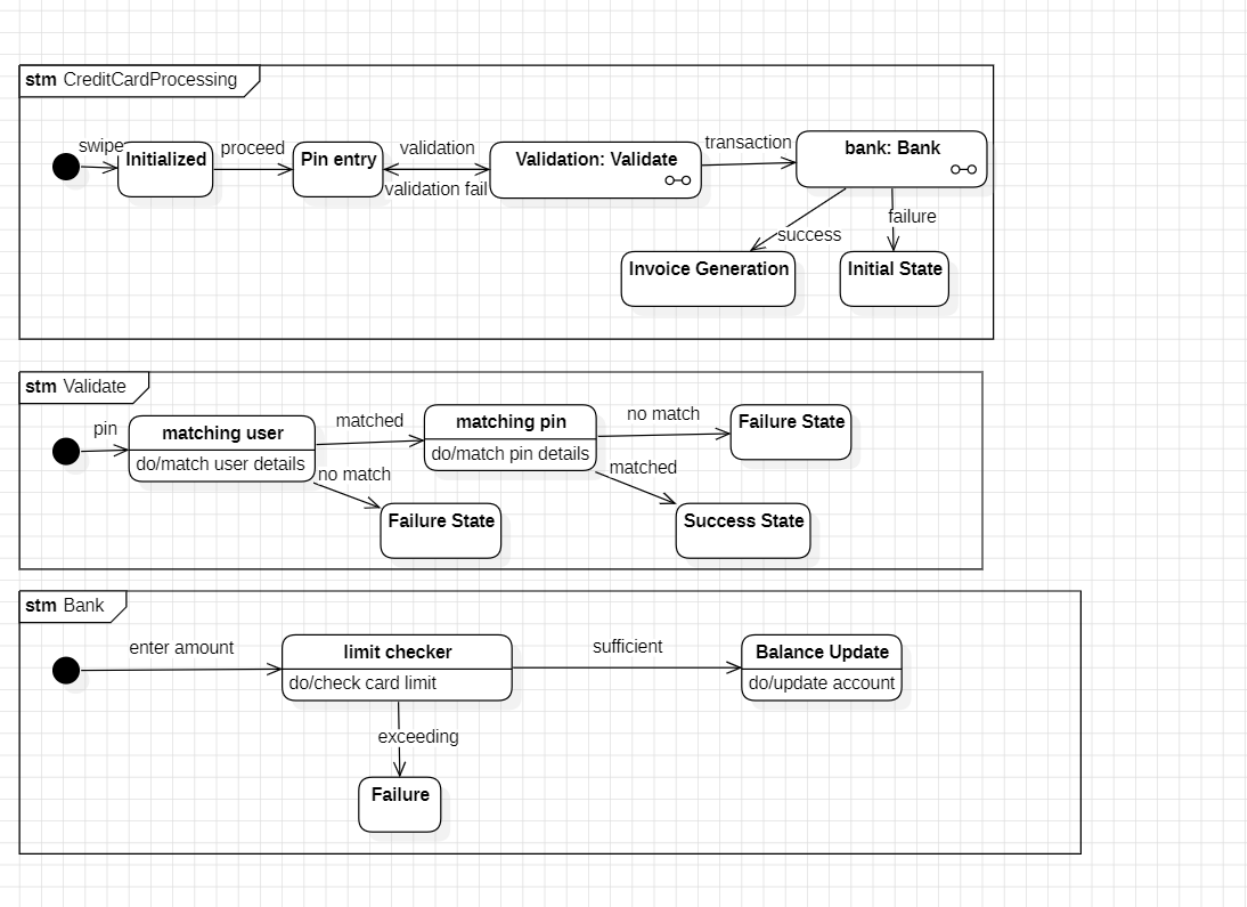



- **Credit Card:** This represents a credit card with details like Account Number, Holder Name, and Expiry Date. It has operations like Validate Card.
- **Customer:** This represents a customer who owns the credit card. It has attributes like Customer ID, Name, and Address.
- **Merchant:** This represents a business that accepts credit card payments. It has attributes like Business Name and Contact Info, and operations like Receive Payment.
- **Payment Gateway:** This is the intermediary that processes transactions between the Customer, Merchant, and the Credit Card network. It has operations like Initiate Transaction and Process Transaction.
- **Transaction:** This represents a single payment transaction, with details like Transaction ID, Amount, and Date.

**How are these entities related?**

- A **Customer** can have **one or more Credit Cards**.
- A **Merchant** can receive payments for **one or more Transactions**.
- A **Transaction** involves a **Credit Card** and a **Merchant**.
- The **Payment Gateway** processes **Transactions**.

## 2.4. State Diagram:




### 1. Credit Card Processing (Top Diagram)

- **Initial State:** This is the starting point of the process.
- **Swipe:** The user swipes their credit card.
- **Initialized:** The system recognizes the card and initializes the process.
- **Pin Entry:** The user enters their PIN.
- **Validation:** The system validates the card details and the entered PIN.
    - If validation fails, the process goes to the **Failure State**.
    - If validation succeeds, the system proceeds to the **Transaction** state.
- **Transaction:** The system processes the transaction with the bank.
    - If the transaction is successful, the system generates an **Invoice** and returns to the **Initial State**.
    - If the transaction fails, the process goes to the **Failure State**.

### 2. Validate (Middle Diagram)

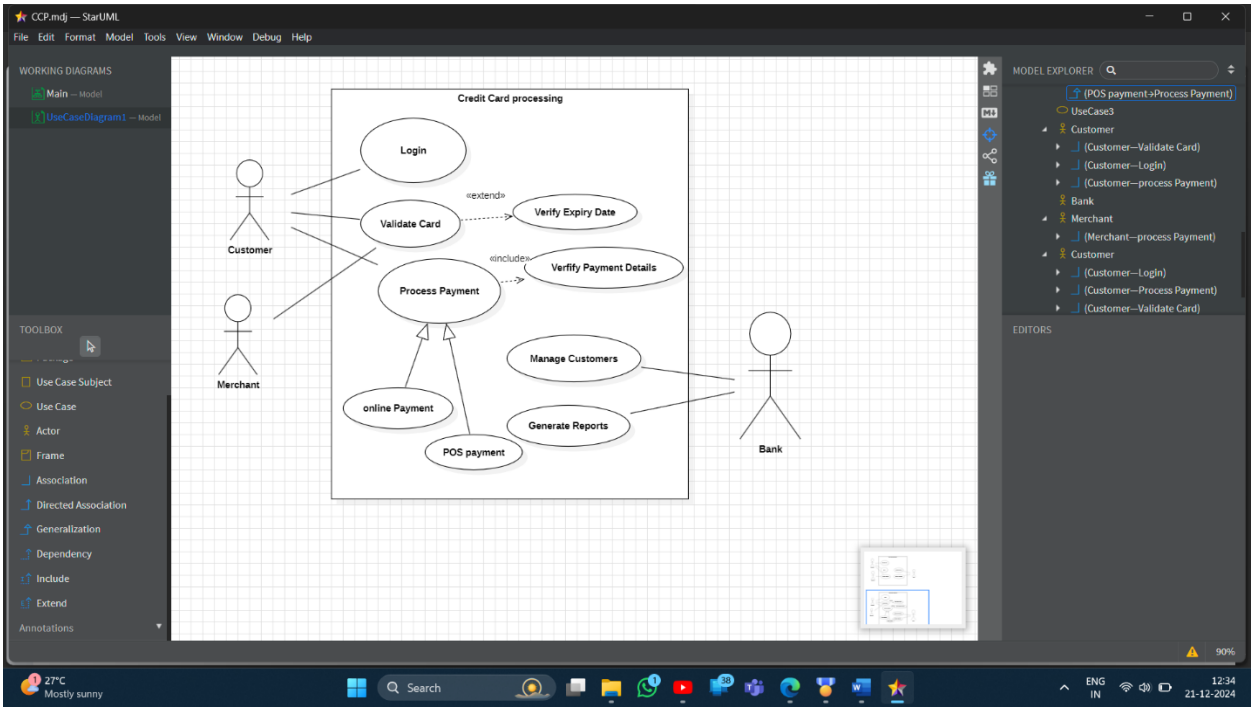
- **Pin:** The user enters their PIN.
- **Matching User:** The system matches the user details associated with the card.
    - If the user details match, the system proceeds to **Matching Pin**.
    - If the user details don't match, the process goes to the **Failure State**.
- **Matching Pin:** The system matches the entered PIN with the stored PIN.
    - If the PIN matches, the process goes to the **Success State**.
    - If the PIN doesn't match, the process goes to the **Failure State**.

**3. Bank (Bottom Diagram)**

- **Enter Amount:** The user enters the transaction amount.
- **Limit Checker:** The system checks if the transaction amount is within the card's credit limit.
    - If the amount is within the limit, the process goes to **Balance Update**.
    - If the amount exceeds the limit, the process goes to the **Failure State**.
- **Balance Update:** The system updates the card's balance and completes the transaction.

**In essence, this diagram shows the sequence of steps involved in a credit card transaction, from the initial card swipe to the final confirmation or rejection of the transaction**

## 2.5. Use Case Diagram



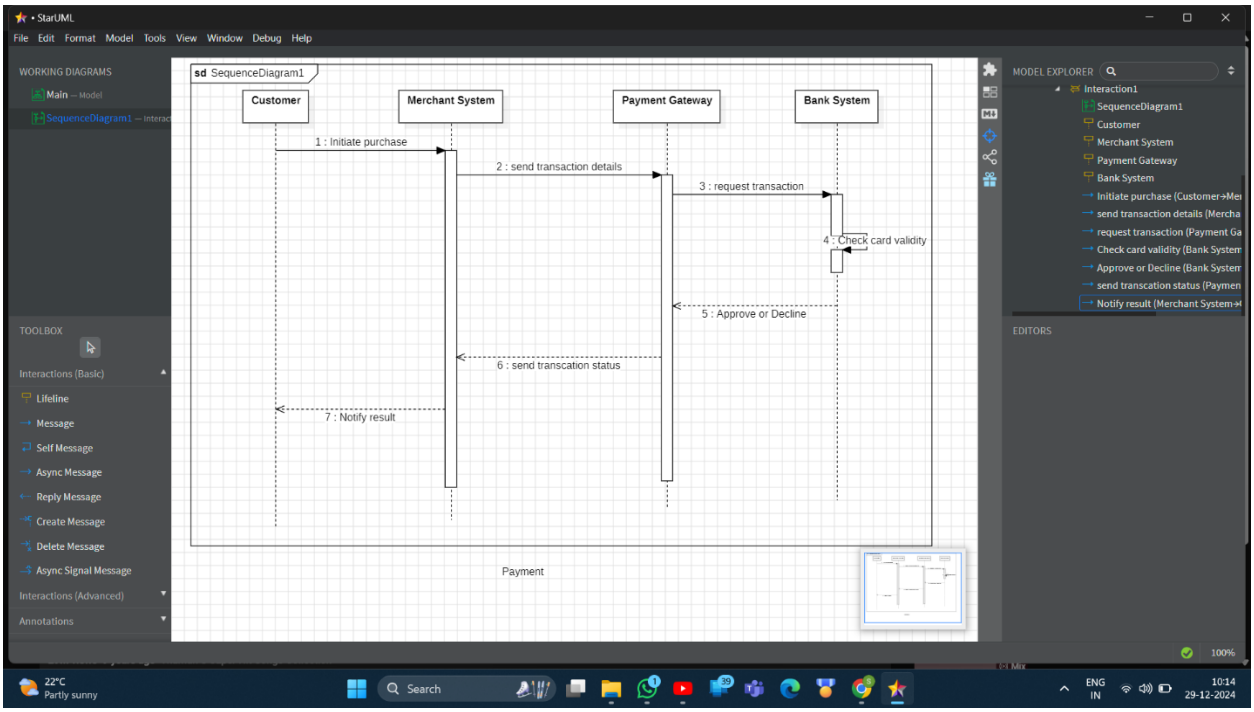
- **Actors:**
    - **Customer:** Represents a user who makes online or POS (Point of Sale) payments.
    - **Merchant:** Represents a business that accepts credit card payments.
    - **Bank:** Represents the financial institution that handles the credit card transactions.
- **Use Cases:**
    - **Login:** The customer logs into the system to make a payment.
    - **Validate Card:** The system validates the customer's credit card information.
    - **Verify Expiry Date:** The system checks if the credit card is still valid.
    - **Verify Payment Details:** The system verifies the payment amount and other details.
    - **Process Payment:** The system processes the payment transaction.
    - **Online Payment:** Represents online credit card payments.
    - **POS Payment:** Represents point-of-sale credit card payments.
    - **Manage Customers:** The system manages customer information.
    - **Generate Reports:** The system generates reports related to transactions.

**Relationships:**

- **Include:** Indicates that one use case includes the functionality of another. For example, "Process Payment" includes "Validate Card" and "Verify Payment Details."
- **Extend:** Indicates that one use case can optionally extend the functionality of another. For example, "Process Payment" can be extended to include "Online Payment" or "POS Payment."

**In essence, this diagram shows the different actions that customers, merchants, and the bank can perform in a credit card processing system.**

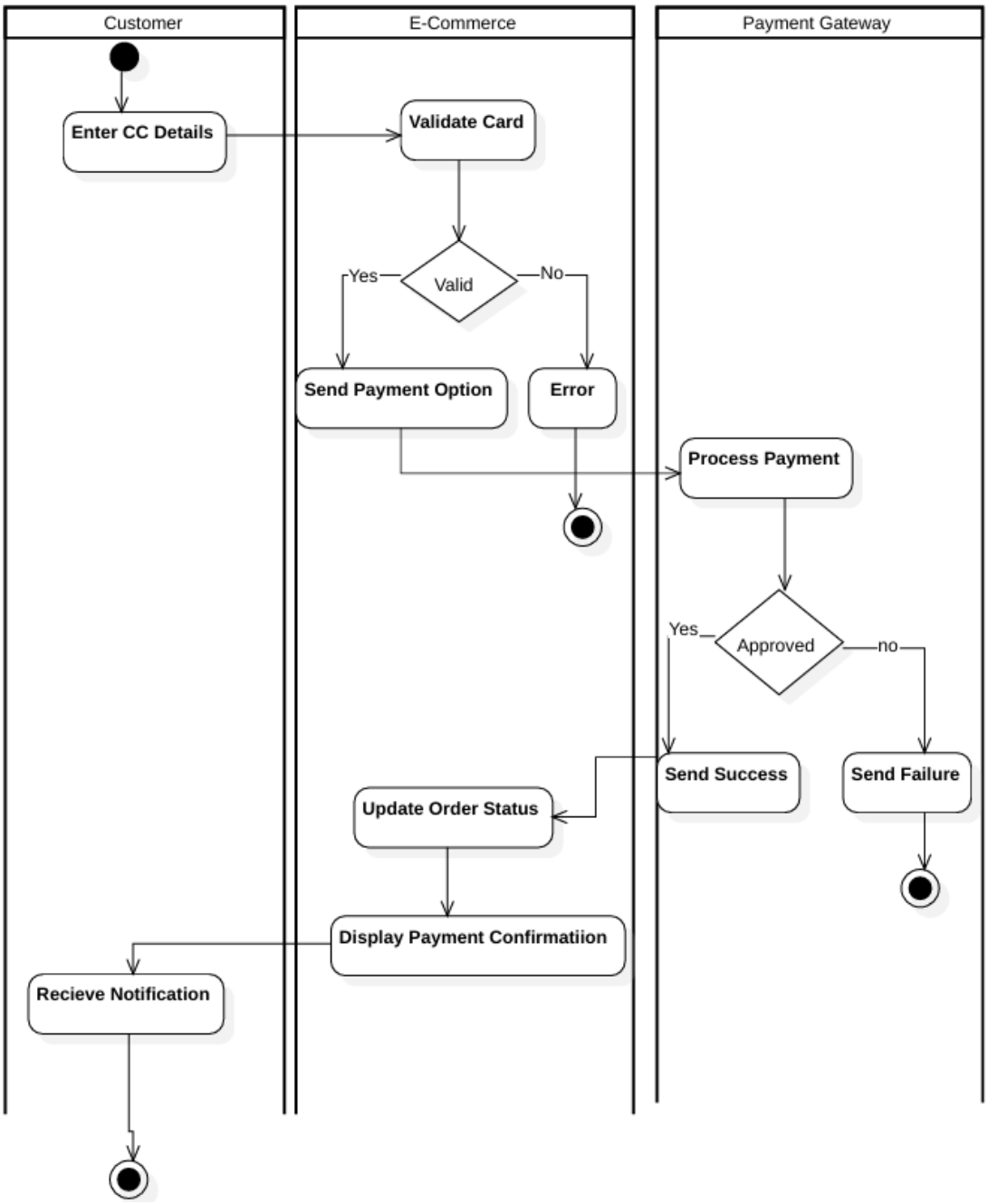
## 2.6. Sequence diagram:




1. **The Customer initiates a purchase.**
2. **The Customer sends the transaction details (like amount, card information) to the Merchant System.**
3. **The Merchant System forwards the transaction request to the Payment Gateway.**
4. **The Payment Gateway sends the transaction request to the Bank System.**
5. **The Bank System checks the validity of the card and the available balance.**
6. **The Bank System sends the approval or decline decision back to the Payment Gateway.**
7. **The Payment Gateway notifies the Merchant System about the transaction status (approved or declined).**
8. **The Merchant System notifies the Customer about the transaction result.**

**In essence, this diagram shows the flow of communication and actions between the Customer, Merchant System, Payment Gateway, and Bank System during a credit card transaction.**

## 2.7 Activity Diagram:




1. **Customer:**
    - The process starts with the **Customer** entering their credit card details.
2. **E-commerce System:**
    - The system then **validates** the entered credit card information.
    - If the card is **valid**, the system sends the payment option to the customer.
    - If the card is **invalid**, the system displays an **error** message.

3. **Payment Gateway:**
    - If the card is valid, the system sends the payment details to the **Payment Gateway** for processing.
    - The Payment Gateway processes the payment and determines if it is **approved** or **declined**.
4. **E-commerce System (continued):**
    - If the payment is **approved**, the system updates the order status and sends a **success** message.
    - If the payment is **declined**, the system sends a **failure** message.
5. **Customer:**
    - The Customer receives a notification of the payment status (success or failure) and a confirmation message if the payment is successful.

**In essence, this diagram shows the flow of events and decision points involved in making an online payment using a credit card.**

**Key Points:**

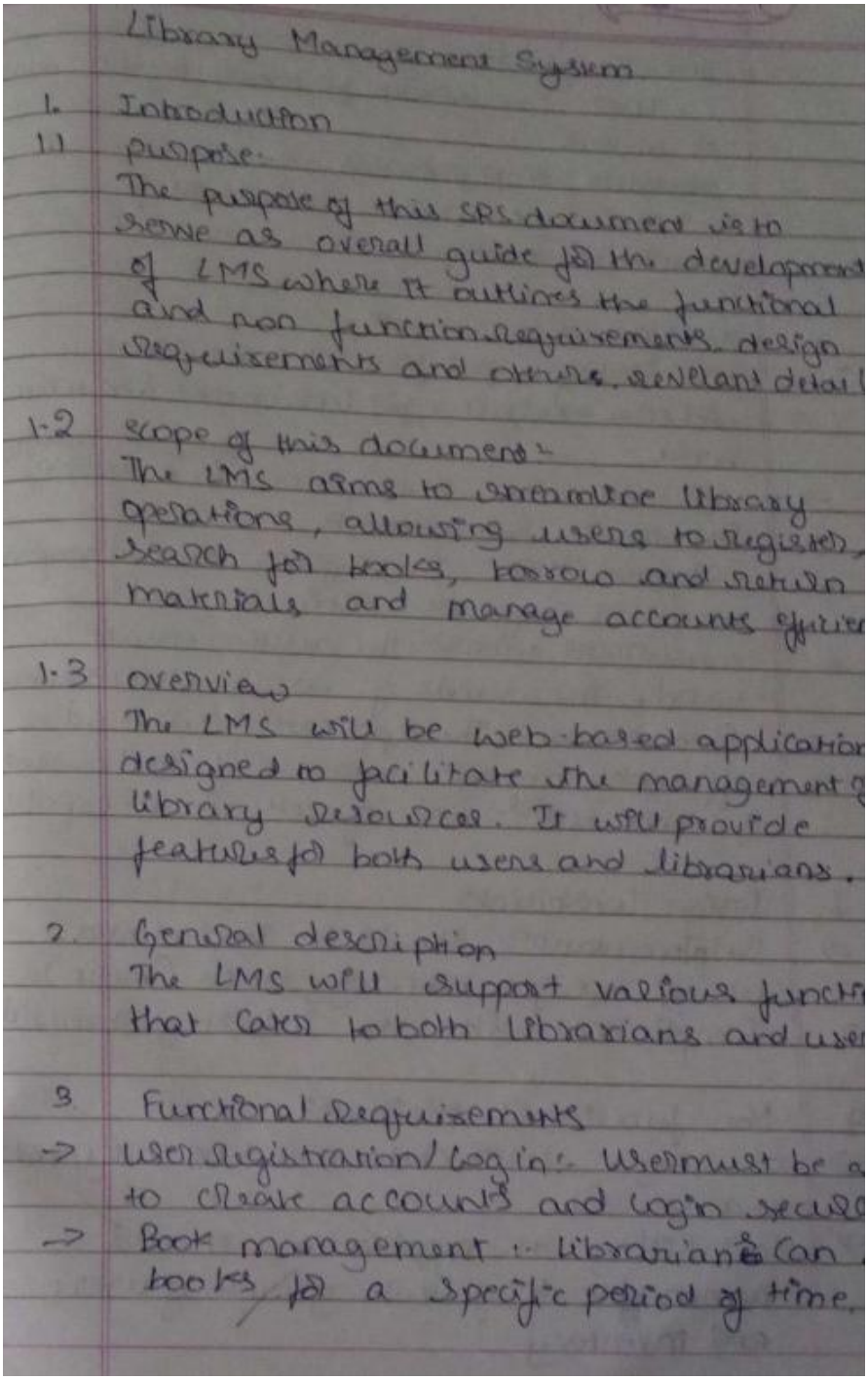
- **Swimlanes:** The vertical columns represent the different actors or systems involved in the process.
- **Arrows:** The arrows indicate the flow of actions and information between the actors.
- **Shapes:**
    - Rounded rectangles represent actions.
    - Diamonds represent decision points.
    - Circles represent the start or end of the process.

# 3. Library Management System

## 3.1 Problem Statement

Traditional library management often relies on manual processes like paper records, manual book checkouts, and slow data retrieval. This can lead to inefficiencies, inaccuracies, and a poor user experience.

## 3.2 SRS Document



Library Management System

1. Introduction

1.1 purpose:

The purpose of this SRS document is to serve as overall guide for the development of LMS where it outlines the functional and non function requirements, design requirements and other relevant detail

1.2 scope of this document:

The LMS aims to streamline library operations, allowing users to register, search for books, borrow and return materials and manage accounts effici

1.3 overview

The LMS will be web-based applicatio designed to facilitate the management library resources. It will provide features for both users and librarians.

2. General description

The LMS will support various functi that cater to both librarians and use

3 Functional Requirements

-> User registration/login: User must be a to create accounts and login secur

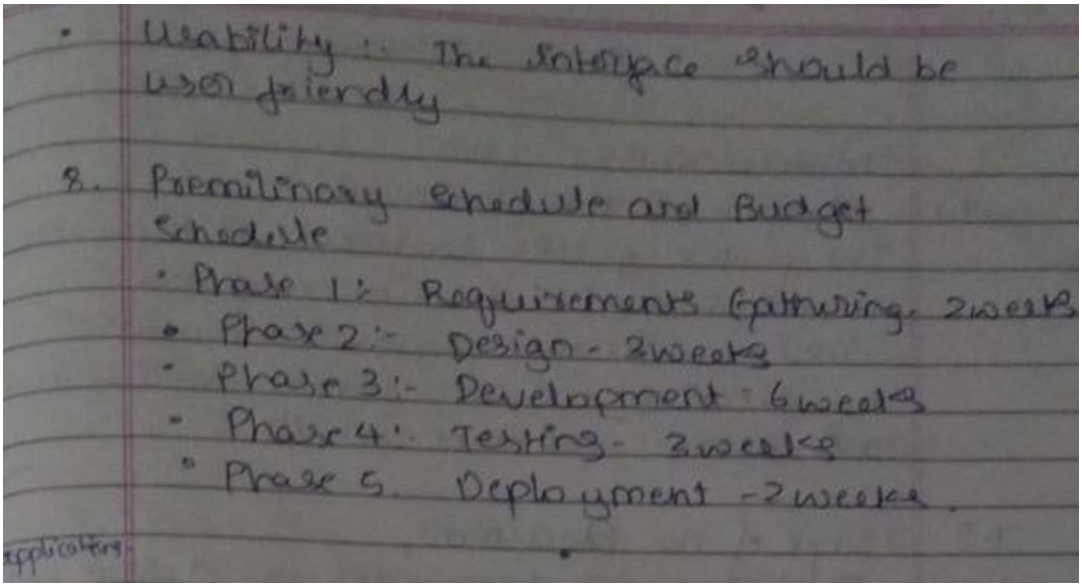
-> Book management: librarians can books for a specific period of time.

→ Search functionality:
Users can search for books by title, author
→ Overdue Notifications:-
The system sends reminders for overdue books.

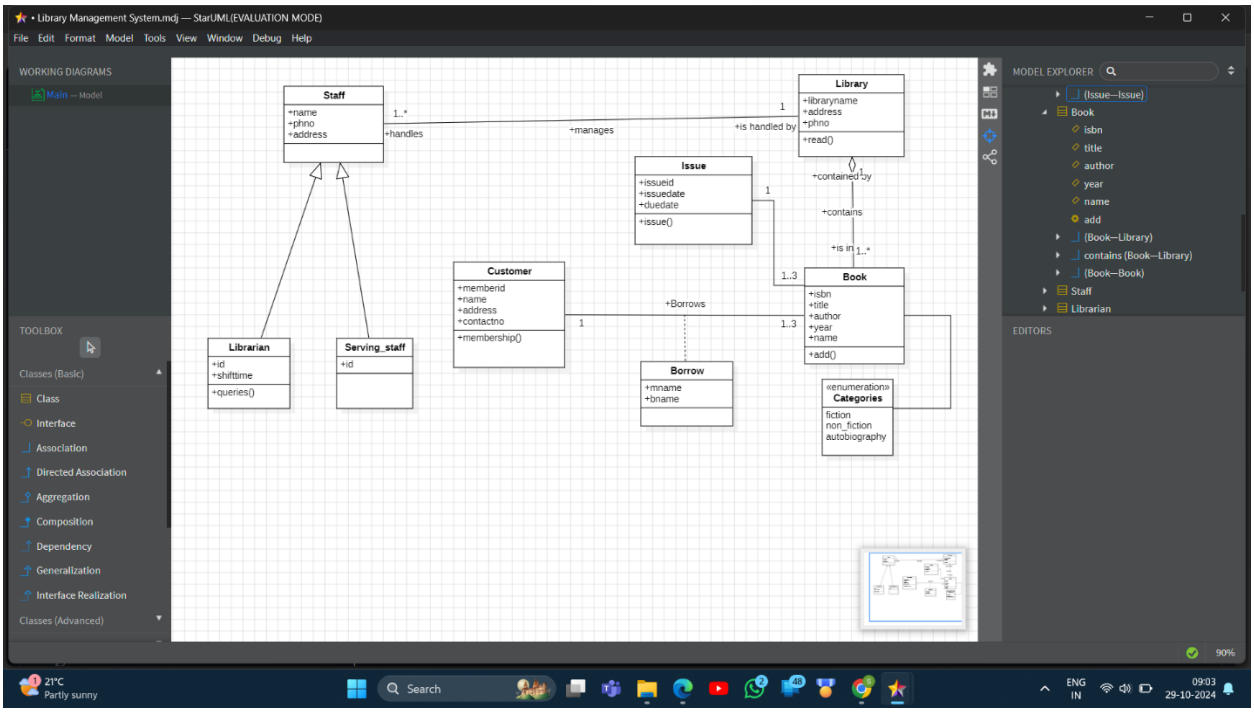
4. Interface Requirements:-
→ User Interface: for user interactions
→ Admin Interface: for librarians and managers
→ API integration: so that it can be accessible through third-party
5. Performance Requirements:-
→ Response time:- The system should respond to user requests within 5 seconds
→ concurrent users:- The system must handle thousands of users simultaneously
→ Data storage:- The application should efficiently manage database of lakhs/thousands of book records and quick search capabilities

6. Design Constraints
→ Implementation:- should be implemented using specific technology stack (mern)
→ Compliance with the industry standards

7. Non functional Attributes
→ security:- User data must be encrypted and secured.
→ Scalability:- The application should accommodate to future growth of users and inventory




## 3.3 Class Diagram:




Certainly, let's break down this diagram in a simpler way.

**What is this diagram about?**

This is a class diagram, a visual representation used in software engineering to model the structure of a system. In this case, it illustrates the entities and relationships involved in a library management system.

**Here's a simplified explanation:**

- **Entities:**

- **Library:** Represents a library with attributes like name and address.
- **Book:** Represents a book with attributes like ISBN, title, author, and year.
- **Customer:** Represents a library member with attributes like membership ID, name, address, and contact information.
- **Staff:** Represents library employees with attributes like name and phone number.
- **Librarian:** A type of staff member, likely with additional responsibilities.
- **Serving Staff:** Another type of staff member, likely responsible for assisting customers.
- **Issue:** Represents a record of a book being issued to a customer.
- **Borrow:** Represents an instance of a customer borrowing a book.
- **Categories:** Represents different categories of books, such as fiction, non-fiction, etc.

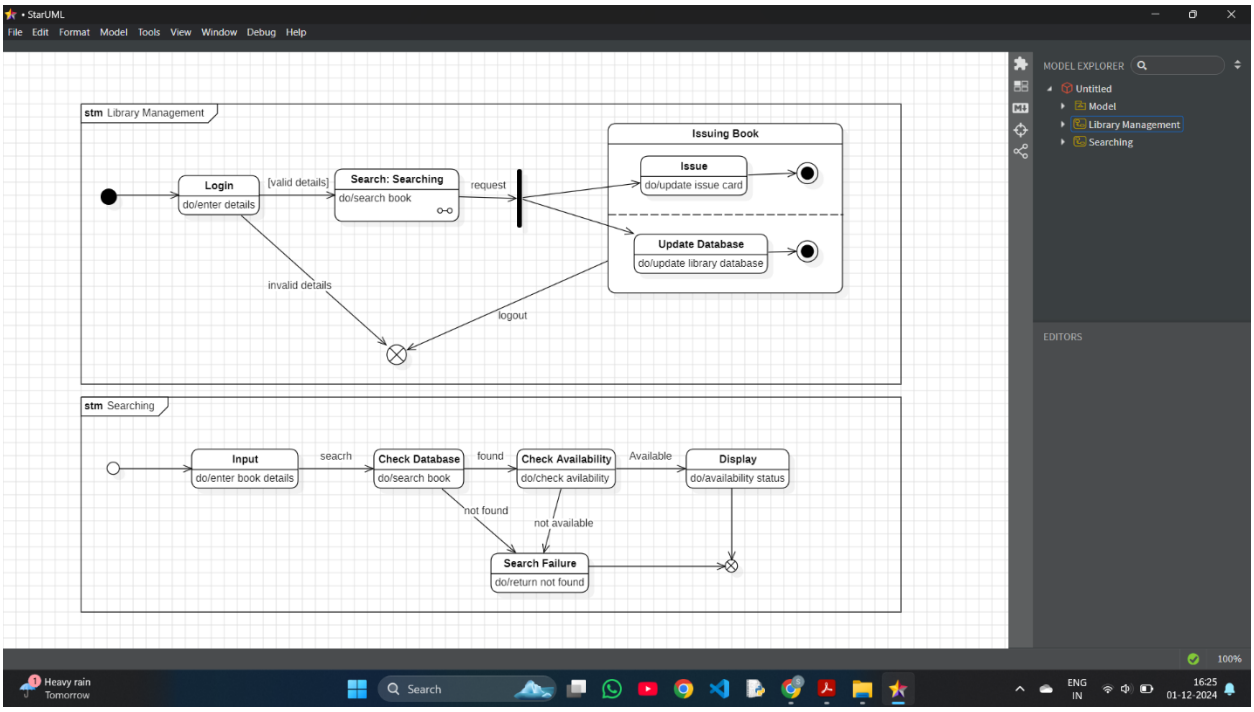
- **Relationships:**
  - **1:1:** A one-to-one relationship, meaning one instance of one entity is associated with only one instance of another entity. For example, a Librarian is also a Staff member.
  - **1:N:** A one-to-many relationship, meaning one instance of one entity can be associated with multiple instances of another entity. For example, a Library can have many Books.
  - **N:M:** A many-to-many relationship, meaning multiple instances of one entity can be associated with multiple instances of another entity. For example, a Customer can borrow multiple Books, and a Book can be borrowed by multiple Customers.

**In essence, this diagram shows the different components of a library management system (like books, customers, staff) and how they are related to each other.**

**Key Points:**

- **Boxes:** Represent the different classes (entities).
- **Lines:** Represent the relationships between classes.
- **Multiplicity:** The numbers next to the lines indicate the cardinality of the relationship (e.g., 1:1, 1:N, N:M).

## **3.4** State diagram




**Issuing a Book:**

- **Initial State:** The process starts here.
- **Login:** The librarian logs into the system using their credentials.
- **Search:** The librarian searches for the book to be issued.
- **Issue:** If the book is found and available, the librarian proceeds to issue the book to the customer. This involves updating records and issuing a library card.
- **Update Database:** The system updates the library database to reflect the book being issued.

**2. Searching for a Book:**

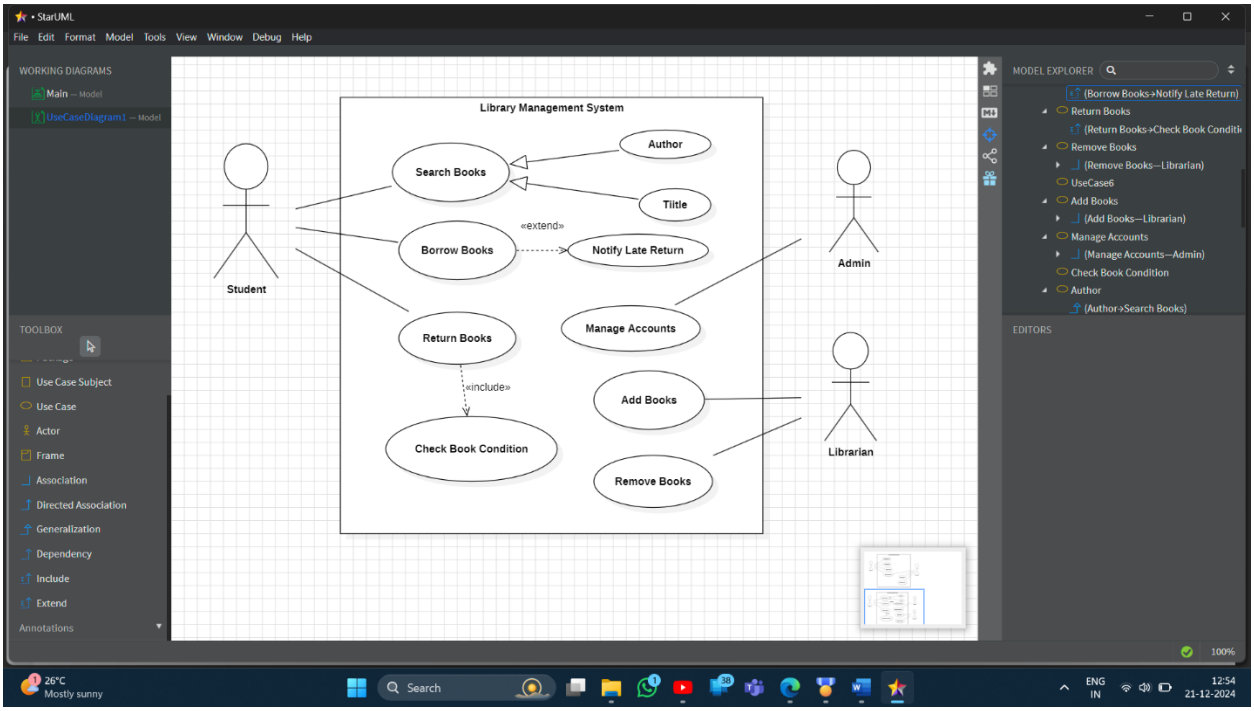
- **Input:** The user enters the book details they are searching for.
- **Check Database:** The system searches the library database for the book.
- **Check Availability:** If the book is found, the system checks its availability status (whether it is available for borrowing).
- **Available:** If the book is available, the system displays the book details to the user.
- **Not Available:** If the book is not available, the system displays a message indicating that the book is not available.
- **Search Failure:** If the book is not found in the database, the system displays a search failure message.

**In essence, this diagram shows the sequence of steps involved in issuing a book and searching for books in a library management system.**

**Key Points:**

- **States:** The rounded rectangles represent different states in the process.
- **Transitions:** The arrows represent the transitions between states, triggered by specific events or conditions.

## 3.5 Use Case Diagram:




**Actors:**

- **Student:** Represents a library user who borrows books.
- **Librarian:** Represents a library employee who manages library operations.
- **Admin:** Represents an administrator with higher privileges, responsible for overall management.

**Use Cases:**

- **Search Books:** Students can search for books in the library's catalog.
- **Borrow Books:** Students can borrow books from the library.
- **Return Books:** Students can return borrowed books to the library.
- **Check Book Condition:** Librarians can check the condition of returned books.
- **Notify Late Return:** The system can notify students about overdue books.

- **Add Books:** Librarians or Admins can add new books to the library's collection.
- **Remove Books:** Librarians or Admins can remove books from the library's collection.
- **Manage Accounts:** Admins can manage user accounts (students and librarians).

**Relationships:**

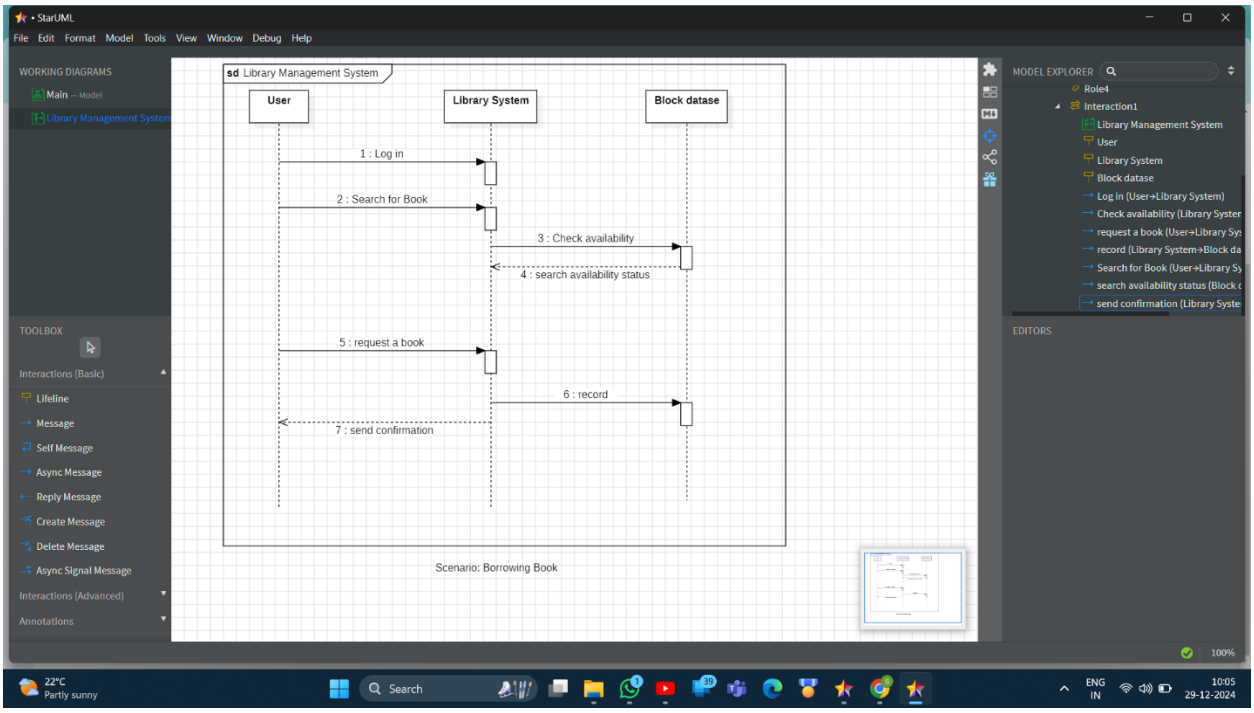
- **Include:** Indicates that one use case includes the functionality of another. For example, "Borrow Books" includes "Search Books."

**In essence, this diagram shows the different actions that students, librarians, and admins can perform in a library management system.**

**Key Points:**

- **Actors:** The stick figures represent the users (actors) interacting with the system.
- **Use Cases:** The ovals represent the different actions that can be performed.
- **Relationships:** The lines and arrows indicate the relationships between use cases.

## 3.6 Sequence Diagram:




1. **The User logs into the library system.**
2. **The User searches for the desired book.**
3. **The Library System checks the availability of the book.**
4. **The Library System searches for the availability status of the book.**

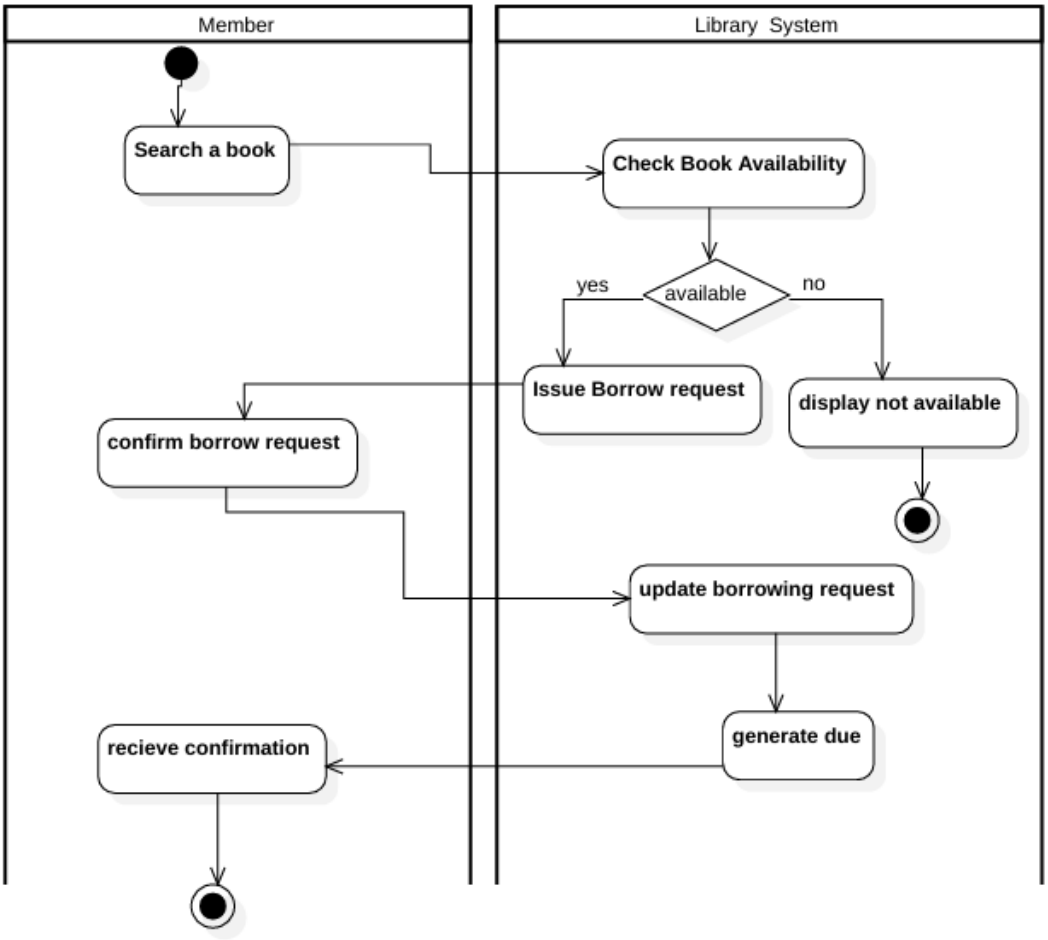
5. **The User requests to borrow the book.**
6. **The Library System records the book issue and sends a confirmation to the User.**

**In essence, this diagram shows the flow of communication and actions between the User and the Library System during the process of borrowing a book.**

**Key Points:**

- **Lifelines:** The vertical lines represent the different components involved (User, Library System, Book Database).
- **Messages:** The arrows represent the messages exchanged between these components.
- **Time:** The horizontal axis represents time, showing the sequence of events.

## 3.7 Activity:



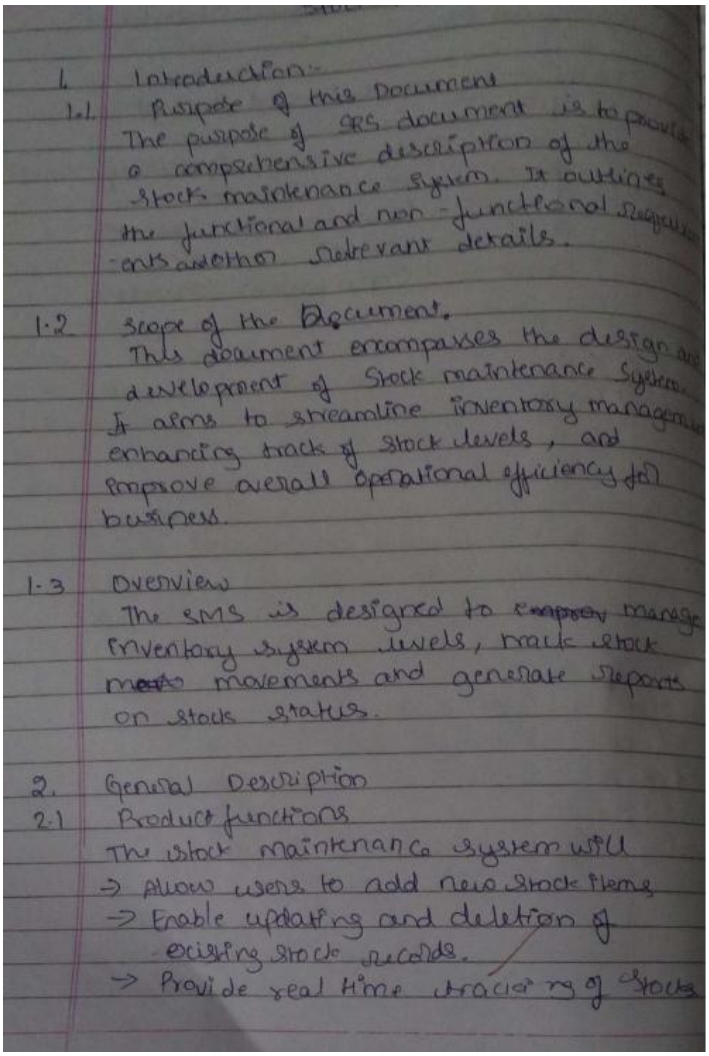
# 4. Stock maintenance System

## 4.1 Problem Statement

Businesses often struggle to efficiently manage their inventory, leading to:

- **Stockouts:** Running out of essential products, resulting in lost sales and dissatisfied customers.
- **Overstocking:** Holding excessive inventory, tying up capital and increasing storage costs.
- **Poor Inventory Visibility:** Lack of real-time data on stock levels, hindering accurate demand forecasting and replenishment planning.
- **Manual Processes:** Time-consuming manual processes for tracking stock movements, increasing the risk of errors and inefficiencies.

## 4.2 SRS Document:



1. Introduction:-

1.1 Purpose of this Document
The purpose of SRS document is to provide a comprehensive description of the Stock maintenance System. It outlines the functional and non-functional requirements and other relevant details.

1.2 Scope of the Document.
This document encompasses the design and development of Stock maintenance System. It aims to streamline inventory management, enhancing track of stock levels, and improve overall operational efficiency for business.

1.3 Overview
The SMS is designed to ~~improve~~ manage inventory system levels, track stock ~~move~~ movements and generate reports on stock status.

2. General Description

2.1 Product functions
The stock maintenance system will
→ Allow users to add new stock items
→ Enable updating and deletion of existing stock records.
→ Provide real time tracking of stocks

2.2 User characteristics

-> Target users include inventory managers, warehouse staff and business who require an efficient system to manage stock.

3. Functional Requirements

-> Stock management

-> Ability to add, update and delete stock items.

-> Reporting:

- Generate reports on stock levels, sales and reordering needs.

-> User management

-> Support multiple user accounts.

-> Notifications

-> Send alerts when stock levels are low.

4. Interface Requirements

-> User Interface: The system will feature a web-based interface accessible via standard browsers.

-> API's: Integrate with external systems.

-> Database Interface: Communicate with a relational database to store and retrieve stock data.

5. Performance requirement

- Response time: Time The system should respond to user actions within 2 seconds
- Memory usage: The application should operate within a memory footprint of 100m

6. Design constraints:
→ Technology Stack: The system must be developed using specified technologies (Java, Python)
→ Deployment Environment: Must be deployable on cloud platforms like AWS or Azure.
→ Compliance: The system must adhere to data protection regulations relevant to industry.

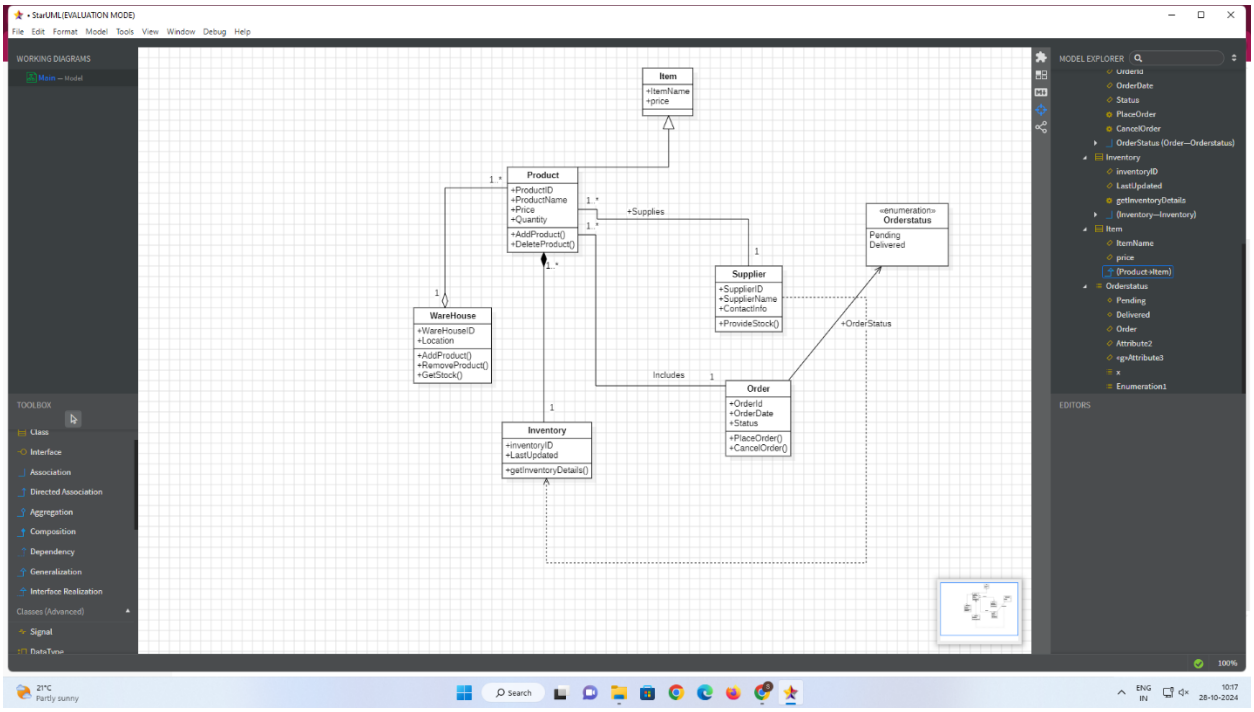
7. Non functional budget:
→ Security: Implement encryption and decryption methodologies
→ Portability: The system should be able to operate on multiple devices
→ Reliability: It should maintain 99% uptime
→ Scalability: Should be capable of increasing no of stocks without performance degradation.

8. Preliminary schedule and budget

| Phases | Tasks | No of weeks |
|---|---|---|
| 1. | Requirements gathering | 2 weeks |
| 2. | Design | 3 weeks |
| 3. | Development | 1 month |
| 4. | Testing | 3 weeks |
| 5. | Deployment | 2 weeks |

Budget estimated about 50 lakhs.

## 4.3 Class Diagram:




**Entities:**

- **Product:** Represents a product with attributes like ProductID, ProductName, and Price.
- **Item:** Represents a specific instance of a product with attributes like ItemID and Quantity.
- **Warehouse:** Represents a location where inventory is stored, with attributes like WarehouseID and Location.
- **Supplier:** Represents a company that supplies products to the warehouse.
- **Order:** Represents an order placed to a supplier to replenish inventory.
- **Inventory:** Represents the overall stock of products in the warehouse.

**Relationships:**

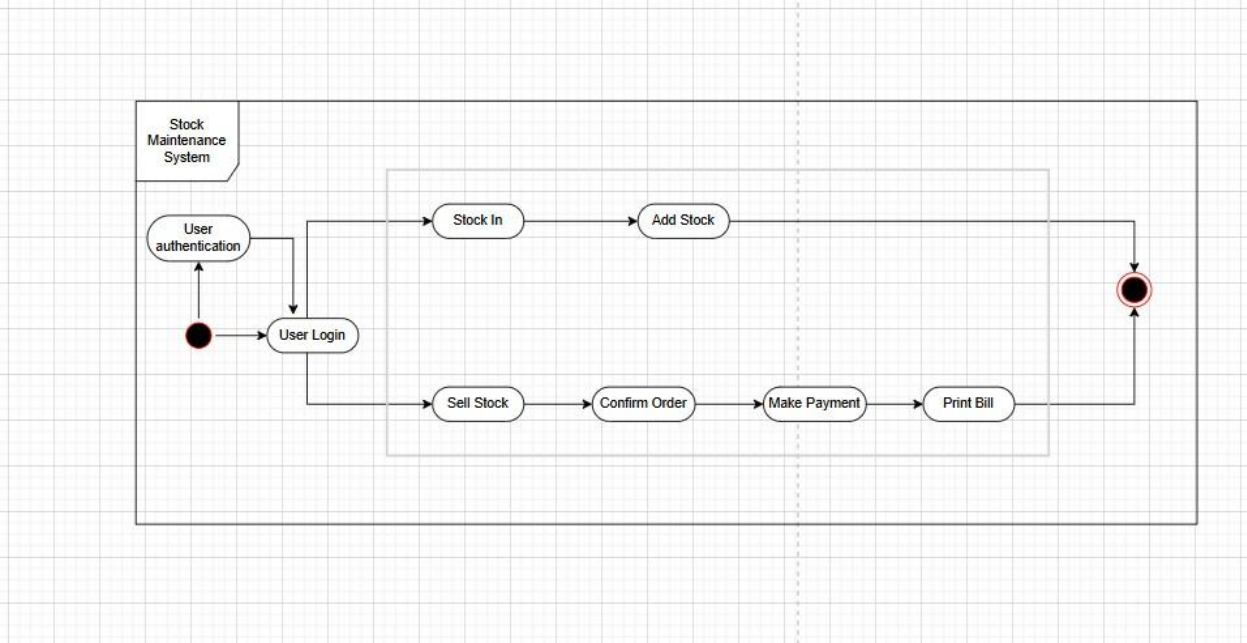
- **1:1:** A one-to-one relationship, meaning one instance of one entity is associated with only one instance of another entity. For example, an Item is associated with one Product.
- **1:N:** A one-to-many relationship, meaning one instance of one entity can be associated with multiple instances of another entity. For example, a Product can have multiple Items in inventory, and a Warehouse can have multiple Items stored.
- **N:M:** A many-to-many relationship, meaning multiple instances of one entity can be associated with multiple instances of another entity. For example, a Supplier can supply multiple Products, and a Product can be supplied by multiple Suppliers.

**In essence, this diagram shows the different components of an inventory management system (like products, warehouses, orders) and how they are related to each other.**

**Key Points:**

- **Boxes:** Represent the different classes (entities).
- **Lines:** Represent the relationships between classes.
- **Multiplicity:** The numbers next to the lines indicate the cardinality of the relationship (e.g., 1:1, 1:N, N:M).

## 4.4 State Diagram:




1. **User Login:** The process begins with the user logging into the system.
2. **Stock Maintenance System:**
    - **User Authentication:** The system verifies the user's credentials to ensure they have the necessary permissions.
    - **Stock In:** This part likely involves adding new stock items to the system, perhaps after receiving a shipment.
    - **Add Stock:** This could involve updating the inventory levels for existing stock items.
    - **Sell Stock:** This represents the process of recording a sale, deducting the sold items from the inventory.
    - **Confirm Order:** After a sale, the system confirms the order details with the customer.
    - **Make Payment:** The system processes the payment for the order.

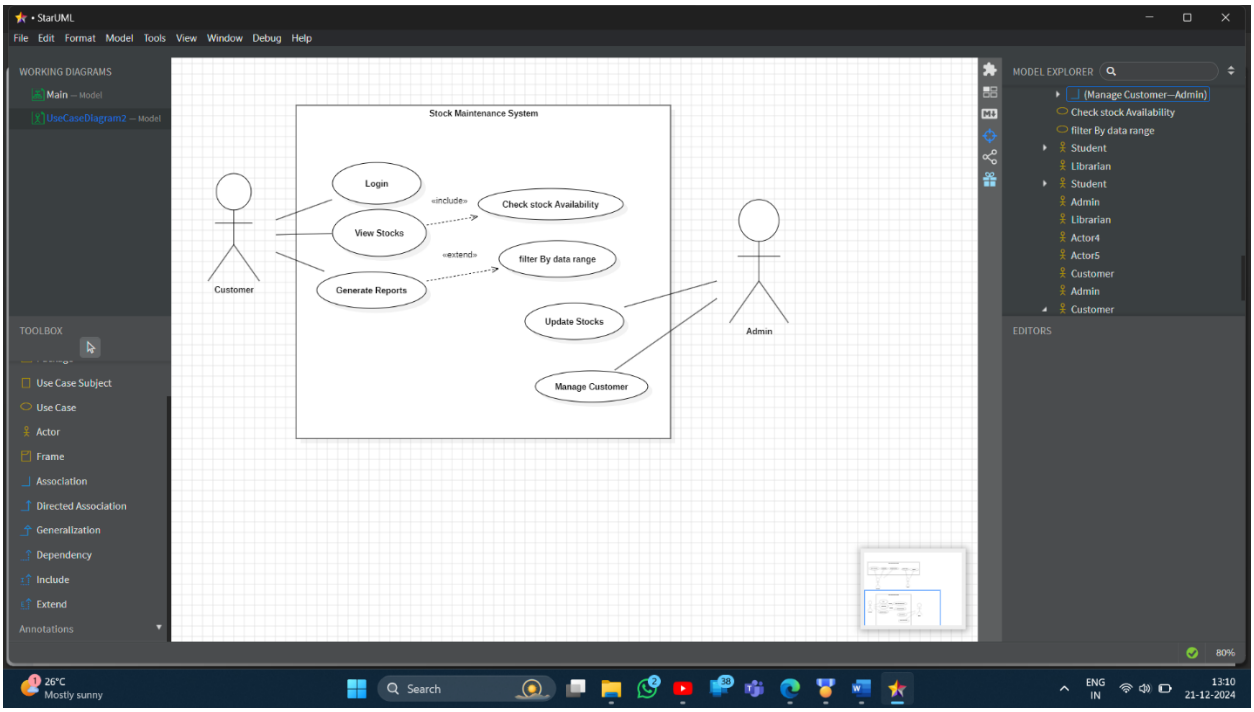
- **Print Bill:** The system generates and prints a bill for the customer.

**In essence, this diagram shows the sequence of operations within a Stock Maintenance System, highlighting the different stages of the process.**

**Key Points:**

- **Swimlanes:** The vertical columns represent the different actors or systems involved in the process (in this case, the User and the Stock Maintenance System).
- **Arrows:** The arrows indicate the flow of actions and information between the different stages.
- **Shapes:**
  - Rounded rectangles represent actions or activities.
  - Circles represent the start or end of the process.

## 4.5 Use Case Diagram:




Certainly, let's break down this diagram in a simpler way.

**What is this diagram about?**

This is a use case diagram, a visual representation used in software engineering to illustrate the interactions between users (actors) and the system. In this case, it illustrates the different actions that can be performed in a stock maintenance system.

**Here's a simplified explanation:**

**Actors:**

- **Customer:** Represents a user who interacts with the system, possibly to check stock availability or place orders.
- **Admin:** Represents an administrator with higher privileges, responsible for managing the system.

**Use Cases:**

- **Login:** Users (customers or admins) need to log in to access the system.
- **Check Stock Availability:** Users can check the availability of products in stock.
- **View Stocks:** Users can view a list of available products.
- **List Stocks:** This use case likely involves generating a list of products in stock, possibly with details like quantity and price.
- **Generate Reports:** Admins can generate reports on stock levels, sales trends, etc.
- **Manage Customers:** Admins can manage customer accounts and information.
- **Filter By Data Range:** This use case allows users to filter stock data based on a specific date range.

**Relationships:**

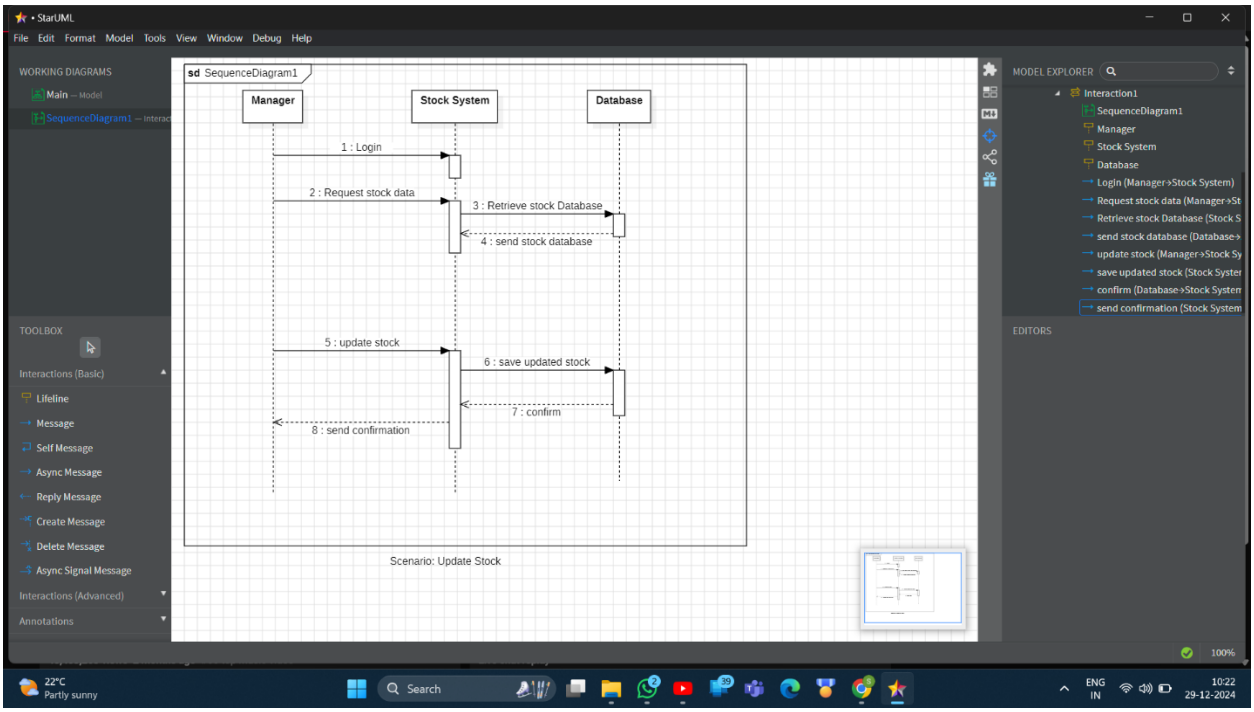
- **Include:** Indicates that one use case includes the functionality of another. For example, "List Stocks" might include "Filter By Data Range."

**In essence, this diagram shows the different actions that customers and admins can perform in a stock maintenance system.**

**Key Points:**

- **Actors:** The stick figures represent the users (actors) interacting with the system.
- **Use Cases:** The ovals represent the different actions that can be performed.
- **Relationships:** The lines and arrows indicate the relationships between use cases.

## 4.6 Sequence Diagram:



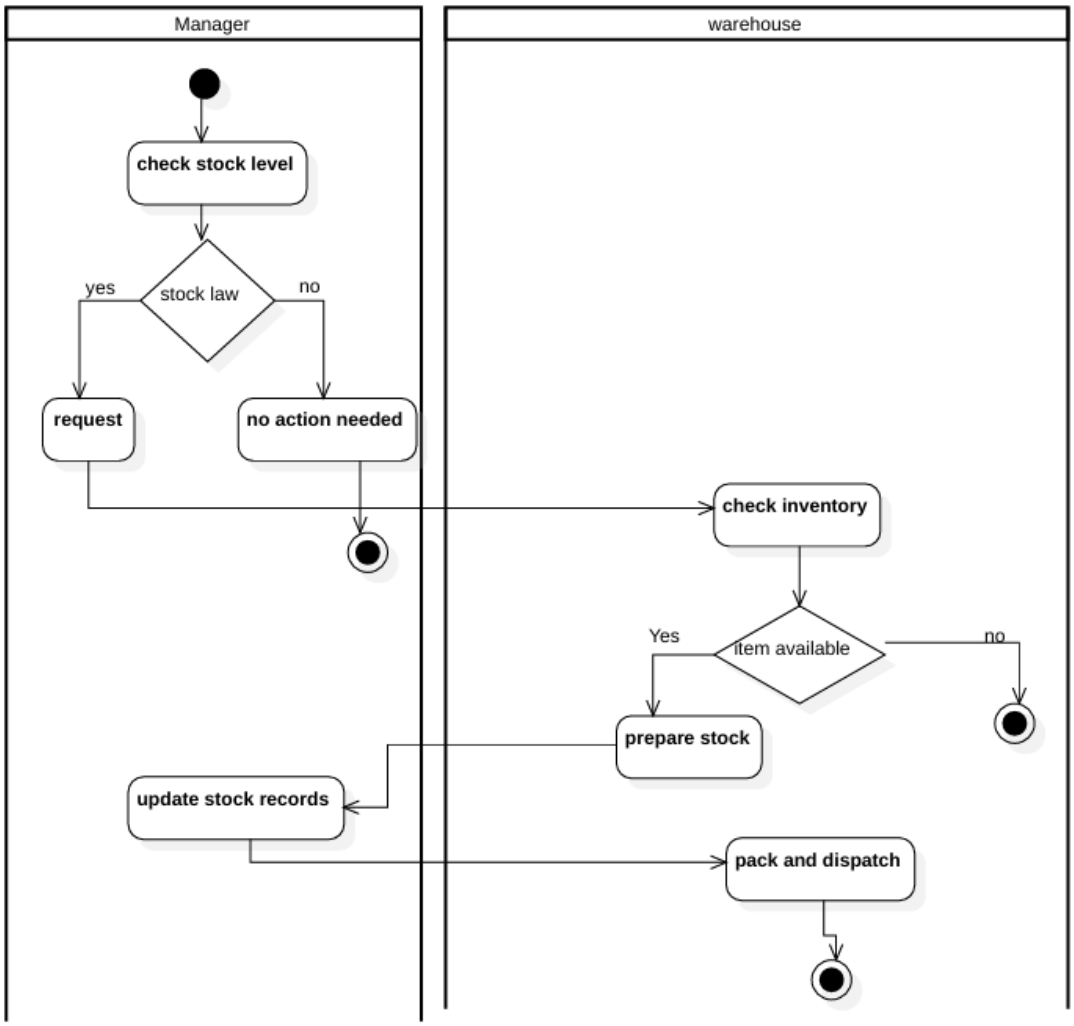

1. **The Manager logs into the system.**
2. **The Manager requests the stock data from the Stock System.**
3. **The Stock System retrieves the stock data from the Database.**
4. **The Stock System sends the stock data back to the Manager.**
5. **The Manager updates the stock data.**
6. **The Stock System saves the updated stock data in the Database.**
7. **The Database confirms the update to the Stock System.**
8. **The Stock System sends a confirmation message to the Manager.**

**In essence, this diagram shows the flow of communication and actions between the Manager, Stock System, and Database during the process of updating stock data.**

**Key Points:**

- **Lifelines:** The vertical lines represent the different components involved (Manager, Stock System, Database).
- **Messages:** The arrows represent the messages exchanged between these components.
- **Time:** The horizontal axis represents time, showing the sequence of events.

## 4.7 Activity Diagram:




Certainly, let's break down this diagram in a simpler way.

**What is this diagram about?**

This is a **swimlane diagram**, a type of visual representation used to show the sequence of actions performed by different actors or systems. In this case, it illustrates the process of fulfilling a stock order.

**Here's a simplified explanation:**

1. **Manager:**
    - The process starts with the **Manager** checking the **stock level** of a particular item.
2. **Decision Point:**

    - If the **stock level** is sufficient to fulfill the order (meets the **stock law**), the Manager proceeds to **request** the item from the warehouse.
    - If the **stock level** is insufficient, **no action** is needed as the order cannot be fulfilled immediately.
3. **Warehouse:**
    - The warehouse receives the **request** for the item.
    - The warehouse checks its **inventory** to see if the item is available.
4. **Decision Point:**
    - If the **item is available** in the warehouse, the warehouse proceeds to **prepare** the item for shipment.
    - If the **item is not available**, the process ends.
5. **Warehouse:**
    - The warehouse **packs and dispatches** the item to the customer.

**In essence, this diagram shows the flow of events and decision points involved in fulfilling a stock order, highlighting the interactions between the Manager and the Warehouse.**

**Key Points:**

- **Swimlanes:** The vertical columns represent the different actors or systems involved in the process (in this case, the Manager and the Warehouse).
- **Arrows:** The arrows indicate the flow of actions and information between the different stages.
- **Shapes:**
    - Rounded rectangles represent actions or activities.
    - Diamonds represent decision points.
    - Circles represent the start or end of the process.

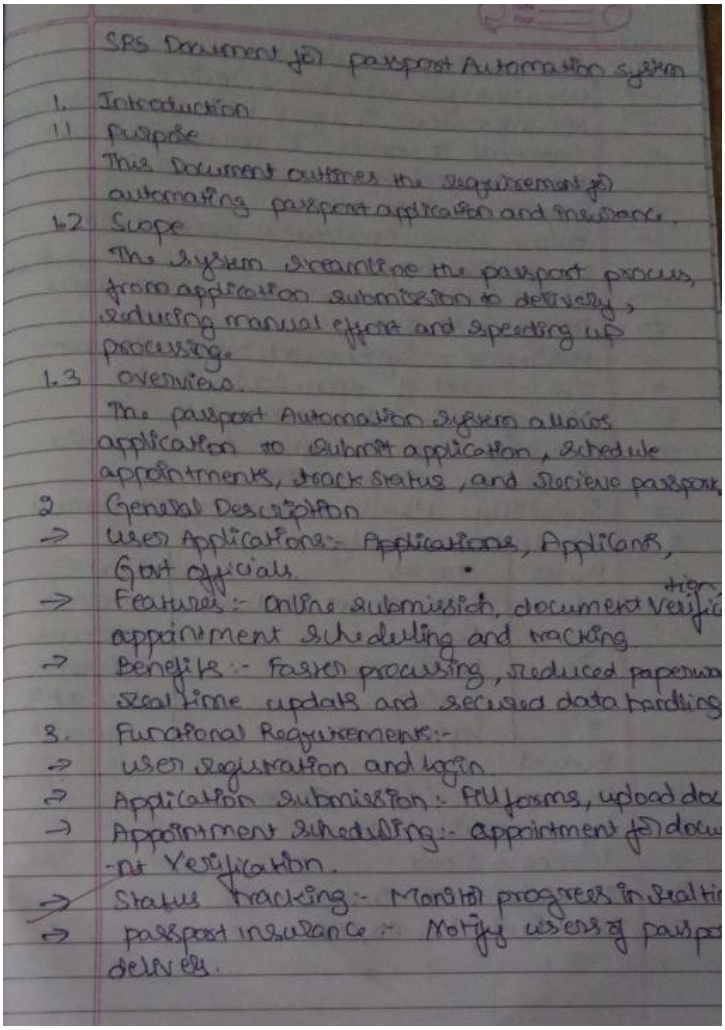
# 5.Passport Automation System:

## 5.1 Problem Statement:

Traditional passport application and issuance processes often involve manual paperwork, long queues, and multiple visits to government offices. This can lead to:

- **Delays and Frustration:** Time-consuming procedures cause significant inconvenience to applicants.
- **Human Error:** Manual data entry increases the risk of errors and delays in processing.
- **Lack of Transparency:** Limited visibility into the application status can cause anxiety and uncertainty for applicants.
- **Security Risks:** Manual handling of sensitive documents increases the risk of fraud and data breaches.

## 5.2 SRS Document:



SRS Document for passport Automation system

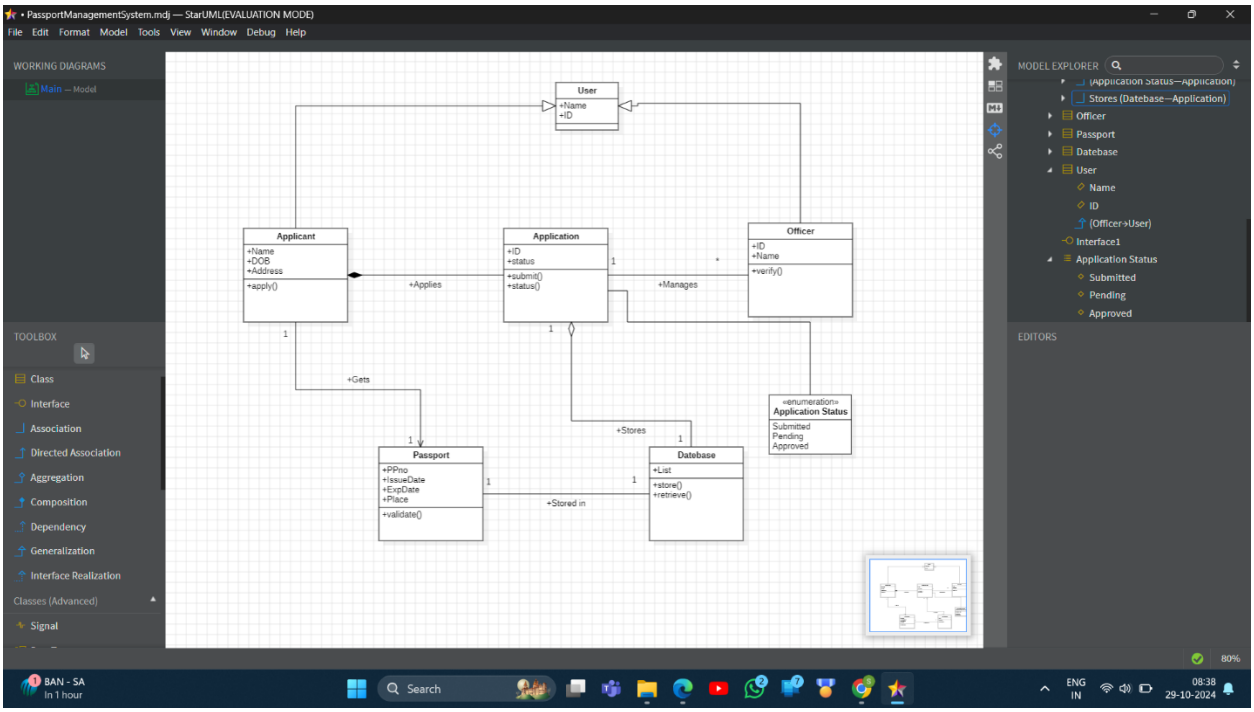
1. Introduction
1.1 purpose
This Document outlines the requirement for automating passport application and renewal.
1.2 Scope
The system streamline the passport process, from application submission to delivery, reducing manual effort and speeding up processing.
1.3 overview.
The passport Automation system allows application to submit application, schedule appointments, track status, and receive passport
2 General Description
→ User Applications:- Applicants, Applicants, Govt officials
→ Features:- Online submission, document Verifi-tion, appointment scheduling and tracking
→ Benefits:- Faster processing, reduced paperwo real time updats and secured data handling
3. Functional Requirements:-
→ user registration and login
→ Application submission:- Fill forms, upload doc
→ Appointment scheduling:- appointment for docu-nt Verification.
→ Status tracking:- Monitor progress in real ti
→ passport insurance:- Notify users of passp delivery.

4. Interface Requirements
→ User Interface :- Web-based, Intuitive
→ Api integration :- Integrate with govt database
→ Database Interface :- To communicate with Database and front end
5. Performance Requirement
→ Process time :- Application processed within 24 hours.
→ System load. Support up to 10,000 users concurrently
6) Design Constraints :-
→ Hardware :- System web server, 8 Gb ram
→ S/W :- used of open-source tools.
7) Non-functional Requirements :-
→ Security :- Data encryption and decryption
→ Reliability :- 99% uptime.
→ Scalability :- Handle increasing user demand.
8. Schedule and Budget

| Phases | Tasks | Weeks |
|---|---|---|
| 1 | Requirement Analysis | 2 weeks |
| 2 | Design | 3 weeks |
| 3 | Development | 8 weeks |
| 4 | Testing | 4 weeks |
| 5 | Deployment | 2 weeks |

Budget
→ Development Costs :- 50,000
→ Infrastructure :- 10,000
→ Testing :- 5,000
→ Miscellaneous :- 5,000
Total Budget :- 70,000

## 5.3 Class Diagram:




**Entities:**

- **User:** Represents a person who interacts with the system, typically as an applicant or an officer.
- **Applicant:** Represents an individual applying for a passport. It has attributes like Name, DOB, and Address.
- **Application:** Represents a passport application submitted by an applicant. It has attributes like ApplicationID, Status, and Date.
- **Officer:** Represents a government official responsible for processing passport applications.
- **Passport:** Represents the issued passport with attributes like PassportID and IssueDate.
- **Database:** Represents the central repository storing all the information related to users, applications, passports, and officers.

**Relationships:**

- **1:1:** A one-to-one relationship, meaning one instance of one entity is associated with only one instance of another entity. For example, an Applicant can have one Application.
- **1:N:** A one-to-many relationship, meaning one instance of one entity can be associated with multiple instances of another entity. For example, an Officer can manage multiple Applications.

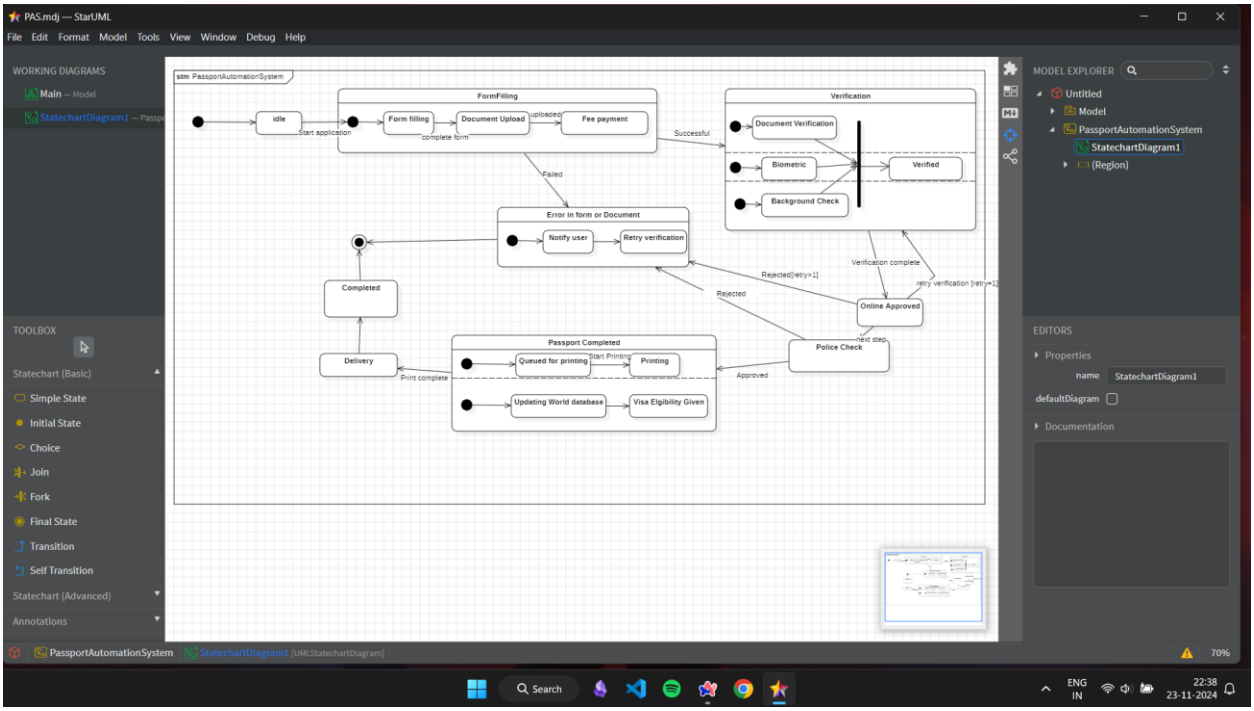
- **N:M:** A many-to-many relationship, meaning multiple instances of one entity can be associated with multiple instances of another entity. For example, a User can be both an Applicant and an Officer.

**In essence, this diagram shows the different components of a passport application system (like applicants, applications, officers) and how they are related to each other.**

**Key Points:**

- **Boxes:** Represent the different classes (entities).
- **Lines:** Represent the relationships between classes.
- **Multiplicity:** The numbers next to the lines indicate the cardinality of the relationship (e.g., 1:1, 1:N, N:M).

## 5.4 State Diagram:




- **Initial State:** The application process starts here.
- **Form Filling:** The applicant fills out the passport application form.
- **Document Verification:** The submitted documents are verified for authenticity and completeness.
- **Police Check:** A police clearance check is conducted.
- **Visa Eligibility:** The applicant's eligibility for a visa is assessed.

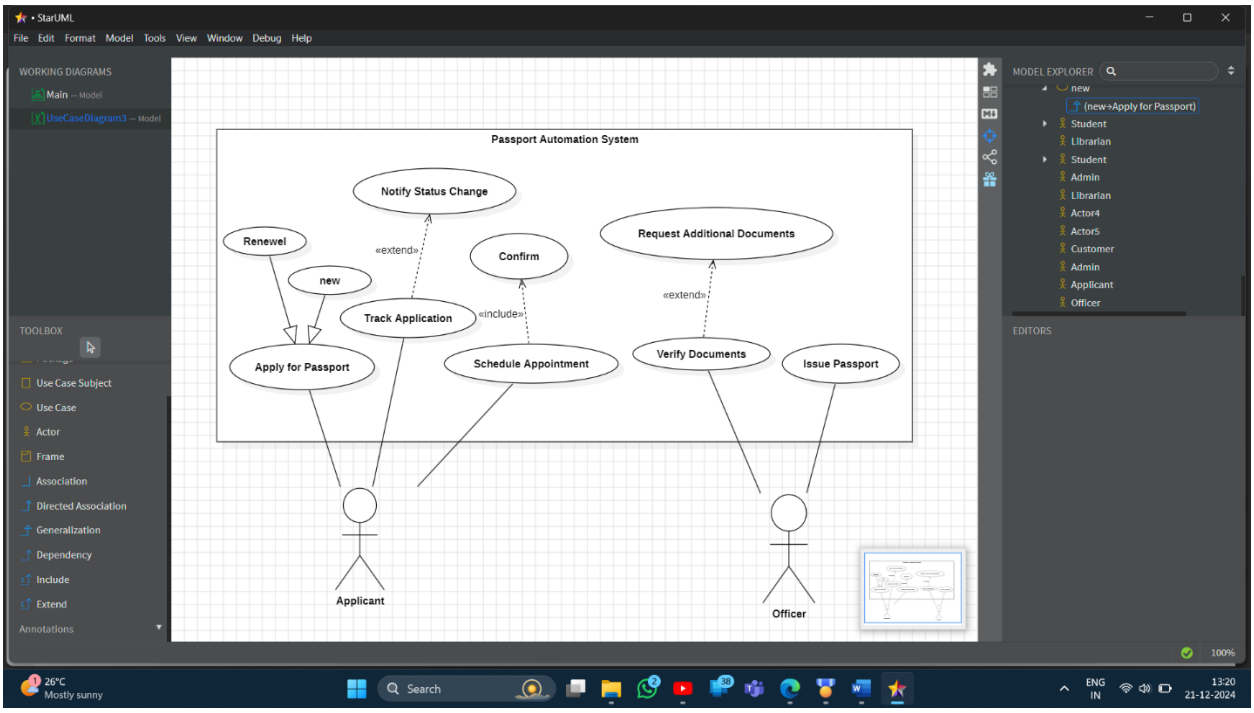
- **Delay:** If any issues arise during verification or eligibility checks, the process might be delayed for further investigation or clarification.
- **Approved:** If all checks are clear, the application is approved.
- **Rejected:** If the application is rejected due to any issues, the process ends at the Rejected state.
- **Delivery:** Once approved, the passport is printed and delivered to the applicant.

**In essence, this diagram shows the various stages and possible outcomes of a passport application process, from the initial form submission to the final delivery of the passport.**

**Key Points:**

- **States:** The rounded rectangles represent different states in the process.
- **Transitions:** The arrows represent the transitions between states, triggered by specific events or conditions.

## 5.5 Use Case Diagram:




**Actors:**

- **Applicant:** Represents an individual who applies for a passport.
- **Officer:** Represents a government official responsible for processing passport applications.

**Use Cases:**

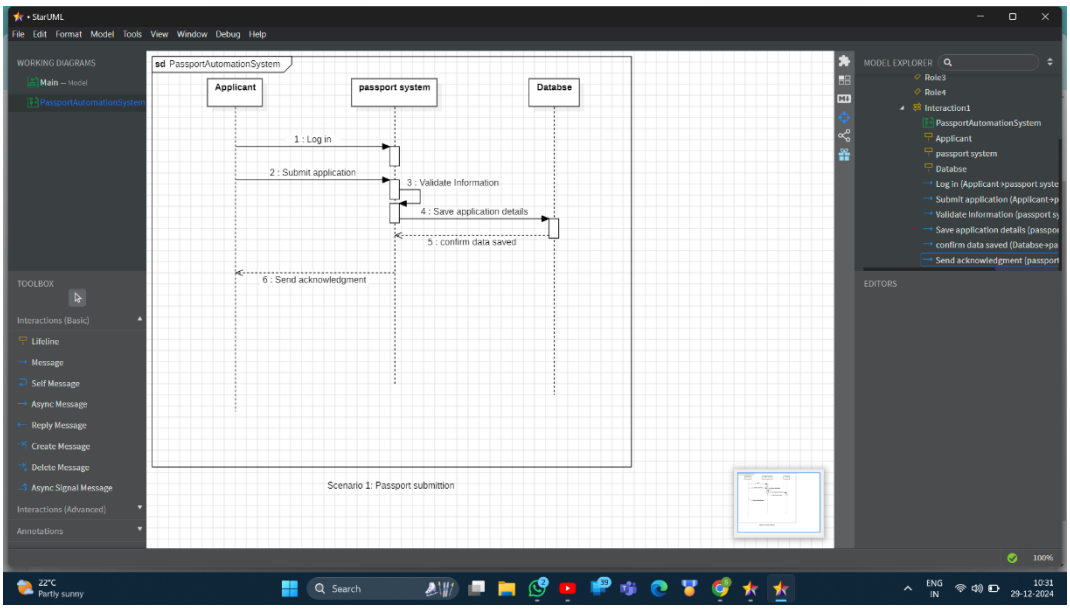
- **Apply for Passport:** The applicant submits a passport application.
- **Schedule Appointment:** The applicant can schedule an appointment to submit documents or for passport collection.
- **Track Application:** The applicant can track the status of their passport application.
- **Verify Documents:** Officers verify the documents submitted by the applicant.
- **Issue Passport:** Officers issue the passport to the applicant.
- **Request Additional Documents:** Officers may request additional documents from the applicant if necessary.
- **Notify Status Change:** The system notifies the applicant about any changes in the application status.
- **Renewel:** Applicants can renew their passports.

**In essence, this diagram shows the different actions that applicants and officers can perform in a passport automation system.**

**Key Points:**

- **Actors:** The stick figures represent the users (actors) interacting with the system.
- **Use Cases:** The ovals represent the different actions that can be performed.
- **Relationships:** The lines and arrows indicate the relationships between use cases. For example, the "Track Application" use case is related to the "Apply for Passport" use case.

## 5.6 Sequence Diagram:



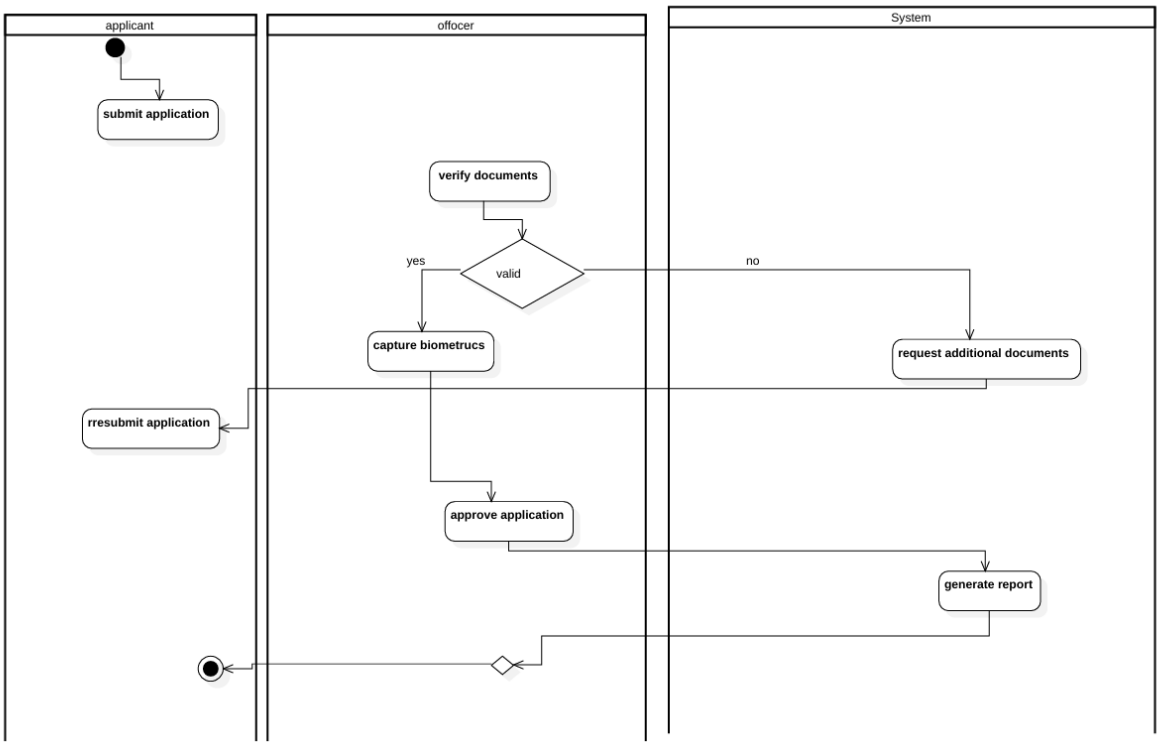
1. **The Applicant logs in to the passport system.**
2. **The Applicant submits their passport application.**
3. **The passport system validates the information provided in the application.**
4. **The passport system saves the application details.**
5. **The passport system confirms that the data has been saved successfully and sends an acknowledgment to the Applicant.**

**In essence, this diagram shows the flow of communication and actions between the Applicant and the passport system during the initial stages of a passport application.**

**Key Points:**

- **Lifelines:** The vertical lines represent the different components involved (Applicant, Passport System, Database).
- **Messages:** The arrows represent the messages exchanged between these components.
- **Time:** The horizontal axis represents time, showing the sequence of events.

## 5.7 Activity Diagram:



1. **Applicant:** The process starts with the **Applicant** submitting their passport application.
2. **Officer:** The **Officer** receives the application and verifies the submitted documents.
3. **Decision Point:**
    - If the documents are **valid**, the Officer proceeds to **capture biometrics** of the applicant.
    - If the documents are **not valid**, the Officer requests **additional documents** from the Applicant.
4. **Applicant:** If additional documents are requested, the Applicant **resubmits the application** with the required documents.
5. **Officer:** After all necessary documents are verified and biometrics are captured, the Officer **approves the application**.
6. **System:** The **System** generates a report based on the approved application.

**In essence, this diagram shows the flow of events and decision points involved in the passport application process, emphasizing the roles and actions of the Applicant, Officer, and the System.**

**Key Points:**

- **Swimlanes:** The vertical columns represent the different actors or systems involved in the process (in this case, Applicant, Officer, and System).
- **Arrows:** The arrows indicate the flow of actions and information between the different stages.
- **Shapes:**
    - Rounded rectangles represent actions or activities.
    - Diamonds represent decision points.
    - Circles represent the start or end of the process.