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:

AGNEYA D A

1BM22CS024

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 Analysis and Design(22CS6PCSEO) laboratory has been carried out by AGNEYA D A (1BM22CS024) during the 5th Semester Oct24-Jan2025.

Signature of the Faculty Incharge:

NAME OF THE FACULTY: Sandhya A Kulkarni

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

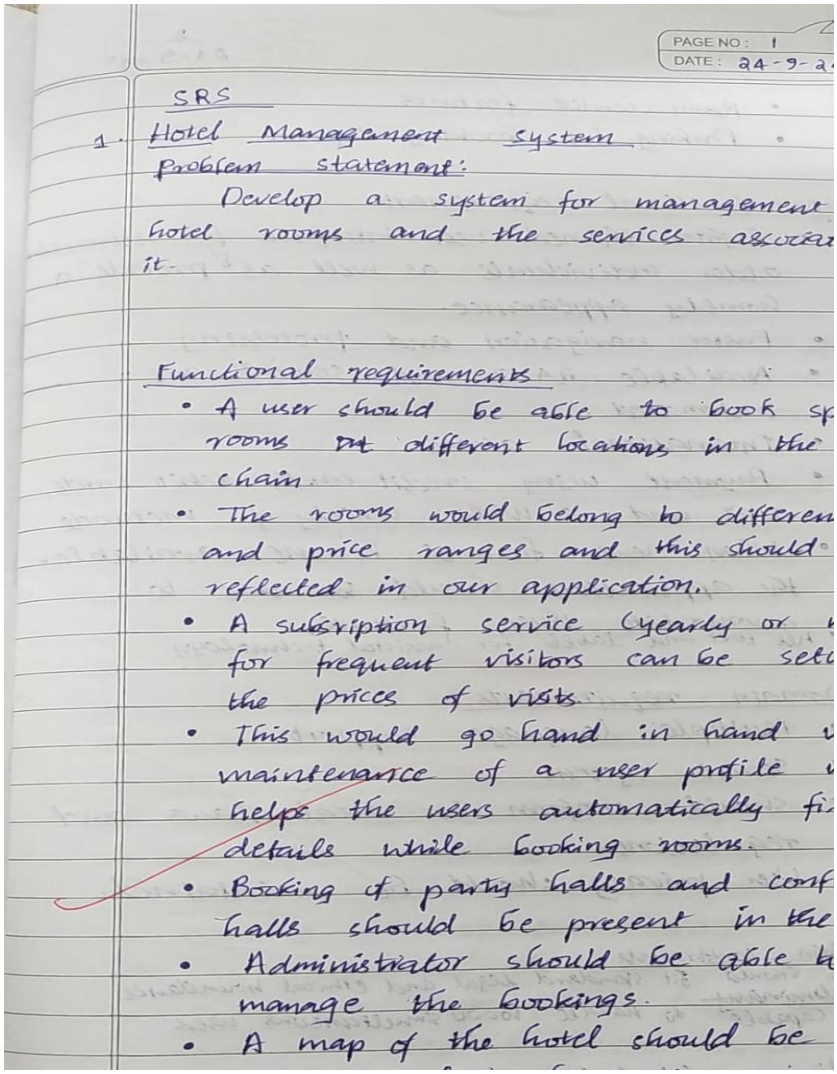
Table of Contents

1. Hotel Management System

Problem statement

In the rapidly evolving hospitality industry, managing hotel operations involves handling multiple complex tasks, including room reservations, guest check-ins and check-outs, housekeeping, maintenance, billing, and guest services. Traditional methods are often inefficient, prone to errors, and time-consuming. Therefore, there is a growing need for a robust Hotel Management System (HMS) that automates and centralizes these functions, leading to improved efficiency, accuracy, and guest satisfaction.

Software Requirements Specification



PAGE NO : 1
DATE : 24-9-2

SRS

1. Hotel Management system

Problem statement:

Develop a system for management hotel rooms and the services associa it.

Functional requirements

- A user should be able to book sp rooms at different locations in the chain.
- The rooms would belong to differen and price ranges and this should reflected in our application.
- A subscription service (yearly or for frequent visitors can be set the prices of visits.
- This would go hand in hand maintenance of a user profile helps the users automatically fi details while booking rooms.
- Booking of party halls and conf halls should be present in the
- Administrator should be able t manage the bookings.
- A map of the hotel should be

PAGE NO :
DATE : 2

- Room service features.
- Parking lot booking system.

Non-functional requirements

- Lighter themed user-interface f
  older individuals as well as to
  homely appearance.
- Faster navigation and processi
- Available 24×7 and services
  automated.
- International access.
- Payment using credit cards, d
  UPI and a wide variety of
- When newer features become
  the application should be able
  accomodate these features.
- Alt text and labels for assisted technol

Domain requirements

- Multiple language support.
- Secure system.
- Should conform to local le
  regulations.
- User privacy should be maint

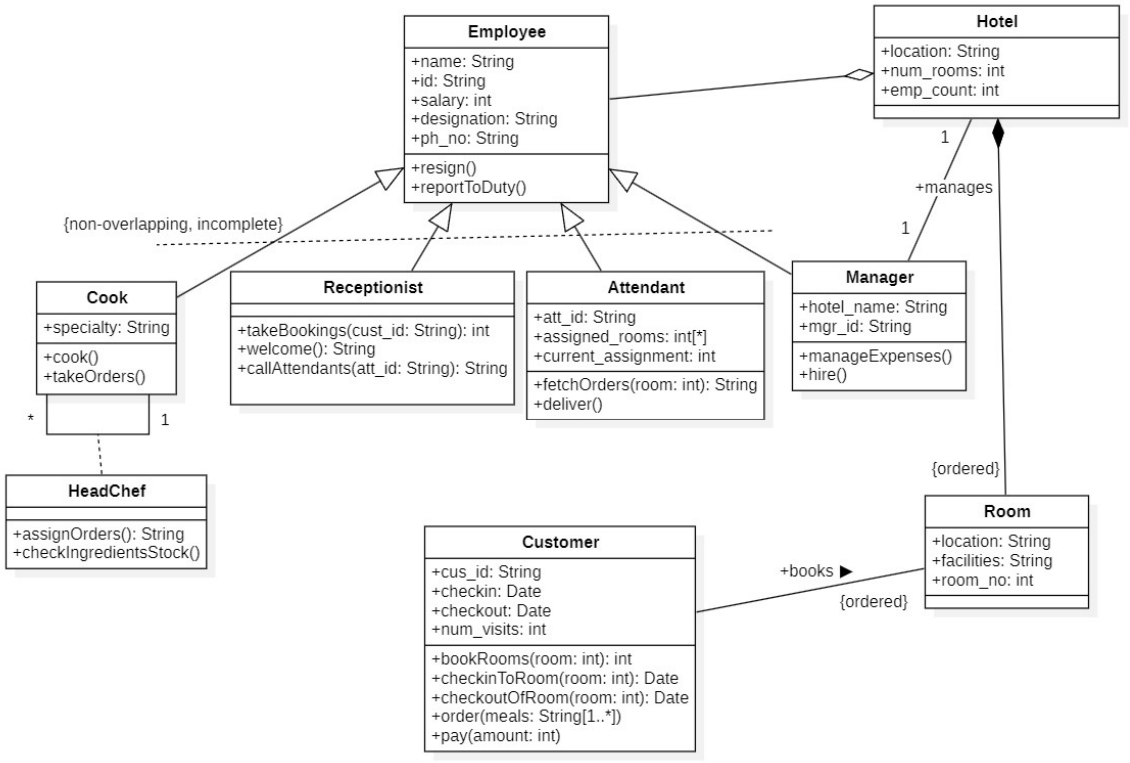
Design constraints

* Should fit standard legal and ethical bou

~~Preliminary~~

* Capable to handle 10000 simultaneous u

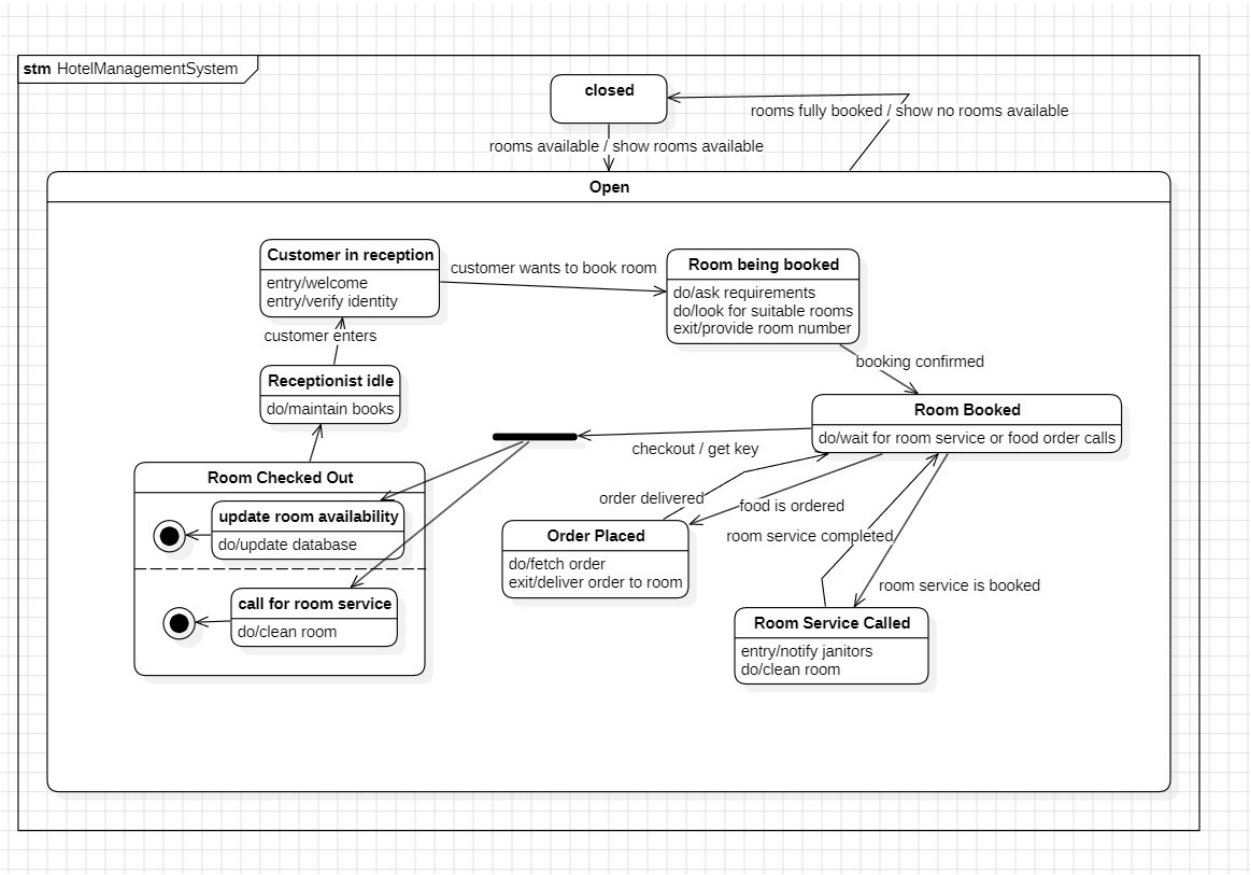
Class Diagram



The class diagram represents the comprehensive structure of a hotel management system, detailing various entities and their relationships. The Employee class serves as the base class with attributes such as name, ID, salary, designation, and phone number, and methods like resign and reportToDuty. It is extended by specialized employee types, including Cook, Receptionist, Attendant, and Manager. The Cook class includes a specialty attribute and methods like cook and takeOrders, and it has an aggregation relationship with HeadChef, indicating that a head chef can manage multiple cooks. The Receptionist class, tasked with handling bookings and welcoming guests, directly inherits from Employee. Attendant has attributes like attendant ID and assigned rooms, with methods to fetch and deliver orders, and it also inherits from Employee. The Manager class, representing the managerial role, includes attributes for hotel name and manager ID and methods for managing expenses and hiring employees, and it is aggregated with the Hotel class, signifying that each hotel has one manager.

The Hotel class, representing the central entity, contains attributes like location, number of rooms, and employee count, and is aggregated with both Manager and Room classes, indicating that a hotel has one manager and multiple rooms. Each Room is characterized by its location, facilities, and room number, and it shares an association with the Customer class, reflecting that customers can book multiple rooms. The Customer class includes customer ID, check-in and check-out dates, and number of visits, with methods to book rooms, check in and out, order meals, and make payments. This structured representation elucidates the intricate relationships and dependencies among various components, ensuring efficient management and seamless operation of hotel services.

State Diagram



The state diagram visually represents the various states and transitions in a Hotel Management System, particularly focusing on the process flow related to managing hotel room bookings, guest services, and checkouts. The diagram is divided into two primary states: "closed" and "open." When the system is in the "closed" state, it signifies that rooms are either fully booked or

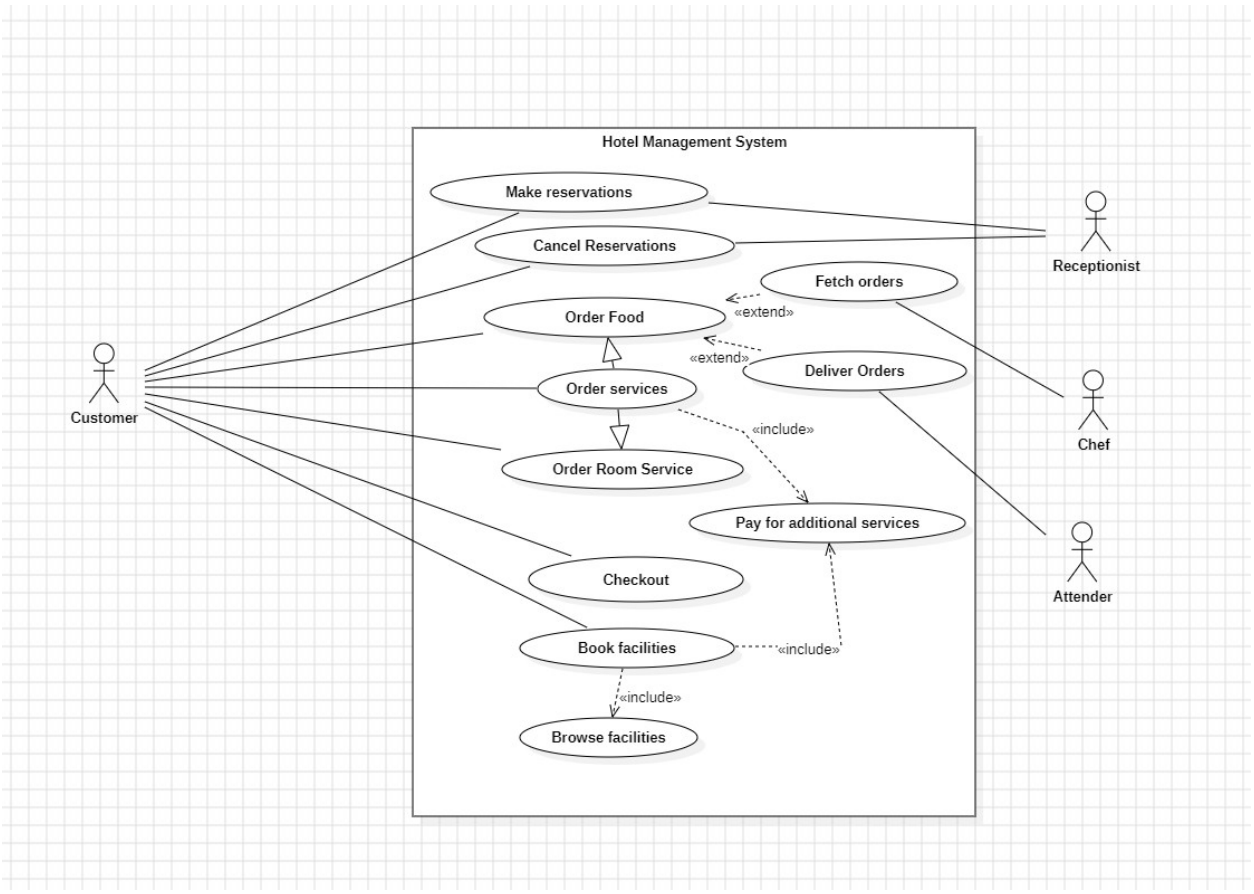
unavailable for some reason, preventing any new bookings. Conversely, the "open" state indicates the availability of rooms and the initiation of the booking process. The flow begins with a customer entering the reception, where their identity is verified, leading to the state "Customer in reception."

Following this, the system transitions to "Room being booked," where the room booking process occurs, including checking room availability and confirming the reservation. Once a room is successfully booked, the state transitions to "Room Booked." Here, various actions are taken, such as maintaining records and updating the database with the booking details.

If the guest places an order for room service, the state shifts to "Order Placed." This is followed by the "Room Service Called" state, where the service staff is notified of the order. Actions in this state include fetching the order and delivering it to the guest's room. The final significant state in the process is "Room Checked Out," which occurs when the guest checks out of the hotel. This state involves updating the system records, cleaning the room, and making it available for the next guest.

Each state has specific actions associated with it, ensuring smooth transitions and efficient management of hotel operations. The diagram clearly illustrates the flow from booking to service delivery and finally to checkout, highlighting the systematic approach to handling hotel management tasks. This structured overview aids in understanding and managing the complex workflows within a hotel management system, ensuring enhanced efficiency and guest satisfaction.

Use Case Diagram



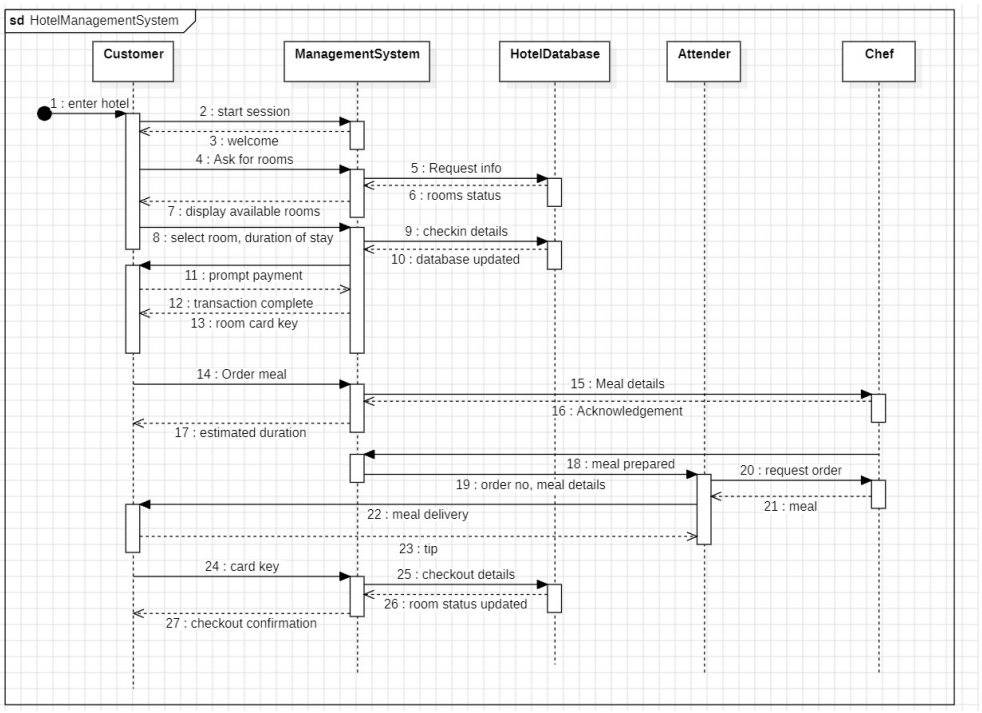
The use case diagram for the Hotel Management System provides a comprehensive overview of the interactions between various users and the system, highlighting key functionalities and processes. The diagram identifies four primary actors: Customer, Receptionist, Chef, and Attender. Each actor interacts with the system to perform specific tasks. For example, the Customer can make and cancel reservations, order food and services, request room service, check out, book facilities, and browse available facilities. The Receptionist is responsible for facilitating reservations, cancellations, and checkouts, acting as the point of contact for guests. The Chef manages the preparation of food orders, while the Attender handles the delivery of orders and room service requests.

The diagram includes several use cases, such as "Make Reservations" and "Cancel Reservations," which allow customers to book and cancel rooms. The "Order Food" and "Order Services" use cases enable customers to request meals and additional services like housekeeping and spa treatments. "Order Room Service" is a specific subtype of "Order Services," indicating a

focused interaction for in-room dining and service requests. The "Checkout" use case facilitates the guest's departure process, including bill settlement and room key return. Additionally, "Book Facilities" and "Browse Facilities" provide customers with options to reserve and view hotel amenities like conference rooms and gyms.

Relationships between use cases are also depicted, such as the inclusion of "Order Room Service" within "Order Services," signifying that room service is a specific aspect of broader service offerings. Similarly, "Order Food" extends to "Fetch Orders" and "Deliver Orders," indicating the steps involved in processing food requests. Overall, this use case diagram effectively illustrates the dynamic interactions between system components and users, ensuring a streamlined and user-friendly hotel management experience.

Sequence Diagram



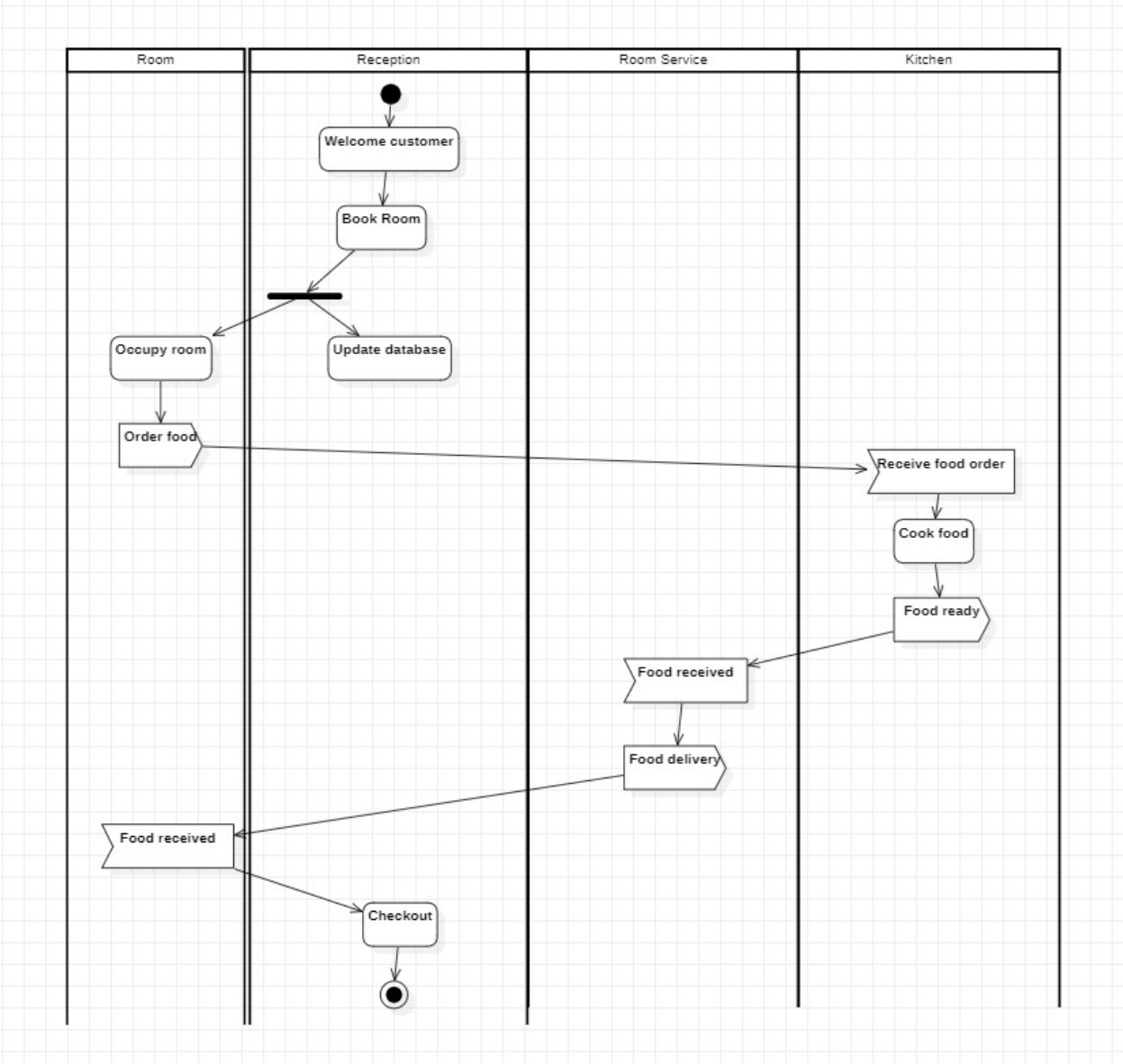
The sequence diagram for the Hotel Management System vividly portrays the interaction between different entities in managing hotel operations, focusing on tasks such as room booking, payment processing, meal ordering, and checkout. The main actors in this diagram include the

Customer, Management System, Hotel Database, Attender, and Chef. The process begins with the Customer entering the hotel and initiating a session with the Management System, which welcomes the Customer and asks for their room preferences. The system then queries the Hotel Database for room information, which responds with the current room status. The Management System displays the available rooms to the Customer, who then selects a room and specifies the duration of stay. Following this, the Management System prompts the Customer for payment, and upon successful transaction completion, provides the Customer with a room card key.

Next, the Customer orders a meal, and the Management System provides an estimated duration for meal preparation. The system sends the meal details to the Attender, who acknowledges the request and forwards it to the Chef. The Chef then prepares the meal and informs the Attender once it is ready. The Attender subsequently delivers the meal to the Customer, who may give a tip for the service.

Finally, when the Customer is ready to leave, they check out by returning the room card key to the Management System. The system updates the room status in the Hotel Database and confirms the checkout to the Customer. This sequence of interactions ensures efficient service delivery, maintaining smooth operations and enhancing customer satisfaction through well-coordinated processes. The diagram effectively illustrates the structured flow of tasks and the communication between various components, providing a clear understanding of the hotel's operational dynamics.

Activity Diagram



This activity diagram provides a structured visualization of the workflow in a hotel, detailing the interactions among different departments such as Reception, Room, Room Service, and Kitchen. The process begins at the Reception, where a customer is welcomed and the room booking process is initiated. Once the booking is confirmed, the details are updated in the database, and the customer occupies the room. The customer may then decide to order food, initiating an interaction with the Room Service. The Room Service relays the food order to the Kitchen, where the food is prepared. Upon completion, the Kitchen marks the food as ready. The Room Service then takes over to deliver the food to the customer's room. After enjoying their stay and

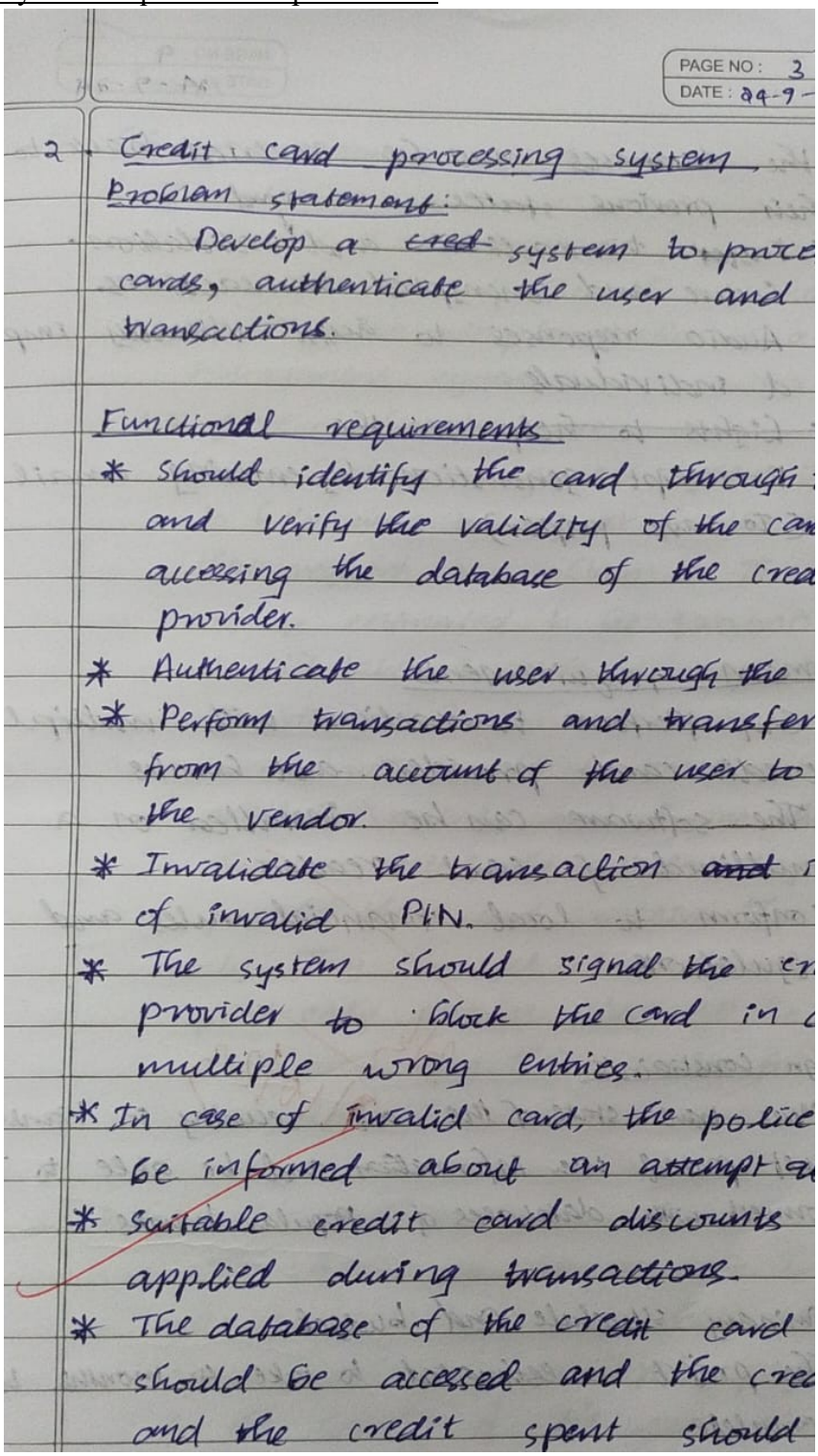
the services provided, the final step in the process is the checkout, where the customer departs from the hotel. This activity diagram effectively illustrates the seamless coordination required among various departments to ensure a smooth and satisfying experience for the customer, highlighting the importance of synchronized actions and communication in hotel management operations.

2. Credit Card Processing System

Problem Statement

In today's digital age, a robust and secure Credit Card Processing System is essential for businesses to facilitate seamless transactions, enhance customer satisfaction, and safeguard sensitive financial information. The primary objective of developing this system is to streamline the process of authorizing, capturing, and settling credit card payments, ensuring efficiency and reliability for merchants and customers alike. The system must be capable of handling high volumes of transactions with speed and accuracy, offering features such as fraud detection, data encryption, and compliance with industry standards like PCI-DSS. Additionally, it should provide real-time transaction monitoring, detailed reporting, and integration with existing financial software. By implementing a comprehensive Credit Card Processing System, businesses can not only improve their operational efficiency but also build trust and confidence among their customers, leading to increased loyalty and growth.

System Requirements Specification



PAGE NO : 3
DATE : 24-9-

2. Credit card processing system.

Problem statement:

Develop a credit system to proce cards, authenticate the user and transactions.

Functional requirements

* Should identify the card through and verify the validity of the car accessing the database of the crea provider.
* Authenticate the user through the
* Perform transactions and transfer from the account of the user to the vendor.
* Invalidate the transaction and of invalid PIN.
* The system should signal the cr provider to block the card in multiple wrong entries.
* In case of invalid card, the police be informed about an attempt a
* Suitable credit card discounts applied during transactions.
* The database of the credit card should be accessed and the cre and the credit spent should

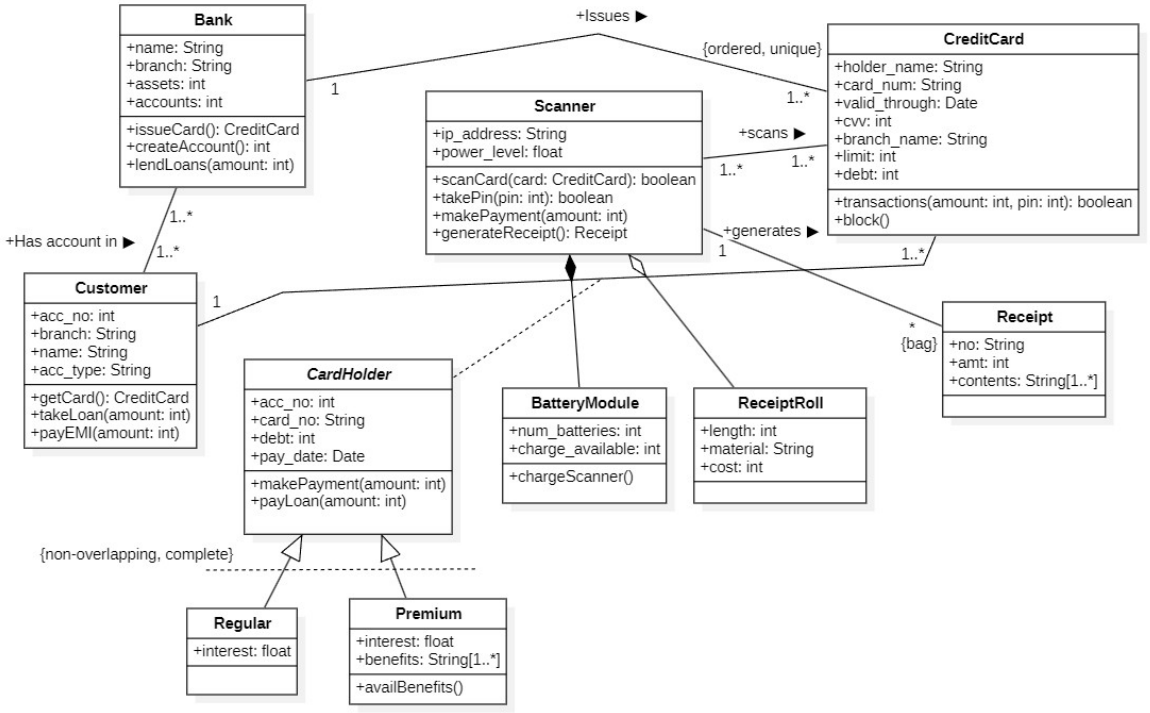
the databases must be reverted their previous states
* Faster transactions and validat
* Secure and encrypted transactio
* Audio responses to assist visua d individuals.
* ~~Lights to help see the~~
* E-receipt generation by entering (To save paper).

Domain requirements
* Can perform transactions with credit card providers and bank
* The software can be installed multitude of card readers.
* conform to local financial rule regulations.

Design constraints
* Must have state of the art security
* Must protect user information and be connect with databases of popular ba

Preliminary schedule and budget
* The project is estimated to take 4

Class Diagram



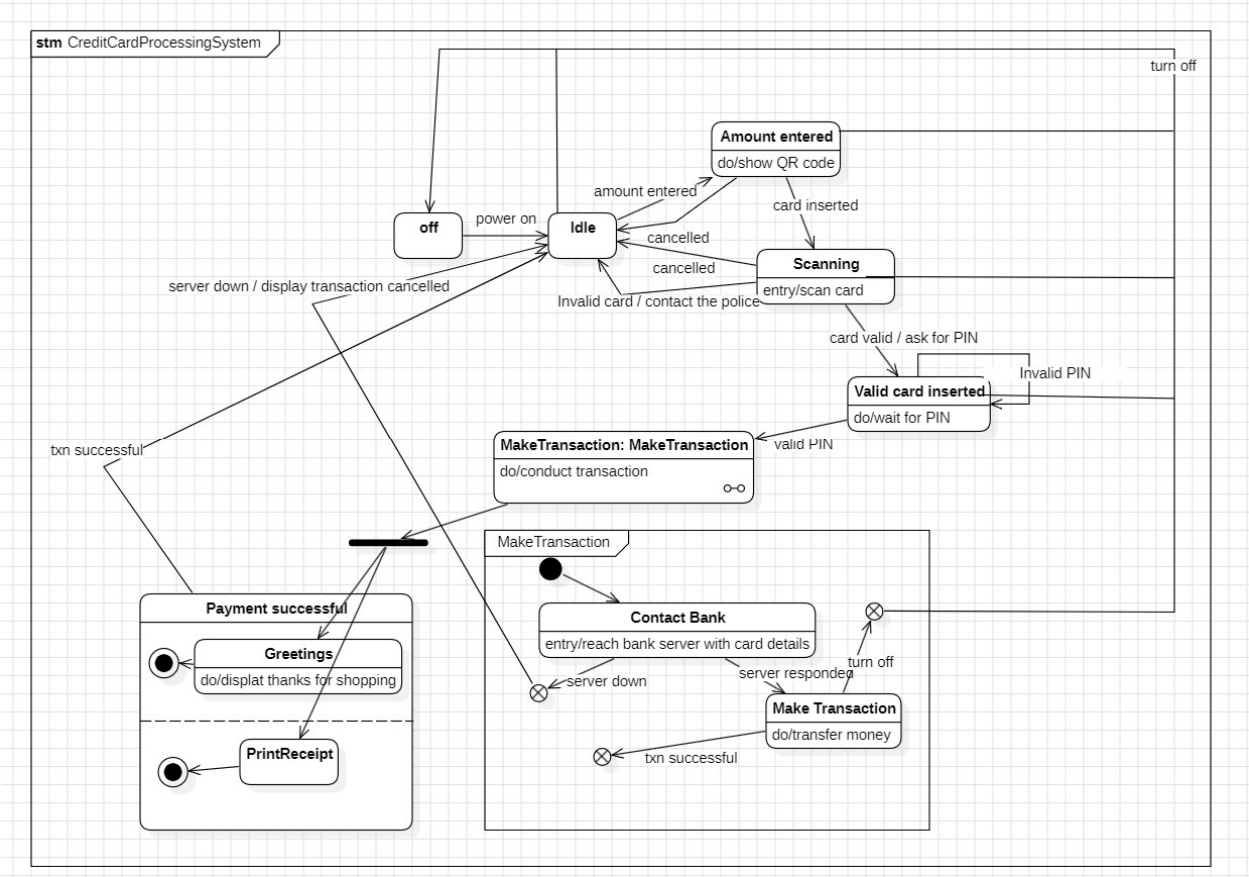
The class diagram represents the architecture of a banking system, detailing the interactions between various entities such as Bank, Customer, CreditCard, Scanner, Receipt, CardHolder, BatteryModule, and ReceiptRoll. At the core of the system is the Bank class, which includes attributes like name, branch, assets, and accounts. The Bank can issue credit cards, create accounts, and lend loans. The Customer class is associated with the Bank, and each customer can have multiple accounts. Customers can get credit cards, take loans, and pay EMIs.

The CreditCard class, which includes details like holder name, card number, validity date, CVV, branch name, limit, and debt, is associated with the Scanner class, allowing multiple credit cards to be scanned by scanners. The Scanner itself has attributes such as IP address and power level, and is capable of scanning cards, taking PINs, making payments, and generating receipts, which is illustrated by its association with the Receipt class.

The Receipt class details include number, amount, and contents, generated by the Scanner during transactions. The CardHolder class, containing attributes like account number, card number, debt, and pay date, is linked to both Regular and Premium classes, distinguishing between different types of cardholders. Regular cardholders have an interest attribute, while Premium cardholders enjoy additional benefits and have a method to avail these benefits.

Supporting the Scanner's operations, the BatteryModule class includes attributes for the number of batteries and available charge, ensuring the scanner remains functional. Similarly, the ReceiptRoll class details include length, material, and cost, representing the physical medium on which receipts are printed by the scanner.

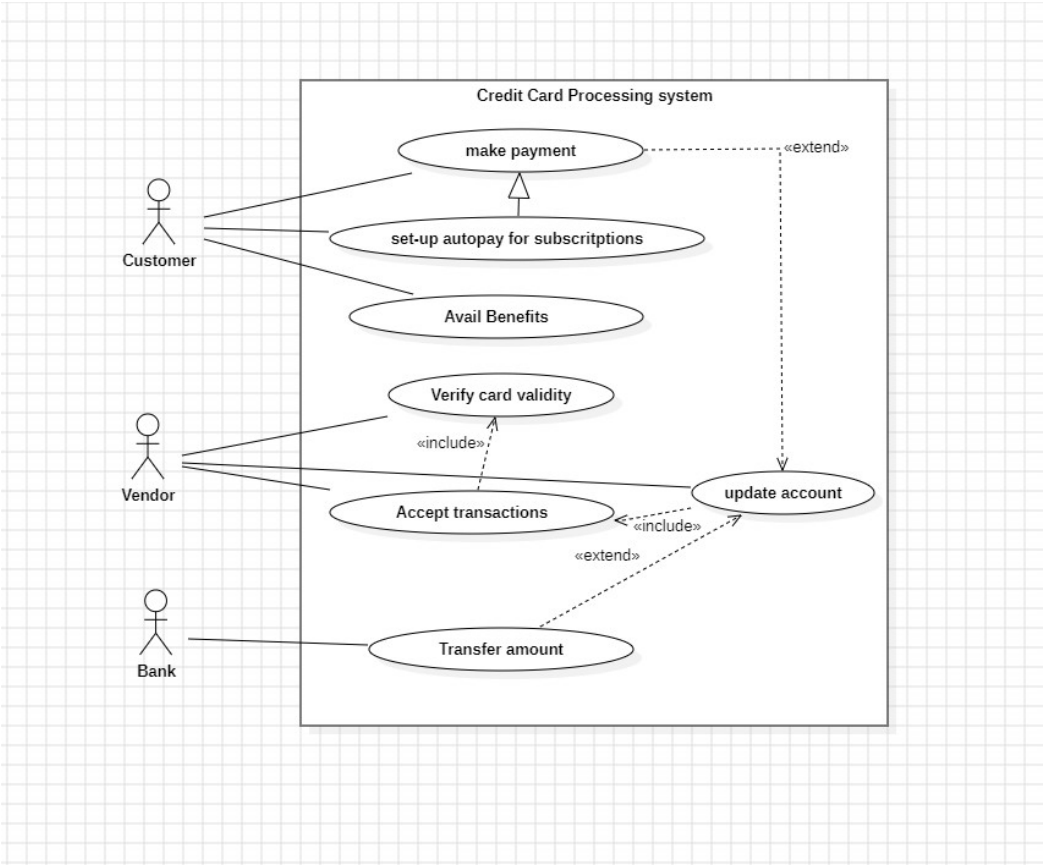
State Diagram



The state diagram illustrates the Credit Card Processing System, detailing the various states and transitions involved in handling a credit card transaction. The process begins with the system in the "off" state. Once powered on, it transitions to the "Idle" state. From here, when an amount is

entered by the user, the system displays a QR code and moves to the "Amount entered" state. If the customer inserts a credit card, the system transitions to the "Scanning" state, where the card is validated. A valid card transitions the system to the "Valid card inserted" state, prompting the user to enter a PIN. If the card is invalid, the system initiates contacting the police. Once a valid PIN is entered, the system proceeds to the "MakeTransaction: MakeTransaction" state, where the transaction is processed. This includes contacting the bank, transferring funds, and handling server responses. If the transaction is successful, the system transitions to the "Payment successful" state, displaying a thank you message and printing a receipt. If the server is down or the transaction is canceled, the system either returns to the "Idle" state or shows a transaction canceled message. This state diagram effectively maps out the logical flow and decision points of the credit card processing, ensuring clarity and efficient handling of various transaction scenarios.

Use Case Diagram



The use case diagram for the Credit Card Processing System provides a detailed visualization of the system's functionalities and interactions between different actors and use cases. The primary actors in this diagram are the Customer, Vendor, and Bank, each interacting with the system to perform specific tasks.

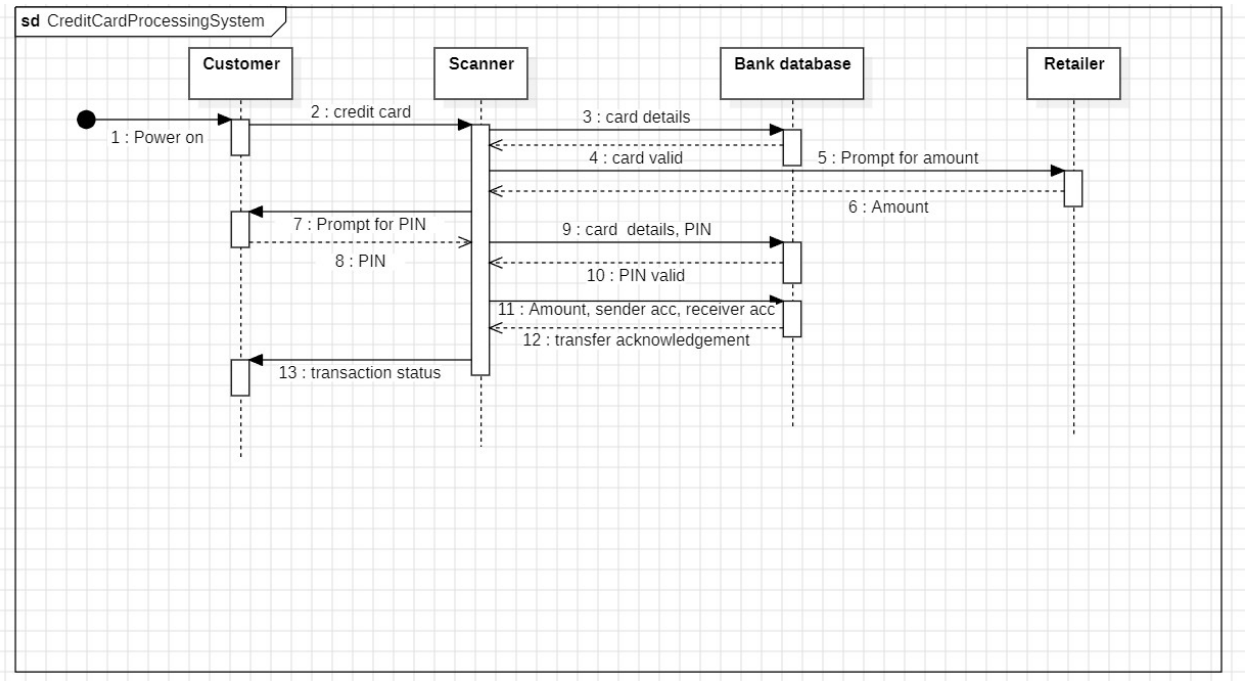
The Customer actor is involved in several use cases, including "make payment," "set-up autopay for subscriptions," and "avail benefits." The "make payment" use case extends to the "update account" use case, indicating that whenever a payment is made, the customer's account information is updated accordingly. Setting up autopay allows customers to automate their subscription payments, enhancing convenience and ensuring timely transactions.

The Vendor actor primarily interacts with the system through the "Accept transactions" use case, which includes verifying the card's validity and updating the account once the transaction is approved. This ensures that the vendor receives confirmation of payment before proceeding with the transaction.

The Bank actor is responsible for verifying card validity and transferring amounts. The "Verify card validity" use case is crucial for ensuring that the card being used for the transaction is genuine and has sufficient credit. Once the card is verified, the bank processes the transaction and transfers the required amount. The "Transfer amount" use case extends the "Accept transactions" use case, indicating that once a transaction is accepted, the amount is transferred from the customer's account to the vendor's account.

This use case diagram illustrates the system's functionalities and how different actors interact with the system to perform various tasks, ensuring secure, efficient, and reliable credit card processing. By showing these relationships and dependencies, the diagram helps in understanding and designing the system to meet the needs of all stakeholders involved.

Sequence Diagram

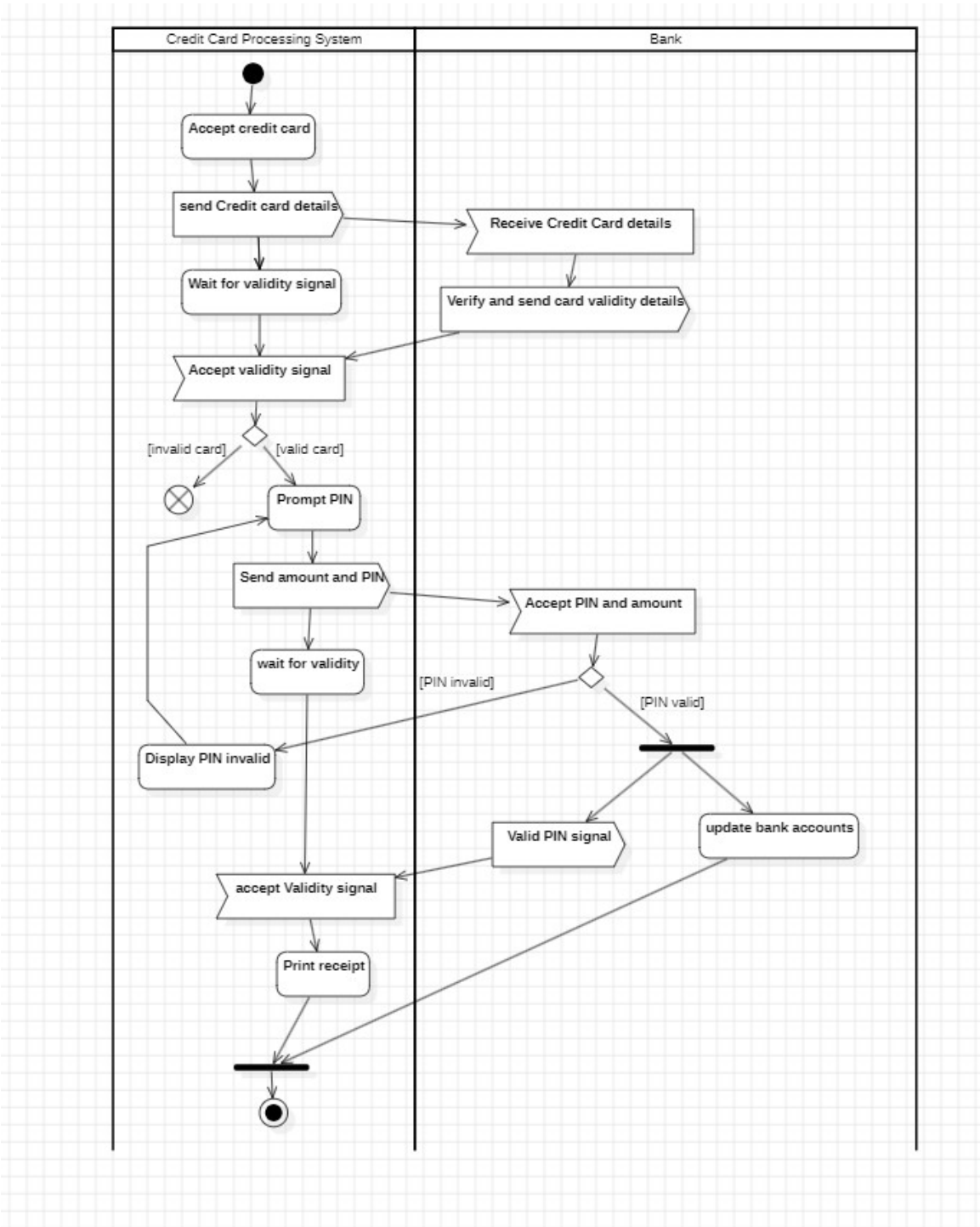


The sequence diagram for the Credit Card Processing System visually outlines the step-by-step interaction among four main entities: the Customer, Scanner, Bank Database, and Retailer, detailing the comprehensive process of a credit card transaction. The interaction begins when the Customer initiates the transaction by presenting the credit card to the Scanner. The Scanner, serving as the intermediary, sends the card details to the Bank Database to validate the card. Upon receiving confirmation of card validity, the Scanner requests the Retailer to provide the transaction amount. Once the Retailer inputs the amount, the Scanner then prompts the Customer to enter their PIN. The Customer's PIN is subsequently verified by the Bank Database to ensure it matches the card's security credentials.

After successful PIN validation, the Scanner forwards the transaction details, including the amount and account information, to the Bank Database to execute the fund transfer. The Bank Database processes the transaction, transferring the specified amount from the Customer's account to the Retailer's account, and sends a confirmation back to the Scanner. Finally, the Scanner communicates the transaction status to the Customer, completing the transaction process. This sequence diagram effectively illustrates the precise flow of information and validation steps required to securely process a credit card transaction, ensuring both the security of financial data and the integrity of the transaction. It highlights the critical role of each entity in

verifying, authorizing, and completing the transaction seamlessly, emphasizing the importance of coordination and secure communication in the credit card processing system

Activity Diagram



The activity diagram provides a comprehensive visualization of the credit card validation and transaction process within a Credit Card Processing System and its interaction with a Bank. The process begins with the system accepting the customer's credit card and transmitting the card details to the Bank for verification. The Bank then verifies the card details and sends back a signal indicating whether the card is valid or invalid. If the card is found to be invalid, the process terminates immediately, ensuring that no fraudulent transactions occur. If the card is valid, the system progresses to prompt the user to enter their PIN.

Upon receiving the PIN input, the system sends the entered PIN along with the transaction amount to the Bank. The Bank checks the validity of the entered PIN. If the PIN is invalid, the system displays an "Invalid PIN" message and waits for a correct PIN to be entered. This loop continues until a valid PIN is provided. Once the PIN is validated, the Bank updates the respective bank accounts to reflect the transaction, transferring the specified amount from the customer's account to the merchant's account. The Bank then sends a signal confirming the successful validation of the PIN and completion of the transaction back to the system.

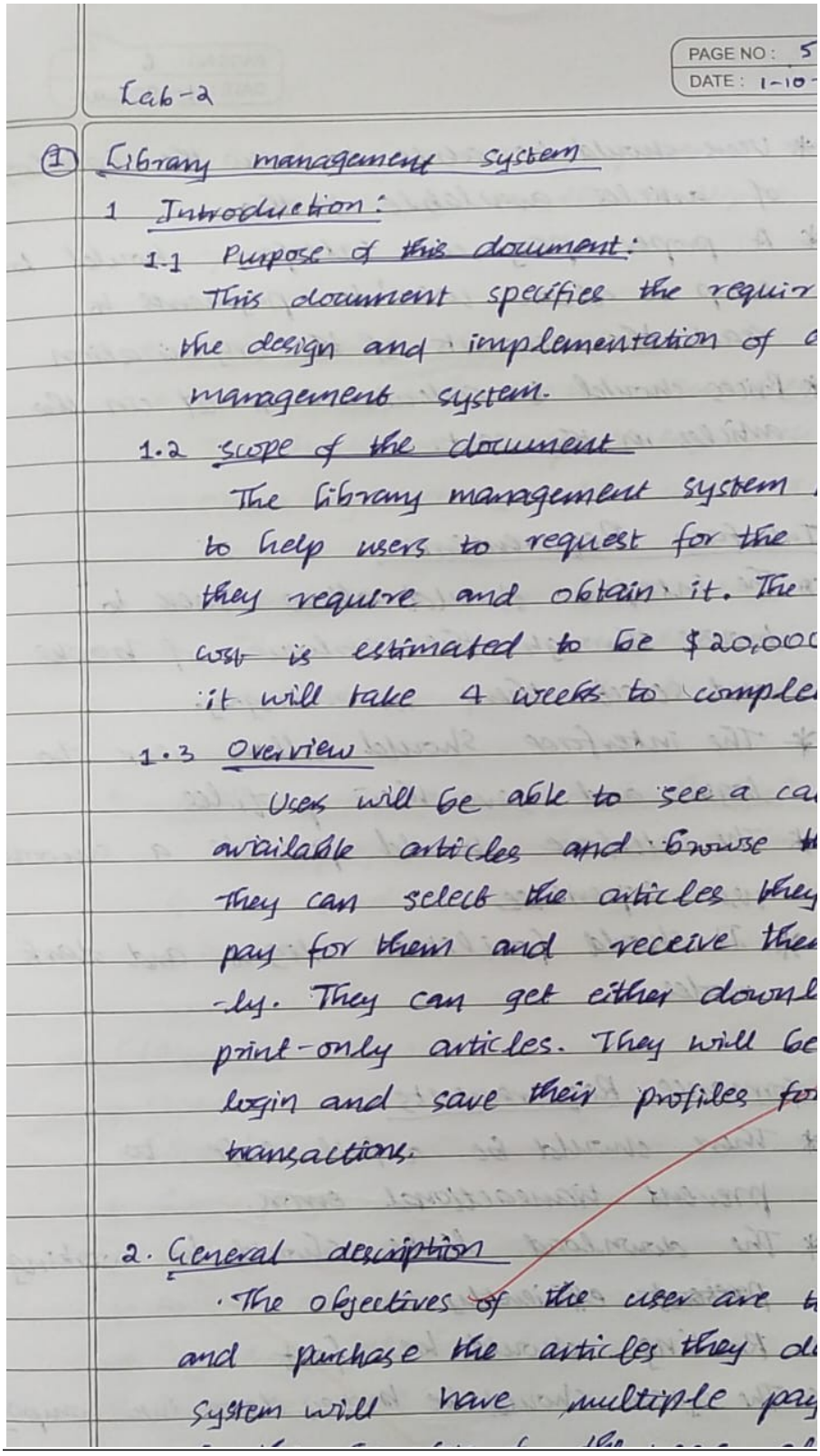
Finally, the Credit Card Processing System acknowledges the successful transaction by printing a receipt for the customer, which serves as proof of payment. This structured flow ensures that all necessary security checks are performed, transactions are accurately processed, and customers receive immediate feedback on their transaction status. The activity diagram effectively highlights the sequential steps and decision points involved in credit card processing, emphasizing the importance of verification and secure communication between the system and the Bank.

3. Library Management System

Problem Statement

In today's digital age, managing a library's vast collection of resources and facilitating efficient access to information is a significant challenge. The objective of developing a Library Management System (LMS) is to streamline the management and operations of a library, ensuring that books, journals, and other resources are easily accessible to patrons while also maintaining accurate records of inventory and transactions. The system must support functionalities such as cataloging, inventory management, user registration, borrowing and returning of items, and tracking overdue materials. Additionally, it should provide robust search capabilities, enabling users to quickly locate resources. Automated notifications for due dates, fines, and reserved item availability can enhance user experience and operational efficiency. Furthermore, the system should offer reporting and analytics features to help library administrators monitor usage patterns, manage acquisitions, and optimize resource allocation. By implementing an efficient LMS, libraries can improve service delivery, enhance user satisfaction, and better support the educational and informational needs of their communities.

System Requirements Specification



Lab-2

PAGE NO : 5
DATE : 1-10-

(1) Library management system

1 Introduction:

1.1 Purpose of this document:

This document specifies the requir
the design and implementation of a
management system.

1.2 Scope of the document

The library management system
to help users to request for the
they require and obtain it. The
cost is estimated to be $20,000
it will take 4 weeks to comple

1.3 Overview

Users will be able to see a ca
available articles and browse th
they can select the articles they
pay for them and receive them
-ly. They can get either downl
print-only articles. They will be
login and save their profiles for
transactions.

2. General description

The objectives of the user are t
and purchase the articles they d
System will have multiple pay

* Users should be able to view the of articles available to them
* A proper payment interface sho setup to allow users' to payments reach the accounts of the organiz
* Prices should be calculated based o articles in the cart.

4. Interface Requirements
* The interface should allow user browse through the catalogue of and select them accordingly.
* The interface should allow use login and save their profiles.
* The interface should provide a user experience.
* It should facilitate light an modes.

5. Performance Requirements
* There should be a fail safe prevent transactional errors.
* The download links should be ~~present~~ efficiently
* Routing should be fast
* The system should be broken down

PAGE NO : 7
DATE : 1-10

* A lot of processing power is re
simultaneously handle multiple
operations.

7. Non-functional requirements

* The system (components) should be reusable
* It should be compatible with
operating systems like windows,
Android and ios.
* Transactions should be secure
* Data should ~~be~~ handle null
it should be possible to modify

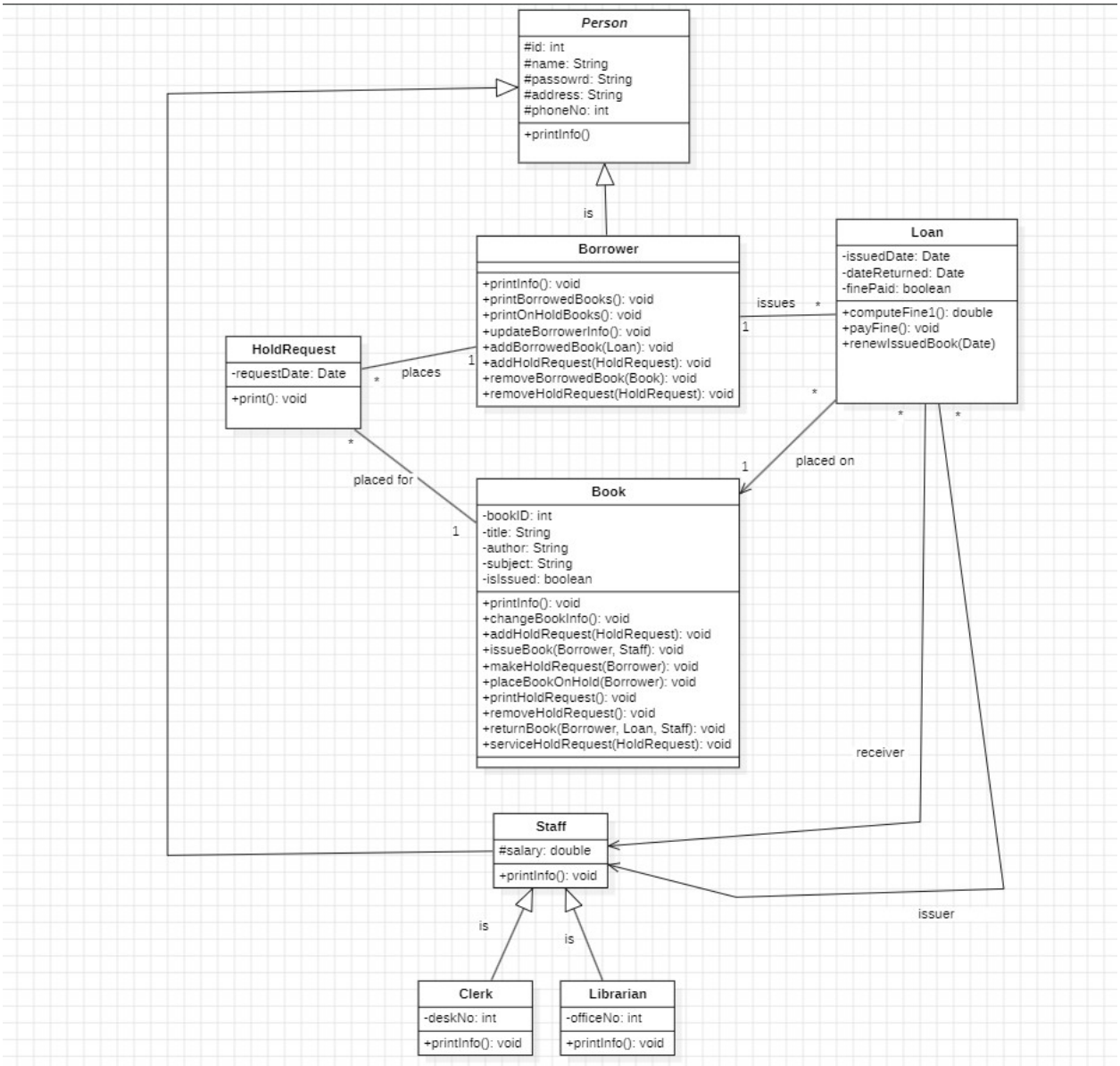
8. Preliminary Schedule and Budget

The project is estimated to be
4 weeks and is estimated to cost
$ 20,000

Classes

* User
* Librarian
* Login page
* Article catalogue
* Article database
* Payment interface
* Bank database.

Class Diagram



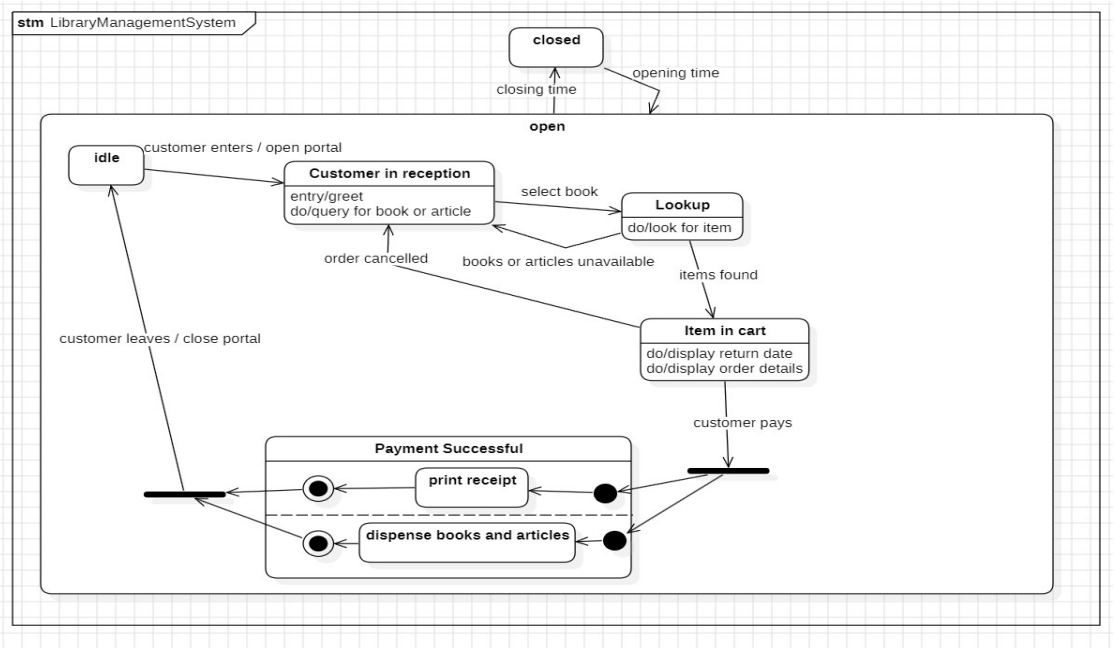
The class diagram represents a Library Management System, illustrating the relationships and interactions between various entities essential for library operations. Central to the system is the Person class, encompassing attributes like ID, name, password, address, and phone number, with a method to print personal information. This class acts as a general entity from which the Borrower class is derived. The Borrower class, inheriting from Person, includes additional methods for managing borrowed books and hold requests, such as updating borrower information, adding and removing borrowed books, and managing hold requests.

The HoldRequest class, linked to Borrower, includes attributes for the request date and a method to print the request details. The Book class comprises attributes like book ID, title, author, subject, and issued status. It provides various methods for printing and updating book information, managing hold requests, issuing and returning books, and servicing hold requests. The Loan class, associated with both Book and Borrower classes, includes attributes for issued date, return date, and fine paid, along with methods to compute and pay fines and renew issued books.

On the administrative side, the Staff class includes an attribute for salary and a method to print staff information, serving as a superclass for the Clerk and Librarian classes. The Clerk class adds an attribute for desk number, while the Librarian class adds an attribute for office number, both inheriting from Staff.

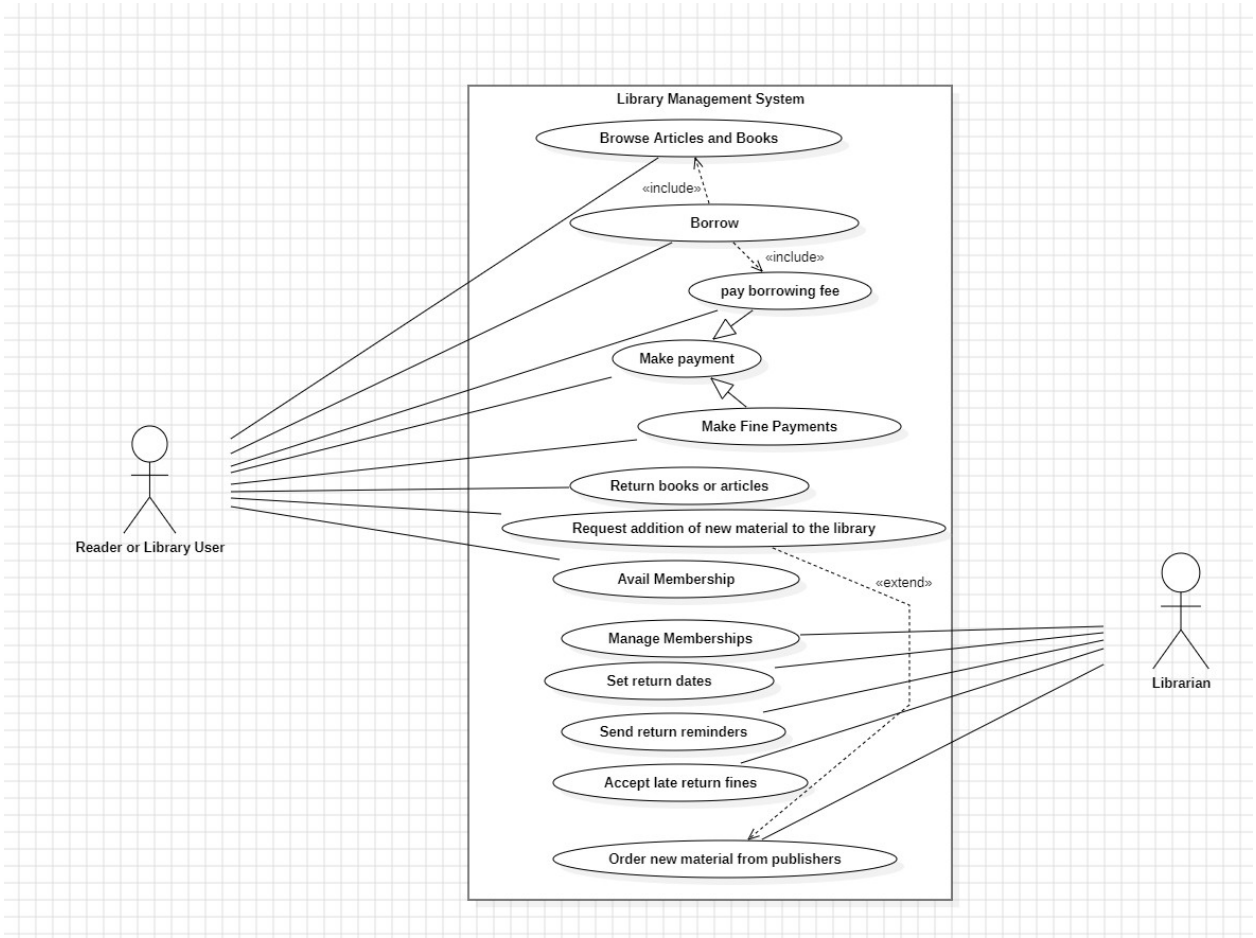
The diagram visually represents the inheritance hierarchy (e.g., Person to Borrower, Staff to Clerk and Librarian), associations (e.g., Borrower to HoldRequest, Book to HoldRequest and Loan), and multiplicity (e.g., a borrower can have multiple hold requests, and a book can be placed on hold by multiple borrowers). This class diagram provides a clear blueprint for implementing the Library Management System, illustrating how various components interact and contribute to the overall functionality and efficiency of the system.

State Diagram



The state diagram for the Library Management System illustrates the various states a customer interaction can transition through during their experience. Initially, the system is in the "closed" state, representing the library being closed. When the opening time is reached, the system transitions to the "open" state, starting in the "idle" state, where it waits for customer interactions. Upon a customer entering the library and engaging with the portal, the system moves to the "Customer in reception" state, where it welcomes the customer and prompts for their query, such as searching for a book or article. If the customer selects a book, the system transitions to the "Lookup" state to search for the item in the database. If the item is found, the state changes to "Item in cart," displaying the return date and order details. If the item is unavailable, the system returns to "Customer in reception." Should the customer decide to cancel the order, the system also reverts to "Customer in reception." If the customer proceeds with the order, the system enters the "Payment Successful" state upon successful payment, generates a receipt, and dispenses the book or article. Afterward, the system returns to the "idle" state, ready for the next interaction. Finally, at closing time, the system transitions back to the "closed" state. This diagram effectively outlines the flow of customer interactions and the corresponding states, ensuring a seamless and efficient library experience.

Use Case Diagram



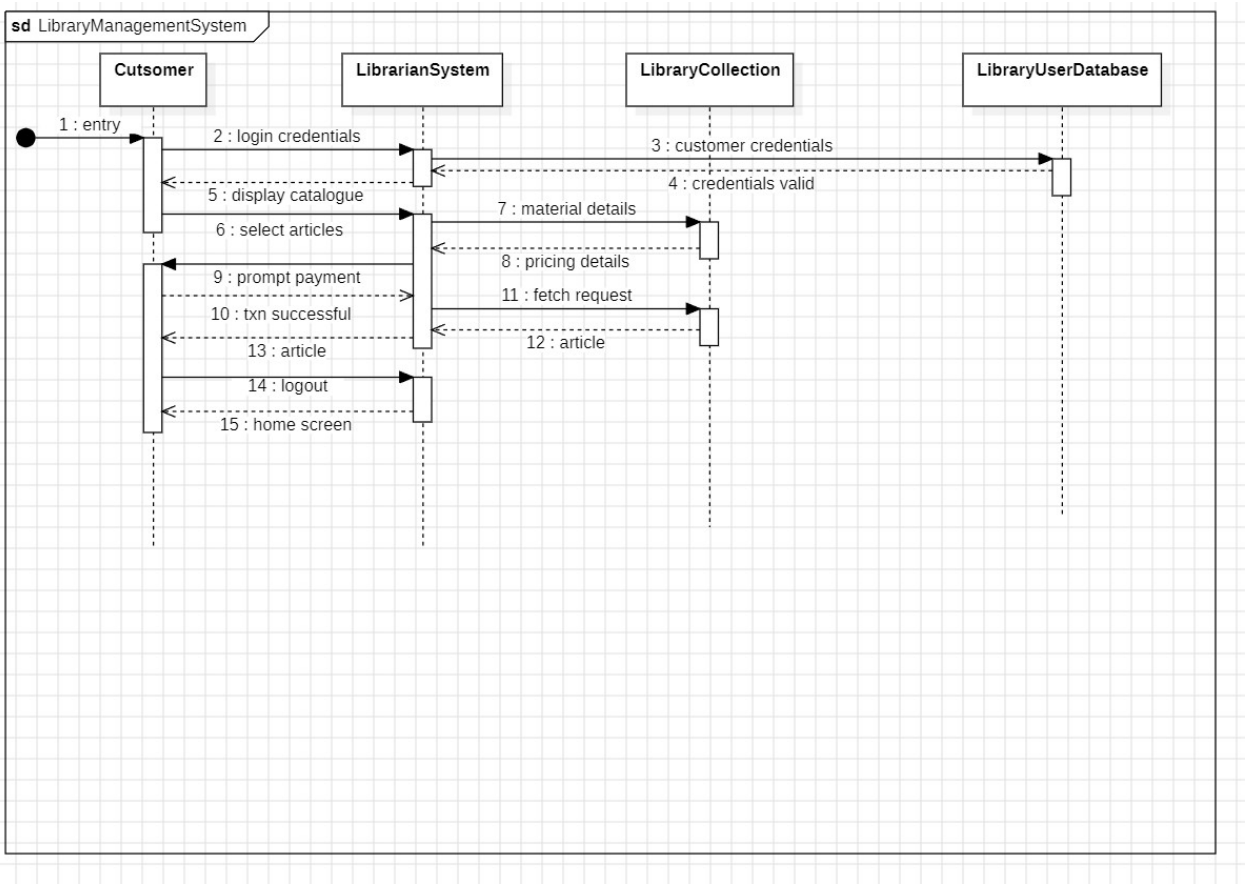
The use case diagram for the Library Management System showcases the various interactions between the primary actors, namely the "Reader or Library User" and the "Librarian," with the library system. For the "Reader or Library User," the system enables several key functionalities: browsing articles and books, borrowing items which entails paying a borrowing fee, making payments including fines, returning borrowed books or articles, requesting the addition of new material to the library, and availing membership. Each of these use cases represents a specific service that the user can access within the system.

On the other hand, the "Librarian" has a distinct set of responsibilities which include managing memberships, setting return dates, sending return reminders, accepting late return fines, and ordering new materials from publishers. These actions ensure the smooth operation of the library and support user interactions.

The diagram also employs "include" and "extend" relationships to indicate dependencies and optional functionalities within the system. For instance, the "Borrow" use case includes the "Pay Borrowing Fee" use case, signifying that payment is a necessary step in the borrowing process. Additionally, the "Avail Membership" use case extends to "Manage Memberships," highlighting the extended functionalities available to users with memberships.

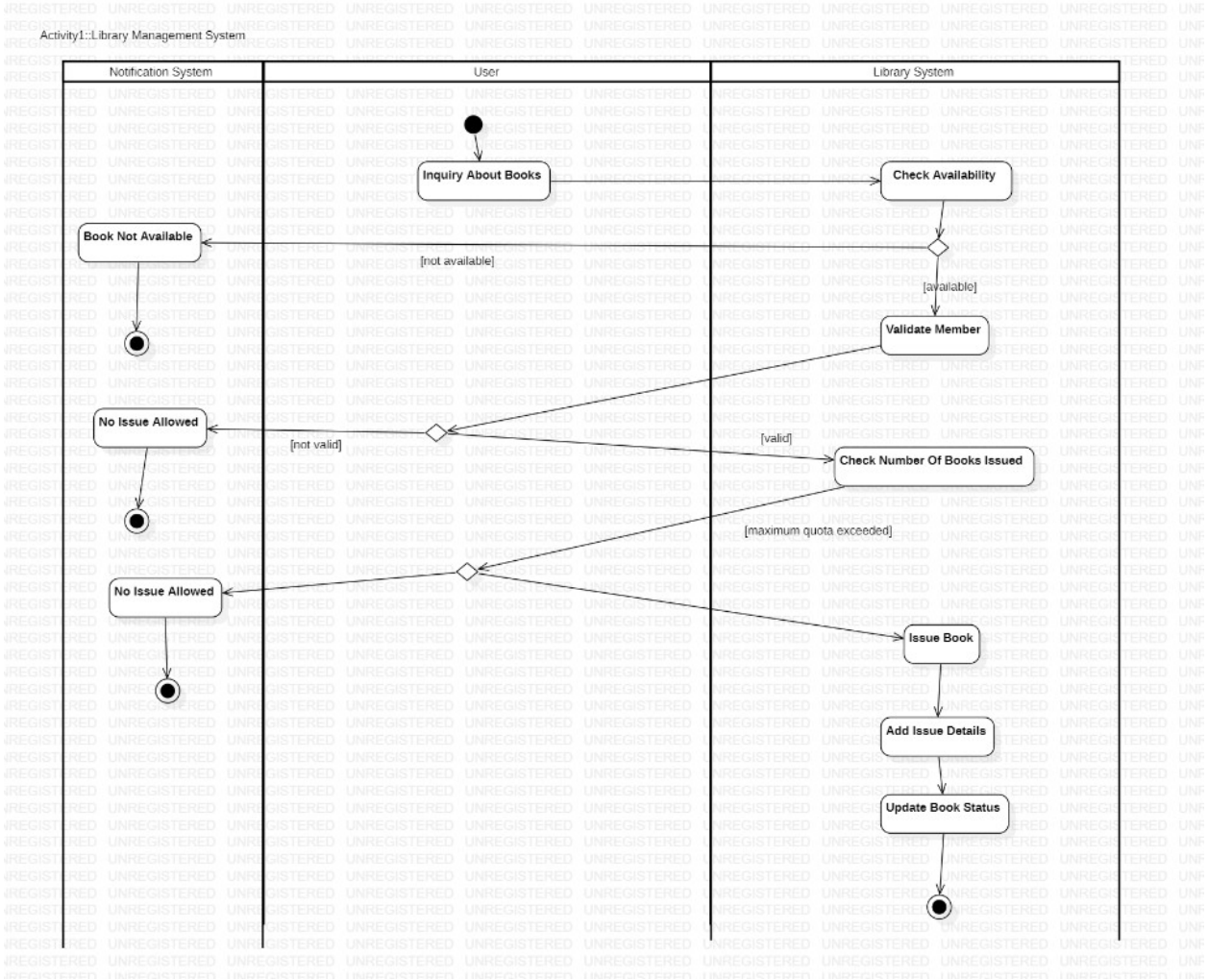
This diagram provides a comprehensive visual representation of the various functionalities of a library management system and the roles of different users. It serves as a valuable tool for understanding the system's requirements, aiding in the design and implementation of an effective and user-friendly library management system

Sequence Diagram



This sequence diagram represents the interactions within a Library Management System involving four main entities: Customer, LibrarianSystem, LibraryCollection, and LibraryUserDatabase. The diagram outlines the process from the customer's entry to the system, beginning with the customer sending their login credentials to the LibrarianSystem. The system then forwards these credentials to the LibraryUserDatabase for validation. Once the credentials are validated, the system displays the catalogue to the customer. The customer selects the desired articles, prompting the LibrarianSystem to request material details from the LibraryCollection. The LibraryCollection provides the pricing details, after which the system prompts the customer for payment. Upon successful payment, the LibrarianSystem sends a fetch request to the LibraryCollection for the selected articles, which are then delivered to the customer. Finally, the customer logs out, and the system returns to the home screen. This sequence diagram effectively illustrates the step-by-step interactions and processes involved in accessing and obtaining library resources, emphasizing the roles of each entity in ensuring a smooth and efficient transaction.

Activity Diagram



The activity diagram for the Library Management System presents a detailed workflow for a user inquiring about books and the subsequent actions taken by the system. The process starts with the user making an inquiry about the availability of books. The system then checks the availability of the requested book. If the book is not available, the notification system informs the user that the book is not available. If the book is available, the system proceeds to validate the user's membership. If the membership is not valid, the notification system alerts the user that no issue is allowed. For valid members, the system checks the number of books the member has already issued to ensure they have not exceeded their quota. If the quota is exceeded, the notification system informs the user that no issue is allowed. If the user is within their limit, the system issues the book to the user, adds the issue details to the system, and updates the book's status to reflect it has been issued. This diagram effectively illustrates the comprehensive process

involved in checking the availability and issuing of books, emphasizing the different validation steps and conditions that ensure efficient library management operations.

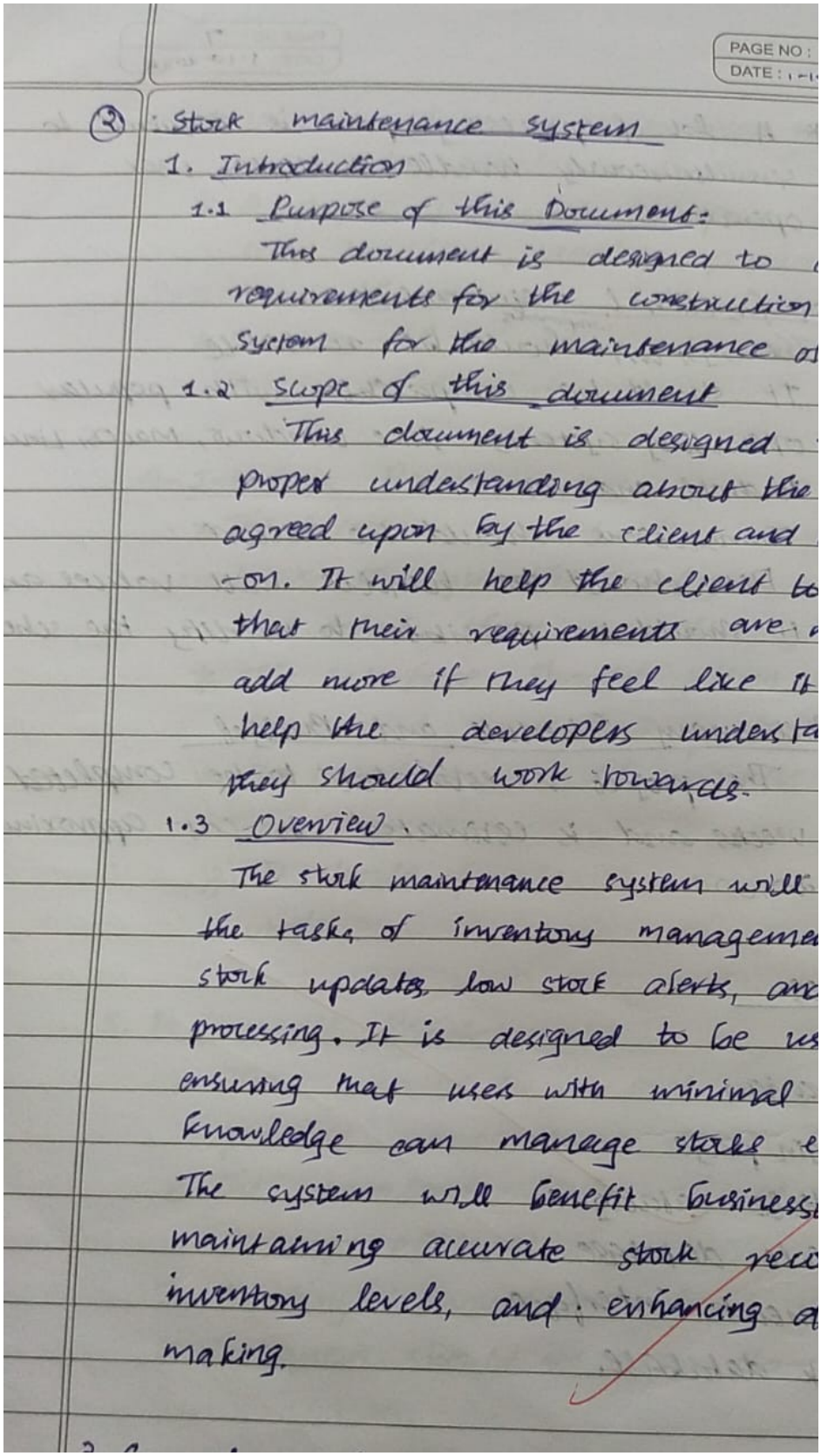
4. Stock Maintenance System

Problem Statement

Efficiently managing warehouse stock is critical for ensuring the smooth operation of supply chains and maintaining optimal inventory levels. The objective of developing a Stock Maintenance System is to streamline the processes involved in tracking, managing, and replenishing warehouse stock, thus enhancing inventory accuracy, reducing stockouts, and minimizing excess inventory. This system must provide functionalities such as real-time inventory tracking, automated reorder alerts, and comprehensive reporting capabilities.

It should allow for the easy addition and removal of stock, accurate recording of stock levels, and tracking of stock movement within the warehouse. The system should also facilitate the management of multiple warehouse locations, support barcode scanning for quick and accurate stock entry, and integrate with other enterprise systems such as procurement and sales. By implementing a robust Stock Maintenance System, businesses can improve operational efficiency, ensure timely fulfillment of orders, and make informed decisions based on accurate inventory data, ultimately leading to increased customer satisfaction and reduced operational costs.

System Requirements Specification



PAGE NO :
DATE :

(2) Stock maintenance system

1. Introduction

1.1 Purpose of this Document:

This document is designed to
requirements for the construction
system for the maintenance of

1.2 Scope of this document

This document is designed
proper understanding about the
agreed upon by the client and
-on. It will help the client to
that their requirements are
add more if they feel like it
help the developers understa
they should work towards.

1.3 Overview

The stock maintenance system will
the tasks of inventory managemen
stock updates, low stock alerts, and
processing. It is designed to be us
ensuring that users with minimal
knowledge can manage stocks e
The system will benefit business
maintaining accurate stock reco
inventory levels, and enhancing d
making.

levels and track stock movement
features include automated stock r
notifications and support for mu
warehouse locations. This system wi
companies that rely on accurate sto
for daily operations.

3. Functional Requirements

* Users can add new stock items upda
stock and remove obsolete items.
* Stock level alerts: The system wi
-y notify the user when stock le
below the defined threshold.
* The system will store supplie
link them to corresponding stock
easy re-ordering.
* Users can generate stock repors,
stock history, stock levels and v
reports.

4. Interface Requirements

*User Interface: The system will ha
graphical user interface for easy
it will support multiple devices.
* Data interfaces: The system will co
with external systems such as a

* Uptime: The system will maintain uptime to support critical business

6. Design Constraints

* The system must use a relational for storing stock data (eg. MySQL or P
* The system should be compatable barcode readers for faster input.
* The system must be accessible on desktop and mobile devices

7. Non-functional Attributes

* The system will use a role-based control and secure login.
* The system must ensure that s accurate and consistently avail business operations.
* The system will be designed to s the increasing stock items and user

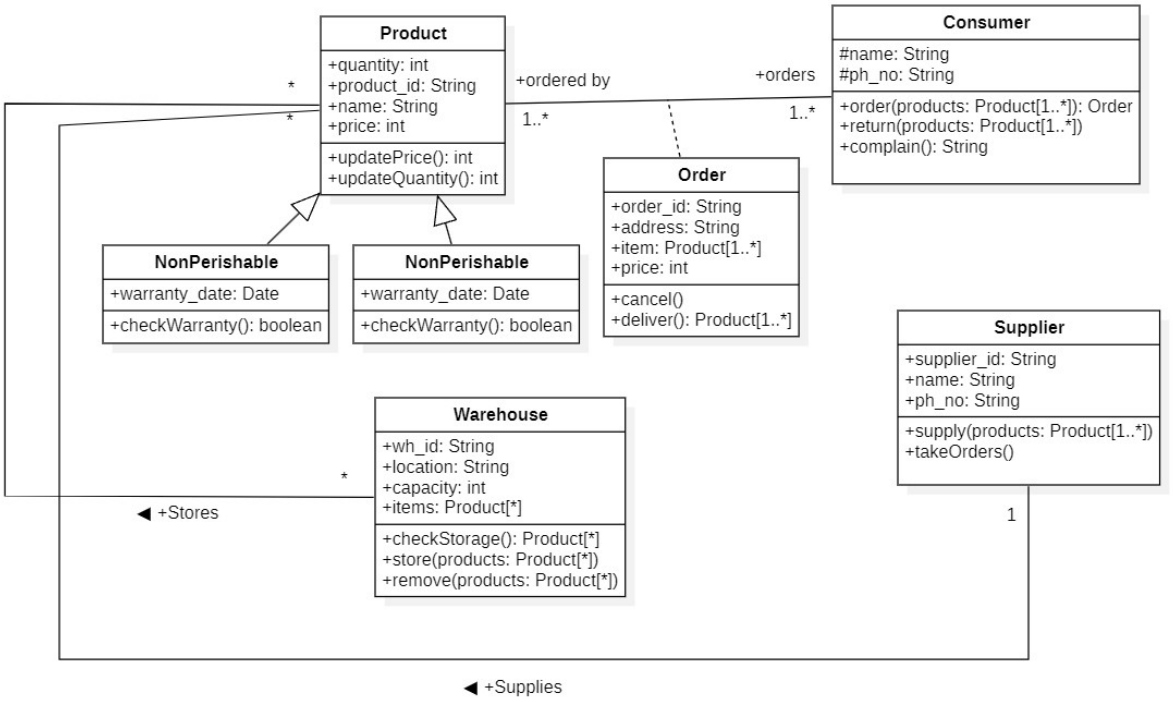
8. Preliminary Schedule and Budget

* Development duration: The system to take 6 months to develop.
* Budget: The estimated budget fo is ₹ 10,00,000.

SRS → ₹ 30,000

Development → ₹ 4,70,000

Class Diagram



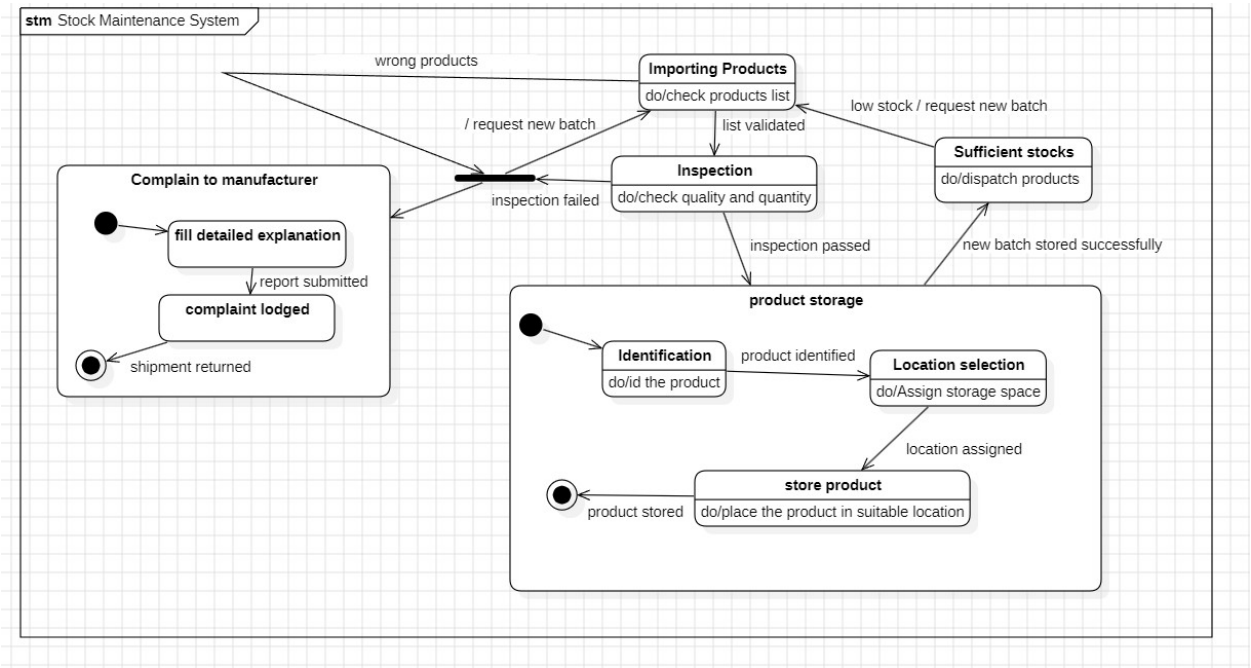
This class diagram provides a detailed representation of a product ordering and warehouse management system, illustrating the relationships and attributes of various entities involved. At the core is the Product class, which includes attributes such as quantity, product ID, name, and price. The class has methods for updating the price and quantity of the product. The NonPerishable class inherits from Product and adds a warranty date attribute along with a method to check the warranty status, indicating that it handles non-perishable items with warranty considerations.

The Order class includes attributes like order ID, address, item list, and price, with methods to cancel and deliver orders. It has a relationship with the Consumer class, which contains attributes such as name and phone number, and methods for ordering, returning products, and lodging complaints. This association signifies that a consumer can place multiple orders, and each order can contain one or more products.

The Supplier class, which features attributes like supplier ID, name, and phone number, and methods for supplying products and taking orders, interacts with the Warehouse class. The Warehouse class, characterized by its warehouse ID, location, capacity, and item list, includes methods to check storage, store products, and remove products. It indicates that a supplier provides products to warehouses, and a warehouse stores multiple products.

Overall, this class diagram effectively illustrates the structure and interrelationships within the product ordering and warehouse management system. It highlights the flow of products from suppliers to warehouses, the management of product inventory, and the interactions between consumers, orders, and products, ensuring efficient and organized operations.

State Diagram



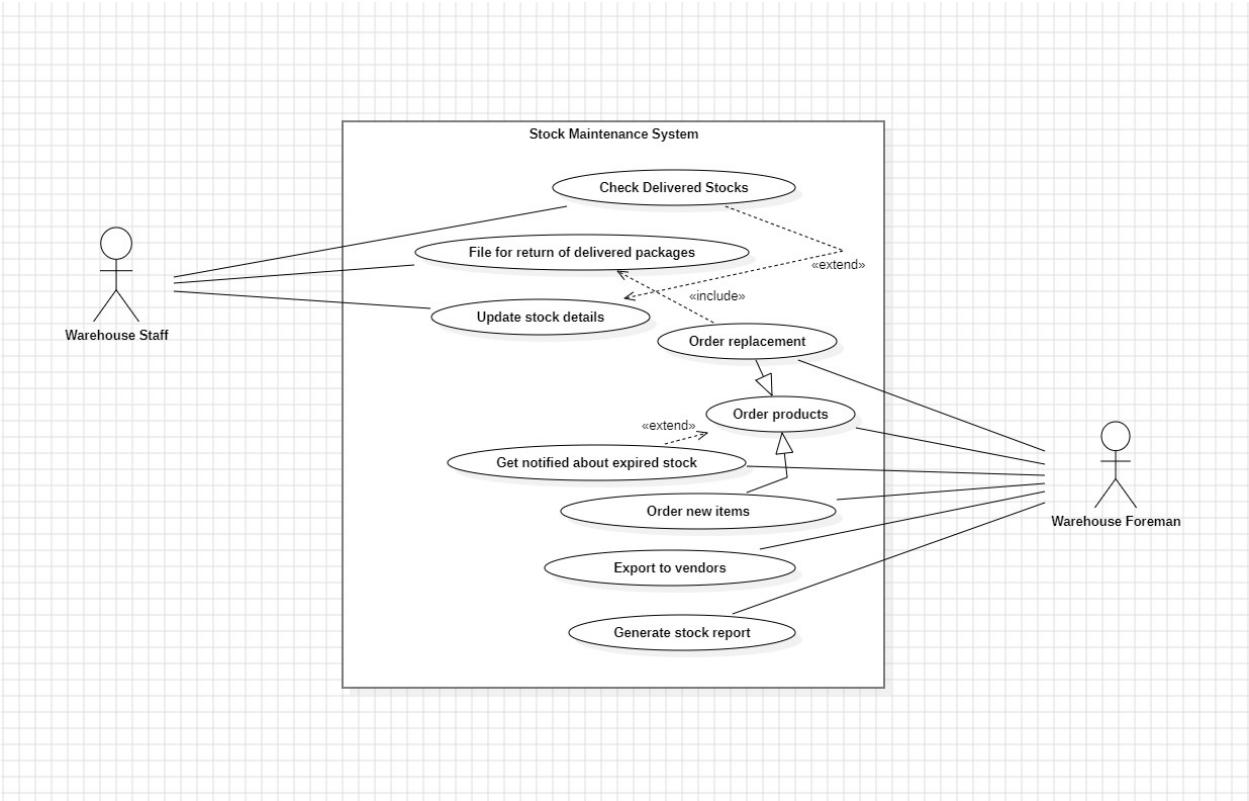
The state diagram for the Stock Maintenance System (STM) provides a detailed visual representation of the various states and transitions involved in maintaining warehouse stock, from importing products to storing them and handling complaints. The process begins with the system in the "Importing Products" state, where the system checks the product list for validation. If the list is validated, the system transitions to the "Inspection" state, where the quality and quantity of the products are checked. If the inspection passes, the system moves to the "Product Storage" state. Within the "Product Storage" state, there are sub-states including "Identification,"

"Location Selection," and "Store Product." During the identification sub-state, the system identifies the product. It then transitions to the location selection sub-state where storage space is assigned, and finally to the store product sub-state where the product is placed in the appropriate location.

If the stock is sufficient, the system moves to the "Sufficient Stocks" state, where products can be dispatched. If there is low stock or a new batch is requested, the system transitions back to the "Importing Products" state. In case of a failed inspection, the system moves to the "Complain to Manufacturer" state, where a detailed explanation is filled, and a complaint is lodged. The shipment is then returned.

This state diagram effectively illustrates the sequential steps and transitions involved in maintaining stock, emphasizing the importance of validation, inspection, storage, and complaint handling to ensure efficient warehouse operations. The detailed flow provides a clear understanding of the actions required at each stage and the conditions that trigger transitions between states.

Use Case Diagram

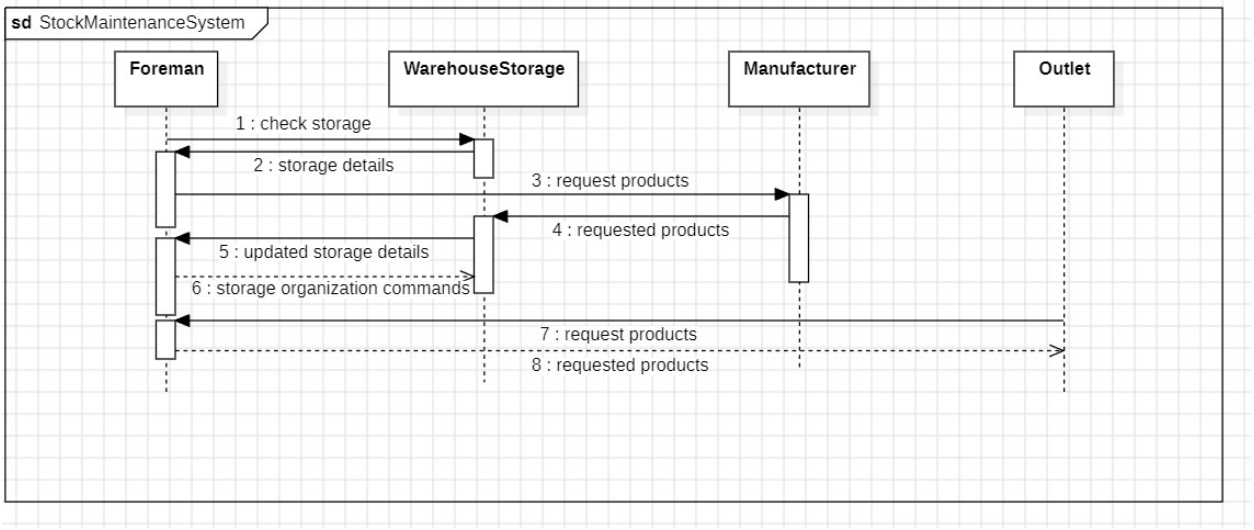


This use case diagram illustrates the interactions between two main actors—Warehouse Staff and Warehouse Foreman—and a Stock Maintenance System. The Warehouse Staff has four primary use cases: checking delivered stocks, filing for returns of delivered packages, updating stock details, and getting notified about expired stock. The "File for return of delivered packages" use case is an extension of "Check Delivered Stocks," highlighting that returns are processed after checking the delivery.

The Warehouse Foreman is involved in ordering products, ordering new items, exporting to vendors, and generating stock reports. The "Order replacement" use case includes "Order products," signifying that ordering replacements is part of the broader product ordering process. Additionally, "Get notified about expired stock" extends "Order products," indicating that notifications about expired stock can trigger new product orders.

This diagram effectively captures the functional requirements of the Stock Maintenance System, showing the different roles and their specific tasks, as well as the relationships between the various processes. It provides a clear overview of the system's capabilities and how it supports the efficient management of warehouse stock.

Sequence Diagram

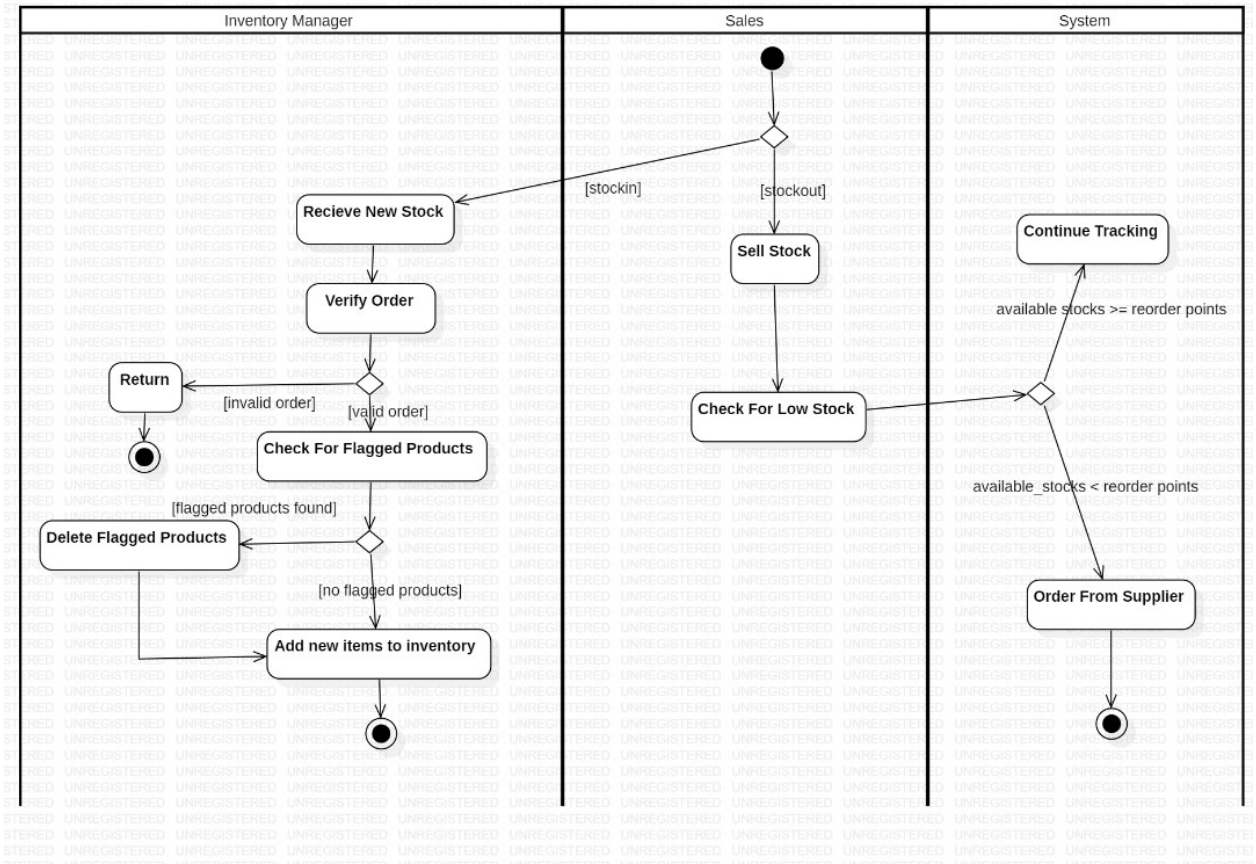


The sequence diagram for the Stock Maintenance System illustrates the interaction between the Foreman, WarehouseStorage, Manufacturer, and Outlet. The process begins with the Foreman checking the storage by sending a message to WarehouseStorage. WarehouseStorage responds with the storage details, allowing the Foreman to request the required products. Once the

products are identified, WarehouseStorage sends the requested items to the Foreman, who then updates the storage details and informs WarehouseStorage. Following this, the Foreman issues storage organization commands to ensure proper arrangement and management of the inventory.

In parallel, WarehouseStorage communicates with the Manufacturer to request new products when stock levels are low or to replenish the inventory. The Manufacturer processes the request and sends the necessary products to WarehouseStorage, which updates its records accordingly. Additionally, WarehouseStorage handles the dispatch of products to the Outlet, ensuring that items are available for customer sales. This sequence diagram effectively highlights the flow of information and products, emphasizing the coordination required between different entities to maintain optimal inventory levels and ensure efficient stock management. It demonstrates the critical role of each actor in facilitating smooth operations within the warehouse and beyond.

Activity Diagram



This activity diagram provides a clear visual representation of the inventory management workflow within a warehouse setting, highlighting the interactions among the Inventory Manager, Sales, and System components. The process initiates with the Inventory Manager receiving new stock, followed by verifying the order. If the order is deemed invalid, it is returned to the supplier. Conversely, if the order is valid, the system checks for any flagged products. Flagged products are removed from the inventory, while unflagged products are added to the inventory database. Simultaneously, the Sales component handles stock sales, and the system monitors stock levels. Upon detecting low stock levels, the system decides whether to continue tracking inventory or place a new order from the supplier, based on the current stock levels compared to the predetermined reorder points. This diagram effectively captures the decision points and actions involved in managing warehouse stock, ensuring efficient inventory control and seamless operations.

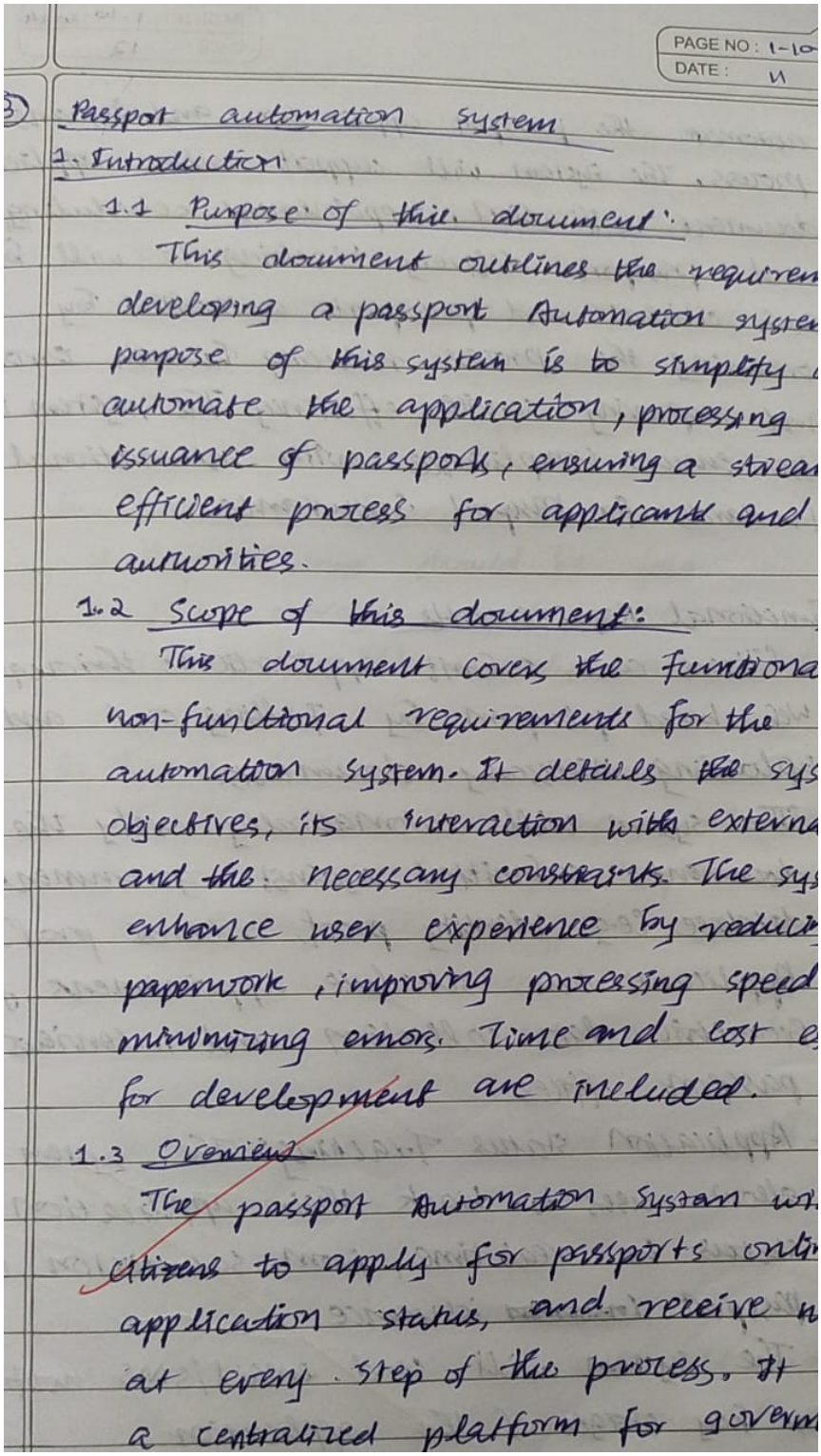
5. Passport Automation System

Problem Statement

In an increasingly globalized world, the need for an efficient and reliable Passport Automation System is paramount to streamline the processes of passport application, renewal, and management. The primary objective of developing this system is to simplify and automate the workflow associated with passport services, reducing manual intervention and minimizing errors. The system must support functionalities such as online application submission, document verification, appointment scheduling, and status tracking. It should also facilitate seamless communication between applicants and the passport office, providing real-time updates and notifications regarding application status.

Additionally, the system needs to incorporate robust security measures to safeguard sensitive personal information, ensuring compliance with data protection regulations. Automated workflows for document validation, fee payment, and biometric data capture are essential to enhance efficiency and reduce processing times. By implementing a comprehensive Passport Automation System, government agencies can improve service delivery, reduce administrative burdens, and enhance the overall experience for citizens seeking passport services, ultimately contributing to a more streamlined and user-friendly process.

System Requirements Specifications



PAGE NO : 1-10
DATE : 11

3) Passport automation system

1. Introduction

1.1 Purpose of this document:

This document outlines the requiren
developing a passport Automation syste
purpose of this system is to simplify
automate the application, processing
issuance of passports, ensuring a strea
efficient process for applicants and
authorities.

1.2 Scope of this document:

This document covers the functiona
non-functional requirements for the
automation system. It details the sys
objectives, its interaction with externa
and the necessary constraints. The sys
enhance user experience by reducin
paperwork, improving processing speed
minimizing errors. Time and cost e
for development are included.

1.3 Overview

The passport Automation System wi
citizens to apply for passports onli
application status, and receive n
at every step of the process. It
a centralized platform for goverm

automate the passport application and process. The system will support online document verification, appointment sch and passport delivery tracking. It both citizens and passport authorities streaming the process, reducing human and improving service efficiency. The s also ensure compliance with interna standards for passport insurance.

3. Functional Requirements:

* Citizens can submit applications t web based platform by filling ou uploading necessary documents.
* The system will automatically veri documents submitted against gove database (e.g. identity proof, address
* Applicants can schedule appointm biometric data collection and ins passport offices.
* Application status Tracking: The allow users to track their applic status in real time, from submis passport issuance.
* The system will send email/SMS at key stages of the application.
* Authorities can generate and issue

devices, enabling applicants to navigat
and complete their tasks.
* Data interfaces: The system will in
government databases for automatic
-tion validation. It will also integrat
postal services for tracking passport
* The system will support biometric

5. Performance Requirements
* Response time should be fast.
* Data storage should be secure
* Uptime should be nearly 100%.

6. Design Constraints
* The system must comply with int
security standards for Passport iss
including data encryption and secu
* The system will use biometric data
devices that are compatible with
government regulations
* The system should integrate with
identification databases.

7. Non-functional attributes
* Security: The system will used ad
encryption techniques to protect p
biometric data.

PAGE NO :
DATE :

8. Preliminary Schedule and Budget

* Development Duration: The system
to take 9 months to develop, incl
testing and deployment.

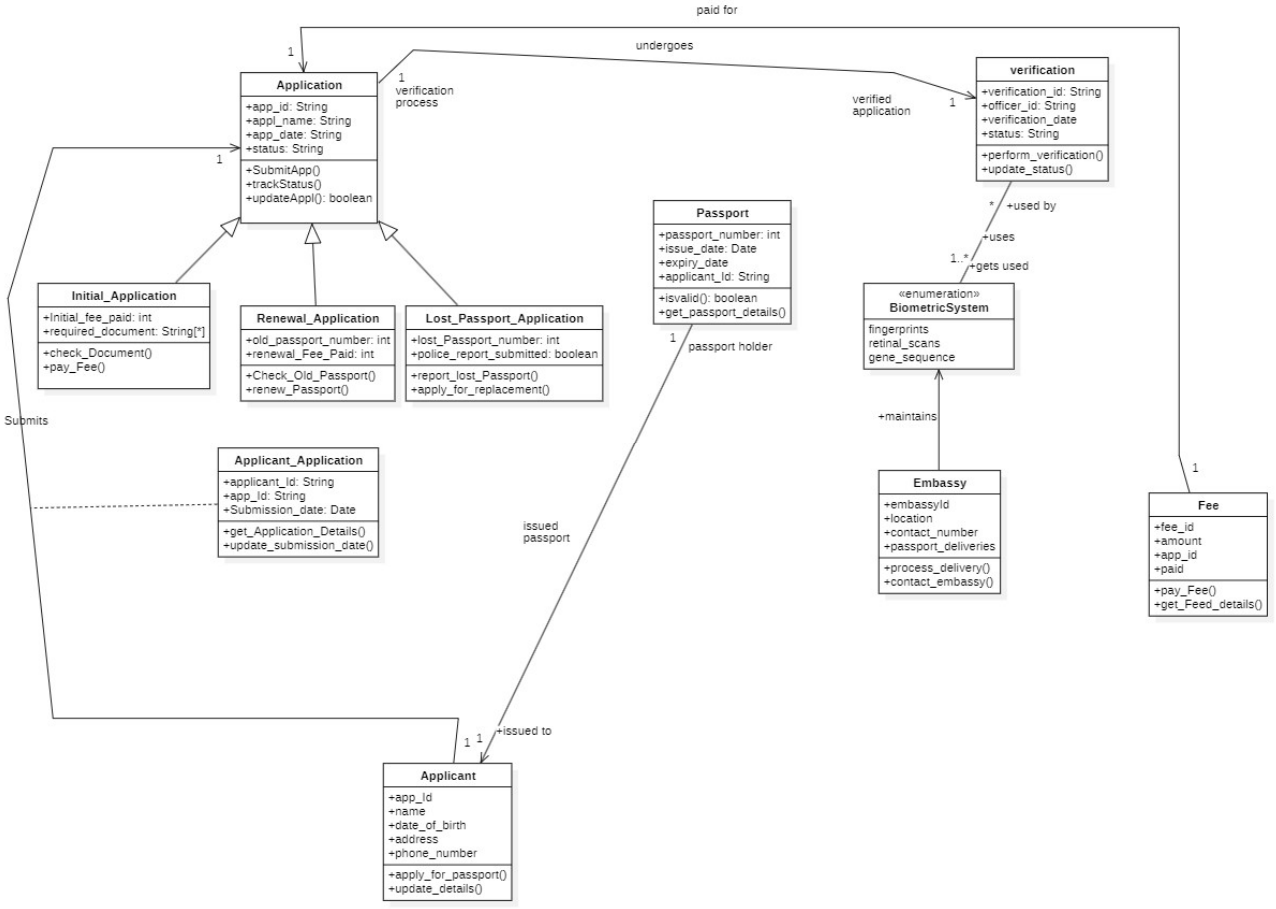
* Budget: ₹ 5,00,000

SRS : ₹ 50,000

Development: ₹ 2,00,000

* Testing Phase : ₹ 1,00,000

Class Diagram



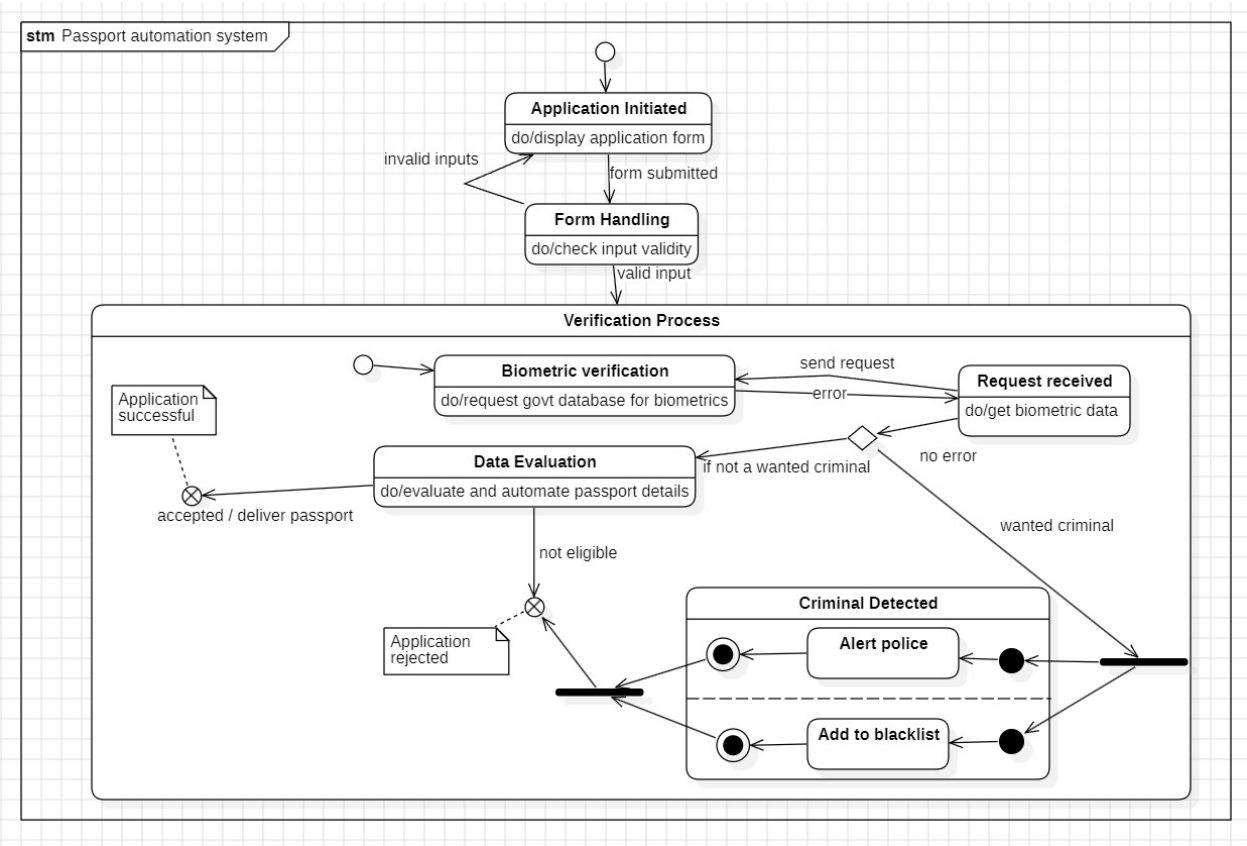
The class diagram represents a comprehensive Passport Application System, highlighting the various classes involved and their interrelationships. The central class is the Application class, which contains attributes such as app_id, appl_name, app_date, and status, with methods to submit an application, track its status, and update it. This class branches into three specific types of applications: Initial_Application, Renewal_Application, and Lost_Passport_Application, each with unique attributes and methods pertinent to their application type. For instance, Initial_Application includes attributes like initial_fee_paid and required_document, with methods to check documents and pay fees. Renewal_Application covers attributes such as old_passport_number and renewal_fee_paid, with methods for checking the old passport and renewing it. Lost_Passport_Application handles attributes like lost_passport_number and police_report_submitted, with methods to report a lost passport and apply for a replacement.

The Applicant_Application class links applicants to their applications, containing attributes like applicant_Id, app_Id, and submission_date, and methods for retrieving application details and updating submission dates. The Applicant class includes personal information attributes and methods for applying for a passport and updating details. The Passport class includes attributes like passport_number, issue_date, expiry_date, and applicant_Id, with methods for checking validity and retrieving passport details.

Additionally, the Verification class manages the verification process with attributes like verification_id, officer_id, verification_date, and status, and methods to perform and update verification. The BiometricSystem enumeration includes biometric types such as fingerprints, retinal scans, and gene sequences. The Embassy class deals with passport delivery, including attributes like embassyId, location, and contact_number, and methods for processing delivery and contacting the embassy. Lastly, the Fee class manages the financial aspects, with attributes such as fee_id, amount, app_id, and paid, and methods for paying fees and retrieving fee details.

This diagram provides a detailed overview of the system's structure, illustrating how applications are processed, verified, and managed, ensuring a smooth and efficient workflow for passport issuance.

State Diagram



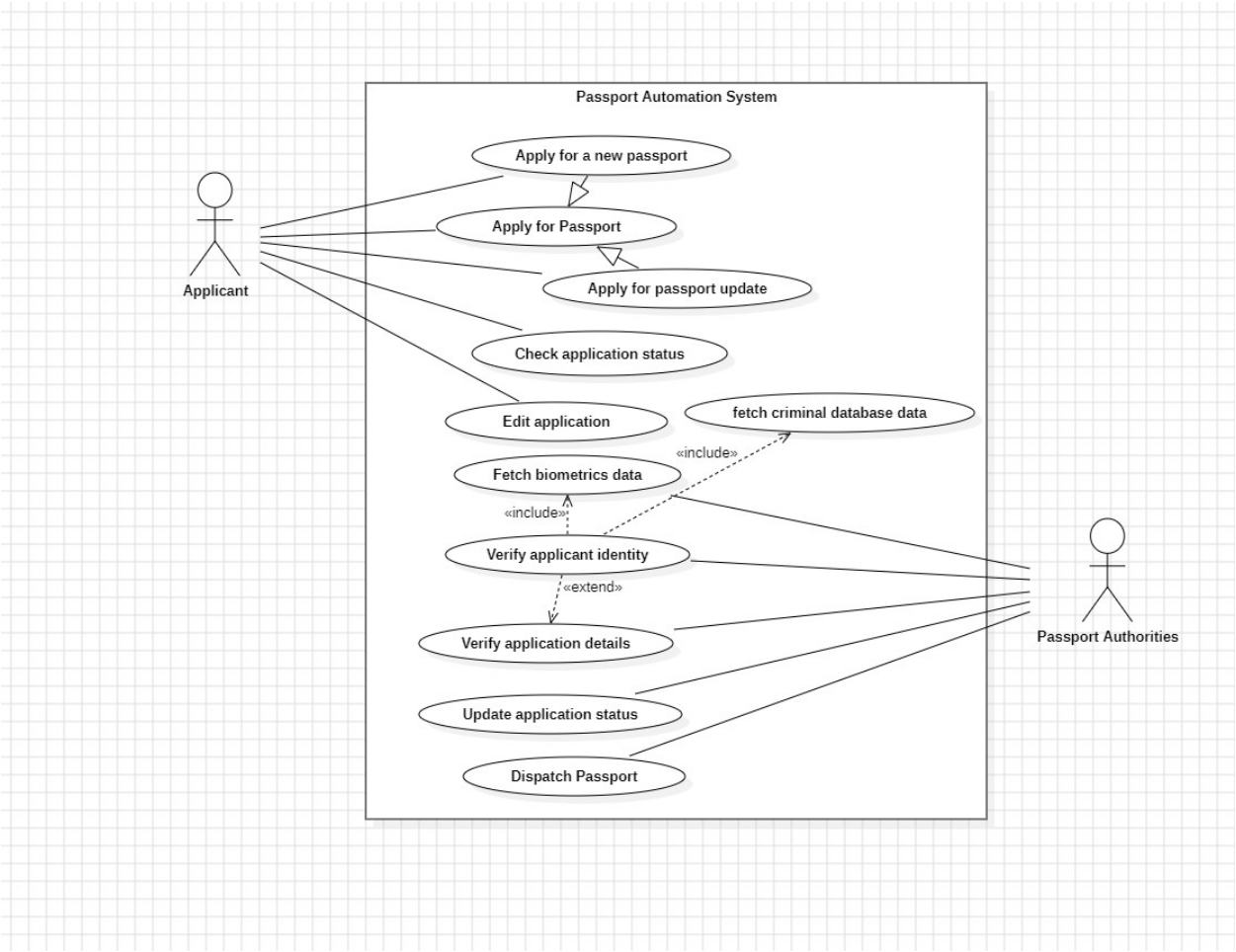
The state diagram illustrates the process flow of a passport automation system, detailing the various states and transitions involved from the application initiation to the final decision. The process begins with the initiation of the application, where the system displays the application form to the user. If the inputs are invalid, the system loops back to the form handling stage, requiring the user to correct the errors. Once the form is submitted with valid inputs, the system proceeds to the verification process, which includes biometric verification.

During biometric verification, the system requests biometric data from a government database. If an error occurs during this stage, the process loops back to the biometric verification stage to rectify the issue. Upon successful retrieval and verification of the biometric data, the system evaluates the data to automate the passport details. If the applicant is found to be eligible and is not a wanted criminal, the passport application is accepted, and the passport is delivered to the applicant, marking the application as successful.

However, if the applicant is a wanted criminal, the system detects this and triggers two actions: alerting the police and adding the applicant to a blacklist. In cases where the applicant is not

eligible for other reasons, the application is rejected. This state diagram effectively highlights the importance of biometric verification and data evaluation in ensuring the security and eligibility of passport applications, providing a clear and structured overview of the steps involved in automating the passport application process.

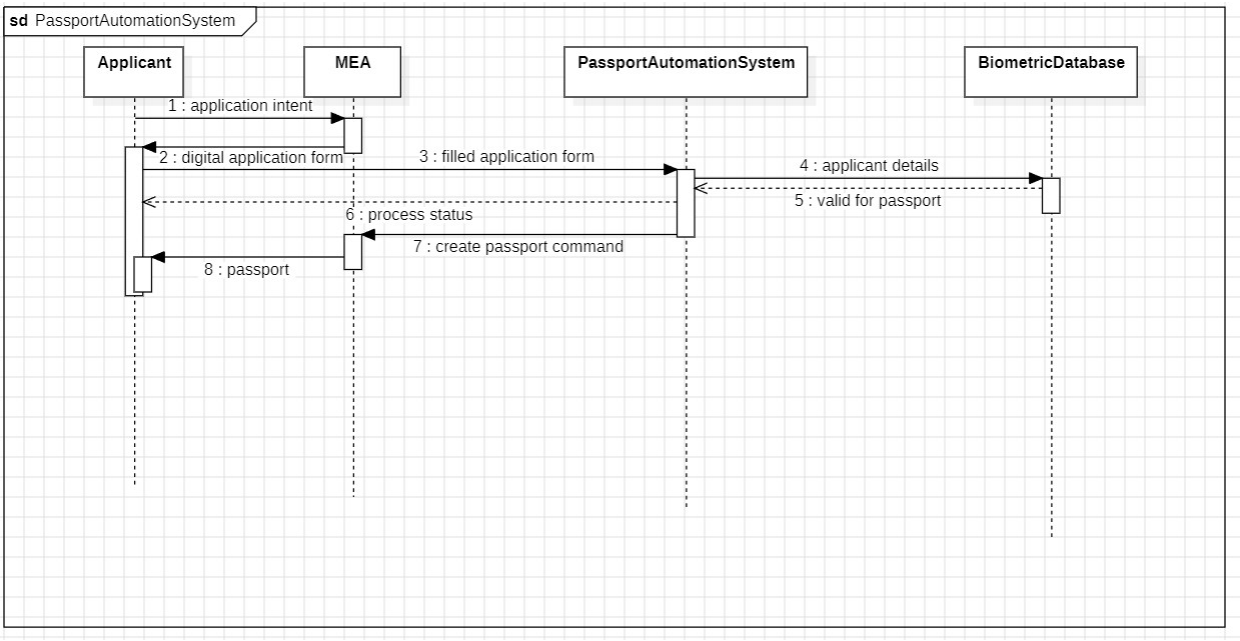
Use Case Diagram



This use case diagram showcases the interactions between the primary actors—Applicant and Passport Authorities—and the Passport Automation System. For Applicants, the system supports actions like applying for a new passport, applying for a passport update, checking application status, and editing applications. Passport Authorities interact with the system to perform tasks such as fetching criminal database data, fetching biometric data, verifying applicant identity, verifying application details, updating application status, and dispatching passports. The diagram clearly defines the relationships between use cases, with "Fetch biometrics data" including

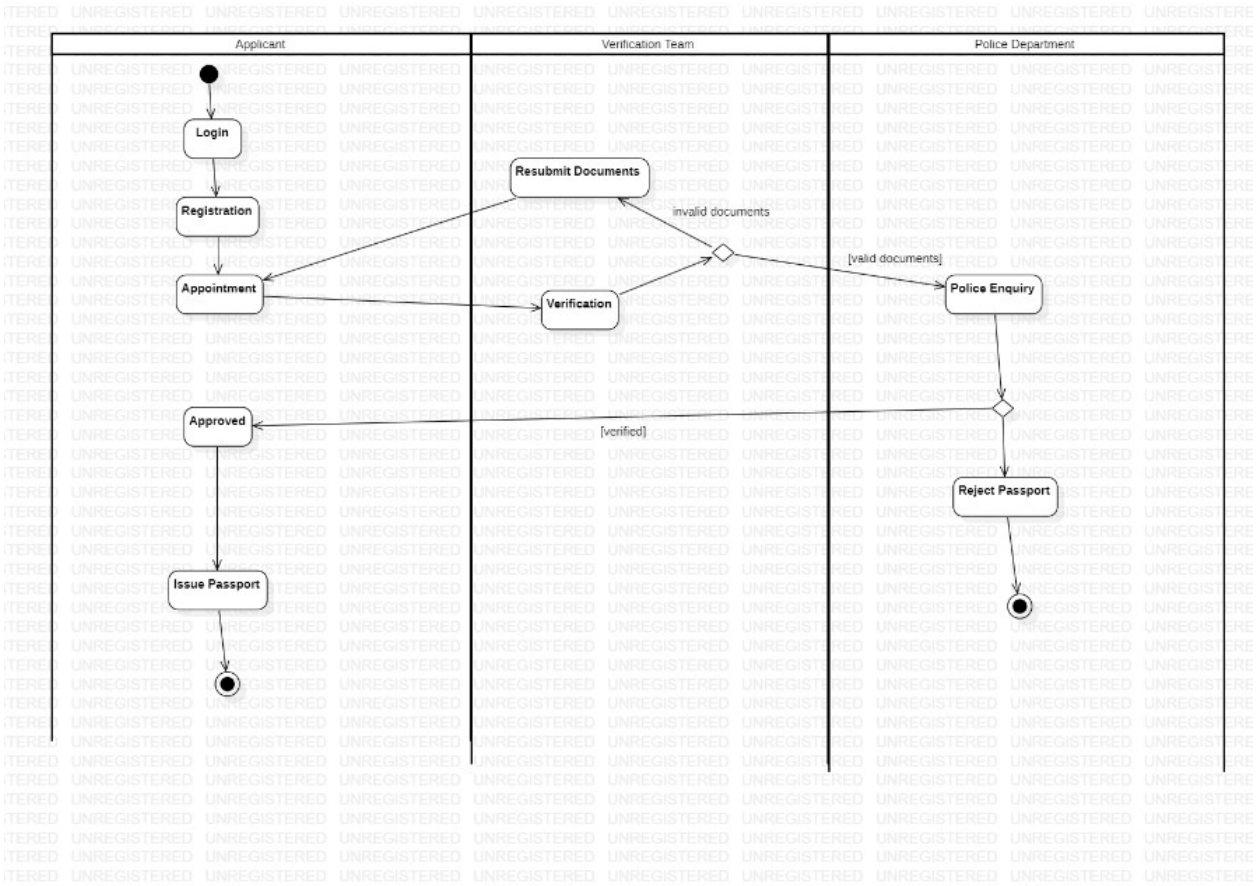
"Verify applicant identity" and "Verify applicant identity" extending "Verify application details." This illustration provides a comprehensive overview of the system's functionality, highlighting the roles and responsibilities of both applicants and authorities, and ensuring a streamlined and efficient process for managing passport applications.

Sequence Diagram



The sequence diagram illustrates the process flow of a Passport Automation System involving four entities: Applicant, MEA (Ministry of External Affairs), PassportAutomationSystem, and BiometricDatabase. The process begins with the Applicant expressing an application intent to the MEA. The MEA then sends a digital application form back to the Applicant. The Applicant fills out the application form and sends it back to the MEA. The MEA forwards the applicant details to the PassportAutomationSystem, which then verifies the details with the BiometricDatabase. Upon validation, the PassportAutomationSystem sends a process status back to the MEA. The MEA then issues a command to create the passport, and finally, the passport is delivered to the Applicant. This diagram provides a clear and structured visualization of the interactions and data flow between different entities in the passport application process, ensuring efficient and accurate handling of applications.

Activity Diagram



This activity diagram outlines the structured process of applying for and issuing a passport, divided into three distinct swimlanes: Applicant, Verification Team, and Police Department. The process initiates with the applicant logging into the system and completing the registration. Following successful registration, the applicant schedules an appointment. If the appointment is successful, the system advances the application to the approval stage, leading to the issuance of the passport. Concurrently, the Verification Team is responsible for receiving the applicant's documents. If any document is deemed invalid, the applicant must resubmit the correct documents. Once the documents are validated, the Verification Team conducts the verification process. Successful verification updates the application's status to verified. Simultaneously, the Police Department performs a police enquiry based on the provided documents. If the enquiry is successful, the application proceeds to the approval stage. Conversely, if the enquiry fails, the application is rejected. This diagram effectively illustrates the coordinated interactions and decision points among the applicant, verification team, and police department, ensuring a streamlined and efficient passport application process.