Ex. No: 01 Date:	**Identify a Software System that needs to be Developed Passport Automation System**

Aim:

To identify a software system that needs to be developed – Passport Automation System.

Introduction:

Passport automation systems represent a significant advancement in streamlining the process of issuing passports, ensuring efficiency, security, and convenience for applicants and governmental authorities alike. These systems leverage modern technologies, including biometrics, secure databases, and automated workflows, to simplify the application, verification, and issuance procedures.

Feature of Passport Automation System:

1. **Online Application Portals**: Passport automation systems typically offer user-friendly online portals where applicants can initiate and complete their passport applications from any internet-enabled device. These portals guide users through the application process, ensuring completeness and accuracy of information.
2. **Biometric Authentication**: Advanced passport automation systems utilize biometric technologies such as fingerprint scanning, facial recognition, or iris scanning to verify the identity of applicants. Biometric data enhances security by providing a unique identifier that is difficult to forge or replicate.
3. **Automated Document Verification:** Automation streamlines the process of verifying supporting documents submitted with passport applications. Optical Character Recognition (OCR) technology and document analysis algorithms automatically extract and verify information from documents such as birth certificates, proof of address, and identity cards.
4. **Real-Time Application Tracking**: Passport automation systems allow applicants to track the status of their applications in real-time. Through online portals or mobile applications, applicants can monitor the progress of their applications from submission to processing and issuance, providing transparency and peace of mind.
5. **Appointment Scheduling**: Integrated appointment scheduling features enable applicants to book appointments for passport application submission, document verification, or

biometric data collection at designated service centers. This helps manage crowd flow and ensures efficient utilization of resources.

6. **Secure Database Management**: Passport automation systems employ secure databases to store and manage applicant information and passport records. Advanced encryption protocols and access controls safeguard sensitive data from unauthorized access, ensuring compliance with data protection regulations.

Result:

Thus the software system that is need to be developed - Passport Automation system was identified successfully.

Ex. No: 02 **Date:**	**Document the software requirements specification (SRS) for the identified system**

Aim:

To implement a Software Requirement Specification (SRS) for Passport Automation System.

1. Problem Statement:

- Passport Automation System is used in the effective dispatch of passport to all of the applicants. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner
- The core of the system is to get the online registration form (with details such as name, address etc.,) filled by the applicant whose testament is verified for its genuineness by the passport automation system with respect to the already existing information in the database.
- This forms the first and foremost step in the processing of passport application. After the first round of verification done by the system, the information is in turn forwarded to the regional administrator's (ministry of external affairs) office.
- The application is then processed manually based on the report given by the system, and any forfeiting identified can make the applicant liable to penalty as per the law.
- The system forwards the necessary details to the police for its separate details to the police for its separate verification whose report is then presented to the administrator. After all the necessary criteria have been met, the original information is added to the database and the passport is sent to the applicant.

Software Requirements Specification:

2. INTRODUCTION:

Passport automation systems represent a significant advancement in streamlining the process of issuing passports, ensuring efficiency, security, and convenience for applicants and governmental authorities alike. These systems leverage modern technologies, including biometrics, secure databases, and automated workflows, to simplify the application, verification, and issuance procedures.

2.1 PURPOSE:

Passport Automation System is an application that is used to deliver the passport to the required users through an online registration process. It aims at improving the efficiency in the issue of passport and reduce the complexities involves in it to the maximum possible extent.

2.2 Document Conventions:

	Font	Style	Size
Heading	Times New Roman	Bold	16
Sub-Heading	Times New Roman	Bold	14
Others	Times New Roman	Regular	12

2.3 INTENDED AUDIENCE AND READING SUGGESTIONS:

This SRS is mainly development team. In this team there are the project manager, developer, coder, tester and documentation writer and the user of the project also.

User (Customer):

This document is intended to user and customer to make ensure that the document satisfies the needs of the customer.

Project Manager:

This SRS document is also very important for the project manager as it helps in cost estimation which can be performed by referring to the SRS document and it contains all the information that is required for the project plan.

Project Developer:

The project developer will refer to the SRS document to ensure that the product developed is as per the needs of the customer.

Tester:

The tester reads the SRS document to ensure that the requirements are understandable based on the functionality specified so that he can test the software and validates its working.

Document Writer:

The document writer reads the SRS document to ensure that they understand the document well enough and write user manuals based on the SRS document.

Maintenance:

The SRS document helps the maintenance engineers to understand functionality of the system. A clear knowledge of the functionality is needed to design and code.

2.4 PRODUCT SCOPE:

- The system provides an online interface to the user where they can fill in their personal details and submit the necessary documents.
- The authority concerned with the issue of passport can use this system to reduce his workload and process the application in a speedy manner.
- Provide a communication platform between the applicant and the administrator.
- Transfer of data between the passport issuing authority and the local police for verification of applicant's information.
- Users/applicants will come to know their status of application and the date in which they must subject themselves for manual document verification.

2.5 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS:

- **Administrator:** Refers to the use who is the central authority who has been vested with the privilege to manage the entire system. It can be any higher official in the regional passport office of ministry of external affairs.
- **Applicant:** One who wishes to obtain the passport.
- **PAS:** Refers to the passport automation system.
- **HTML:** Markup Language user for creating web pages.
- **J2EE:** Java 2 Enterprise Edition.

3. OVERALL DESCRIPTION:

3.1 PRODUCT PERSPECTIVE:

The passport automation system acts as an interface between the applicant and the administrator. This system tries to make the interface as simple as possible and at the same time not risking the security of data stored in. This minimizes the time duration in which the user receives the passport.

3.2 PRODUCT FUNCTION:

- Secure registration of information by the applicants.
- Schedule the applicants an appointment for manual verification of original documents.
- Panel for passport application status display by the administrator.
- SMS and mail updates to the applicants by the administrator.
- Administrator and generate reports from the information and is the only authorised personnel to add the eligible application information to the database.

3.3 USER CLASSES AND CHARACTERISTICS:

- **Applicant:** They are the people who desires to obtain the passport and submit the information to the database.
- **Administrator:** He has the certain privileges to add the passport status and to approve the issue of passport. He may contain a group of persons under him to verify the documents and give suggestion whether or not to approve the dispatch of passport.
- **Police:** He is the person who upon receiving intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present. He has been vetoed with the power to decline an application by suggesting it to the administrator if he finds any discrepancy with the applicant. He communicates via this PAS.

3.4 OPERATING ENVIRONMENT:

Particulars	Client System	Server System
Operating System	Windows/Linux/Android	Linux
Processor	Intelor AMD	Intelor AMD
Hard Disk	1GB	1TB
RAM	256 MB	8GB

3.5 DESIGN AND IMPLEMENTATION CONSTRAINTS:

- The applicants require a computer to submit their information
- The user has to be careful while submitting the information.

- Although the security is given high importance there is always a chance of intrusion in the web world which requires constant monitoring.

3.6 ASSUMPTIONS AND DEPENDENCIES:

- The applicants and administrator must have basic knowledge of computers and English language.
- The applicants may be required to scan the documents to send.
- Each user must have a user id and password.
- Internet connection is must.

4. EXTERNAL HARDWARE REQUIREMENTS:

4.1 USER INTERFACE:

- Applicant: They are the people who desires to obtain the passport and submit the information to the database.
- Administrator: He has the certain privileges to add the passport status and to approve the issue of passport
- Police: He is the person who upon receiving intimation from the PAS, perform a personal verification of the applicant and see if he has any criminal case against him before or at present.

4.2 Hardware Interface:

The server is directly connected to the client systems. The client systems have access to the database in the server.

4.3 Software Interfaces:

- Front End Client: The applicant and administrator online interface is built using JSP and HTML. The administrator's local interface is built using Java.
- Web Server: Glassfish application server
- Back End: Oracle database.

4.4 Communications Interface:

Online passport registration user internet. Hence, it uses HTTP for transmission of data. This protocol allows easy interaction between client and server.

5. SYSTEM FUNCTION:

- Secure registration of information by the applicants
- Schedule the applicants an appointments for manual verification of original documents.
- Panel for passport application status display by the administrator.
- SMS and mail updates to the applicants by the administrator.

6. OTHER NON-FUNCTIONAL REQUIREMENTS:

6.1 Performance requirements:

- The response time of the system should be less. The applicant under the criminal act are not allowed to issue passport.
- Sometimes the workload will be high, the is in certain period it will be less, then it should be managed properly by employing more staffs to process the system.

6.2 Security Requirements:

Every user is provided with unique ID within their password. Every user is authentication before accessing their account. If authentication doesn't provided then illegal usage of passport will occur.

6.3 Software Quality Attributes:

The system is highly reliable. The system is also adaptive under any conditions

6.4 Business Rules:

To get the passport, the address proof and age proof should be provided and the applicant should be free of criminal cases.

RESULT:

Thus the Software Requirement Specification (SRS) for Passport Automation System was implemented successfully.

Ex. No: 03 Date:	Identify use cases and develop the Use Case model

Aim:

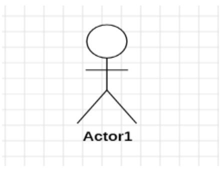
To identify the use cases and develop use case model.

Introduction of Use Case Diagram:

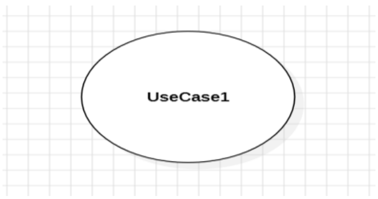
A use case diagram is a type of behavioural diagram in the Unified Modelling Language (UML) that represents the interaction between a system and its actors in terms of the system's behaviour by illustrating how users interact with the system to accomplish specific tasks or goals. Use case diagrams facilitate communication between stakeholders, including developers, designers, and client by providing a clear and concise representation of the system's functionality.

Components of Use Case Diagram:

Actors: Actors represent the users or external systems interacting with the system being modular. In the Passport Automation System, the actors include the user, regional administrator, passport administrator, police and the database.



Use Cases: Use cases represent the functionalities or services provided by the system. Each use case describes a specific interaction between an actor and the system to achieve a particular goal. In the Book bank management system, examples of use cases include "login, check status, issue passport".



Relationships: Relationships, such as associations and dependencies, depict the connections between actors and use cases. Associations represent the interaction between actors and use cases, while dependencies represent the reliance of one use case on another

Benefits of Use Case Diagram:

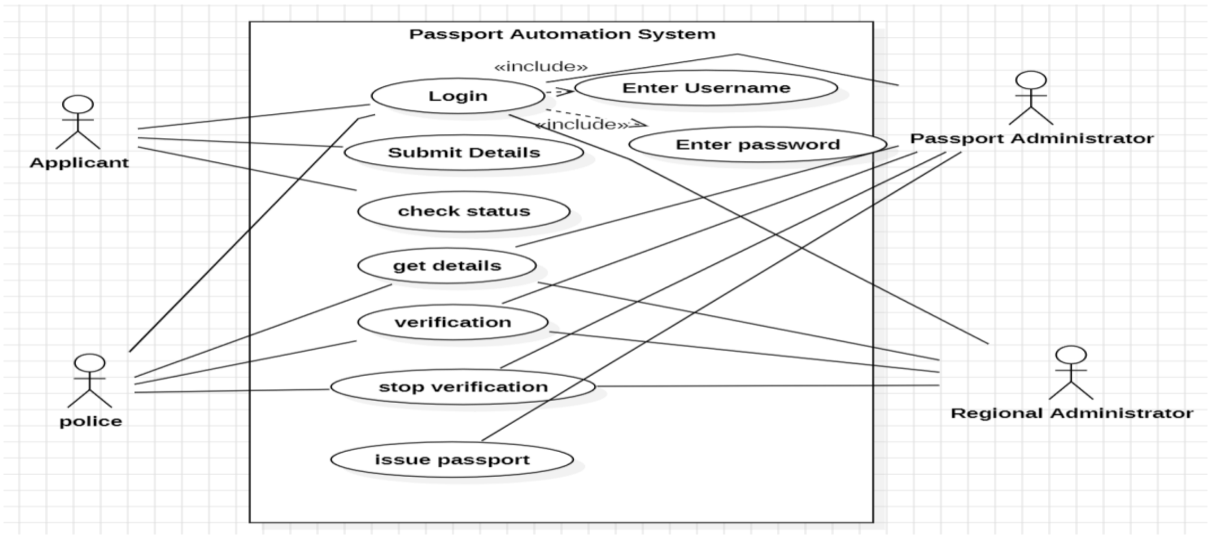
Requirements Elicitation: Use case diagrams help in eliciting and defining the functional requirements of a system by identifying the interactions between users and the system.

Visualization: Use case diagrams provide a visual representation of the system's functionality, making it easier for stakeholders to understand and analyse the system's behaviour.

Communication: Use case diagrams serve as a communication tool between stakeholders, including developers, designers, and clients, by providing a common understanding of the system's requirements and behaviour.

Requirements Verification: Use case diagrams can be used to verify whether all the necessary functionalities and interactions have been captured and addressed in the system design.

Use Case Diagram for Passport Automation System:



Applicant: Initiates the passport application process and tracks the application status.

- Apply for Passport: Applicant applies for a passport.
- Track Application: Applicant checks the status of their passport application.

Passport Admin: Manages the passport application process and issues passports.

- Submit: Receives passport applications.

- View status: Checks the status of passport applications.
- Issue Passport: Issues passports to approved applicants.

Police: Conducts verification checks for passport applications.

- Perform: Performs police verification for applicants.

Regional Admin: Manages administrative tasks related to passport issuance.

- Manage: Manages administrative tasks related to passport issuance, such as coordinating with passport admin and overseeing regional operations.

RESULT:

Thus the use case diagram for Passport Automation system is implemented and executed successfully.

Ex. No: 04 **Date:**	**Identify the conceptual classes and develop a domain model and also derive a class diagram from that**

Aim:

To identify the conceptual classes and develop a domain model and derive class diagram for Book bank management system.

Conceptual Classes:

Applicant: Represents individuals who apply for passports.

Police: Represents law enforcement agencies responsible for conducting background checks and verification for passport applicants.

Database: Represents the storage and management system for passport-related data.

Regional Admin: Represents administrative personnel responsible for managing regional operations related to passport issuance.

Passport Admin: Represents administrative personnel responsible for managing overall passport issuance processes.

Domain Model for Passport Automation System:

This domain model represents the core entities and their relationships within a book bank management system. Depending on the specific requirements of the system, you may need to further refine and expand upon these entities.

Applicant:

- Name: string
- Father name: string
- Date of Birth: String
- Permanent Address: String
- Temporary Address: String
- Email ID: String
- Phone No: Integer

- Pan No: Integer
- Aadhar No: Integer
- Application No: Integer
- Username: String
- Password: String

Operations:

- Login()
- Checkstatus()
- Submitdetails()

Police:

Attributes:

- Username: String
- Password: String

Operations:

- Login()
- Verify()
- Update()

Database:

Attributes:

- Username: String
- Password: String

Operations:

- Store()

Passport Administrator:

Attributes:

- Username: String
- Password: String

Operations:

- Login()
- Verify()
- Update()
- IssuePassport()

Regional Administrator:

Attributes:

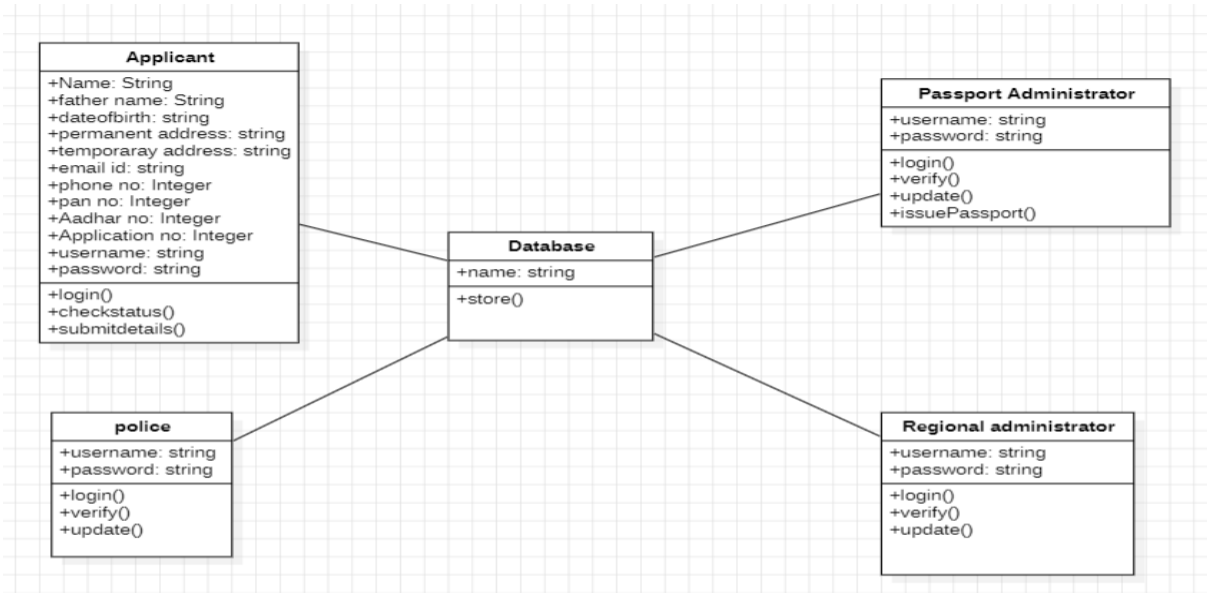
- Username: String
- Password: String

Operations:

- Login()
- Verify()
- Update()

In this domain model for the Passport Automation System outlines key entities and their interactions. It includes the Applicant class for individuals applying for passports, the Police class responsible for verification, the Passport Admin class managing issuance, the Database class for data storage, and the Regional Admin class overseeing regional operations. These entities collaborate to ensure a streamlined and secure process from application submission to passport issuance.

Class diagram for the Passport Automation System:



RESULT:

Thus the class diagram for Passport Automation system is implemented and executed successfully.

Ex. No: 05 Date:	**Using the identified scenarios, find the interaction between objects and represent them using UML sequence and collaboration diagram.**

Aim:

To find the interaction between the objects and represent them using UML Sequence and Collaboration diagram.

INTRODUCTION OF INTERACTION DIAGRAMS:

• Overview of Interaction Diagrams:

1. Interaction diagrams are graphical representations in UML (Unified Modelling Language) used to visualize the dynamic behaviour of a system by showing how objects interact over time.

2. These diagrams capture the flow of messages or actions between objects during the execution of a system or process.

• Categorization into Two Types:

3. Interaction diagrams are categorized into two main types: Activity Diagrams and Sequence Diagrams.

4. Activity Diagrams: These diagrams focus on modelling the flow of activities or tasks within a system, showing the sequence of actions and decisions.

5. Sequence Diagrams: These diagrams focus on modelling the chronological order of messages exchanged between objects in a system, illustrating how objects interact over time.

SEQUENCE DIAGRAM:

- A Sequence diagram is a type of interaction diagram in UML that represents the interactions between objects over time.

- It illustrates the sequence of messages exchanged between objects within a system to accomplish a specific task or scenario.

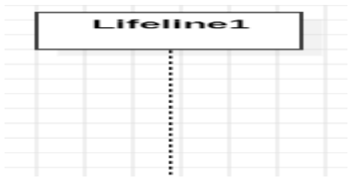
Components of a Sequence Diagram:

1. Lifelines:

• Lifelines represent the lifespan of objects participating in the interaction within the system.

• Each object involved in the interaction is represented by a lifeline, which is depicted as a vertical dashed line.

• The length of the lifeline represents the duration of the object's existence or involvement in the interaction.

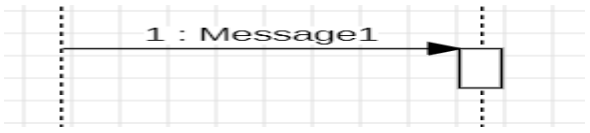


2. Messages:

• Messages represent communication or interactions between objects in the system.

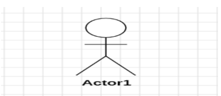
• They depict the flow of control or data between objects during the execution of a scenario.

• Messages are represented by arrows that connect lifelines, indicating the direction of communication.

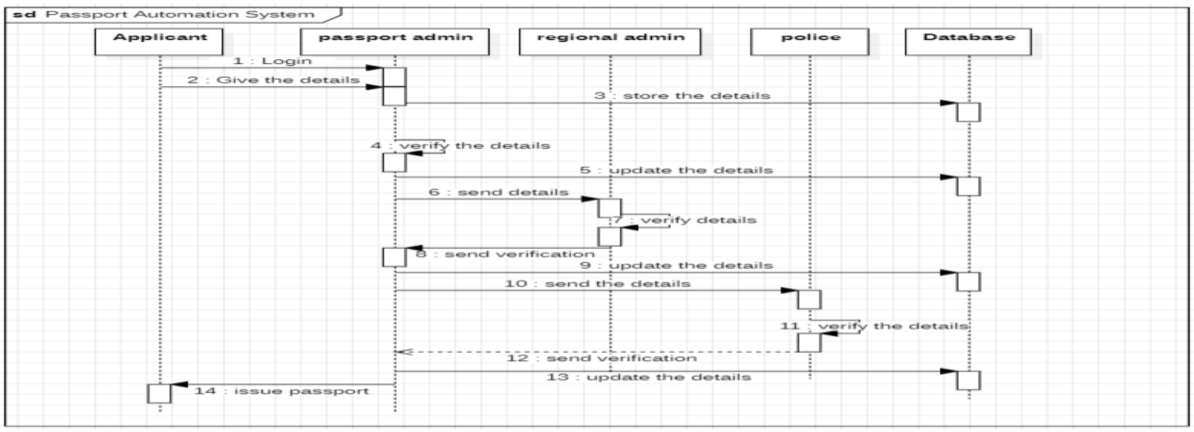


3. Actors:

An UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.



SEQUENCE DIAGRAM FOR PASSPORT AUTOMATION SYSTEM:



COMMUNICATION DIAGRAM:

A communication diagram in UML (Unified Modelling Language) is a type of interaction diagram that visualizes the interactions between objects or parts in terms of sequenced messages. It emphasizes the structural organization of the objects and how they collaborate to achieve a specific task or behaviour. Communication diagrams are particularly useful for understanding the dynamic aspects of a system's behaviour during runtime.

Communication Diagram Notations:

1. Object/Participants: Objects are represented by rectangles with the object’s name at the top. The diagram shows each object participating in the interaction as a separate rectangle. Objects are connected by lines to indicate messages being passed between them.

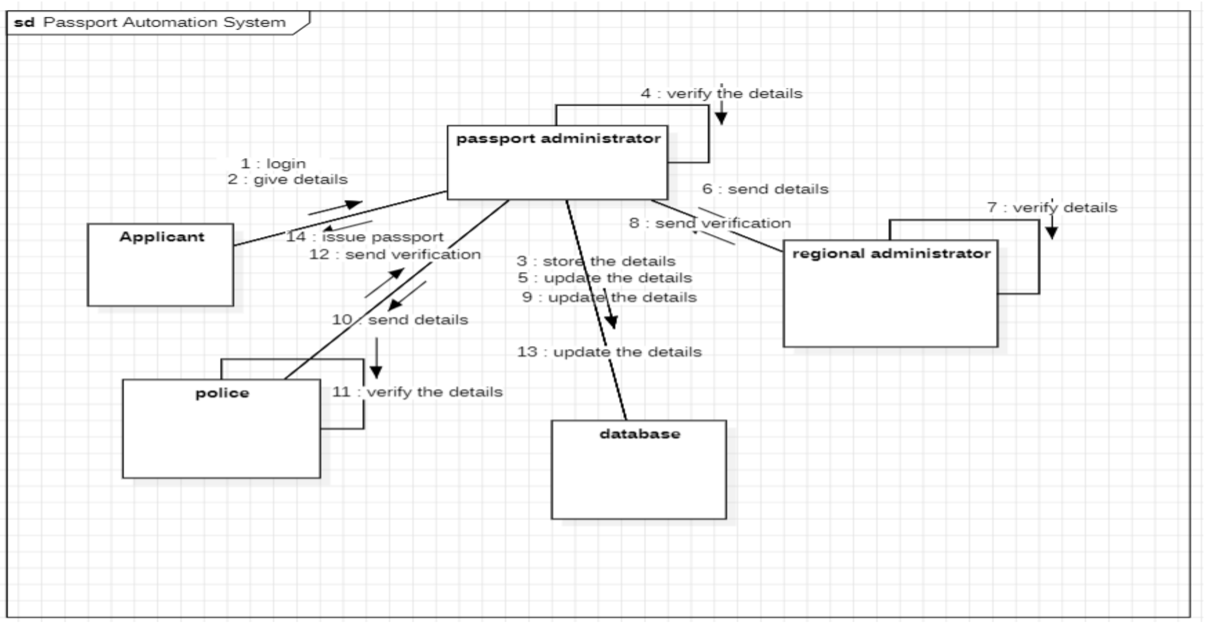
Object Name

2. Actors: They are usually depicted at the top or side of the diagram, indicating their involvement in the interactions with the system’s objects or components. They are connected to objects through messages, showing the communication with the system.

3.Messages:

Messages represent communication between objects. Messages are shown as arrows between objects, indicating the flow of communication. Each message may include a label indicating the type of message (e.g., method call, signal). Messages can be asynchronous (indicated by a dashed arrow) or synchronous (solid arrow).

COMMUNICATION DIAGRAM FOR PASSPORT AUTOMATION SYSTEM:



Result:

Thus to find the interaction between objects and represent them using UML Sequence and Collaboration diagram was done successfully.

Ex. No: 06 Date:	Draw relevant state chat and activity diagram for the same system

Aim:

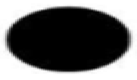
To draw State Chart diagram and Activity diagram for Book Bank System.

State Chart Diagram:

A state chart diagram in UML (Unified Modelling Language) is a behavioural diagram that depicts the different states that an object or system undergoes during its lifetime and the transitions between these states triggered by events. State chart diagrams are particularly useful for modelling the behaviour of complex systems with multiple states and transitions.

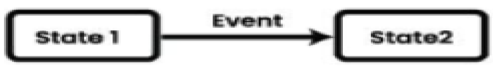
State Chart Diagram Notation:

1. Initial State:



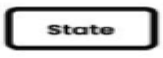
We use a black-filled circle to represent the initial state of a System or a Class.

2. Transition:



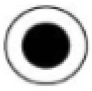
We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

3. State:



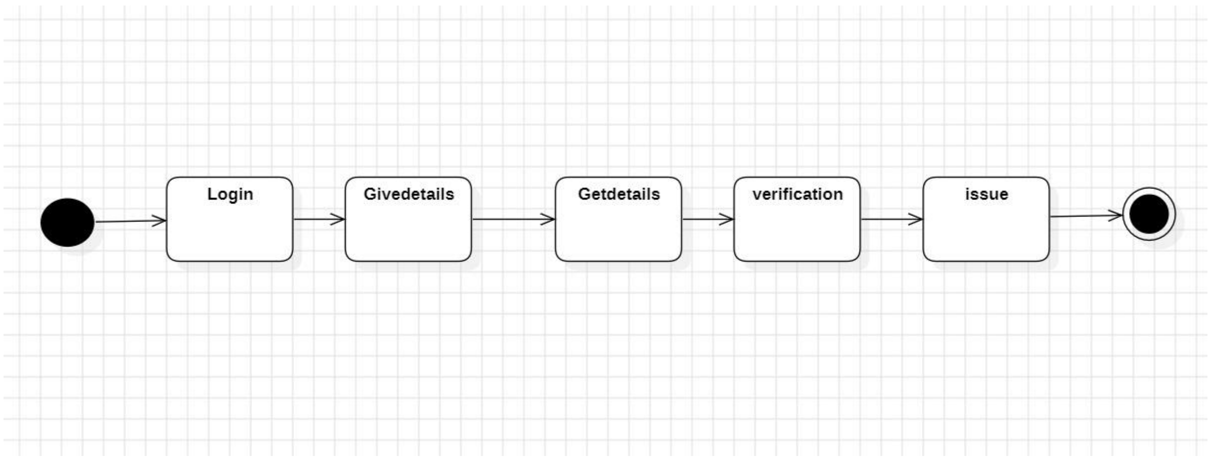
We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

4. Final State:



We use a filled circle within a circle notation to represent the final state in a state machine diagram.

State Chart Diagram for Passport Automation System:



ACTIVITY DIAGRAM:

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We can depict both sequential processing and concurrent processing of activities using an activity diagram i.e., an activity diagram focuses on the condition of flow and the sequence in which it happens.

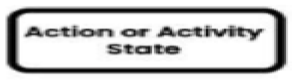
- We describe what causes a particular event using an activity diagram.
- An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.
- They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.

Activity Diagram Notation:

1. State:

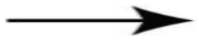
A process can have only one initial state unless we are depicting nested activities. We use a black-filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

2. Action or Activity State:



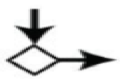
An activity represents the execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically, any action or event that takes place is represented using an activity.

3. Action Flow or Control Flow:



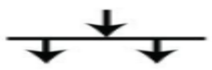
Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another activity state.

4. Decision Node or Branching:



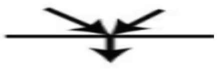
When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.

5. Fork:



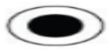
Fork nodes are used to support concurrent activities. When we use a fork node both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts.

6. Join:



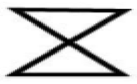
Join nodes are used to support concurrent activities converging into one. We have two or more incoming edges and one outgoing edge for join notations.

7. Final State:



The state that the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

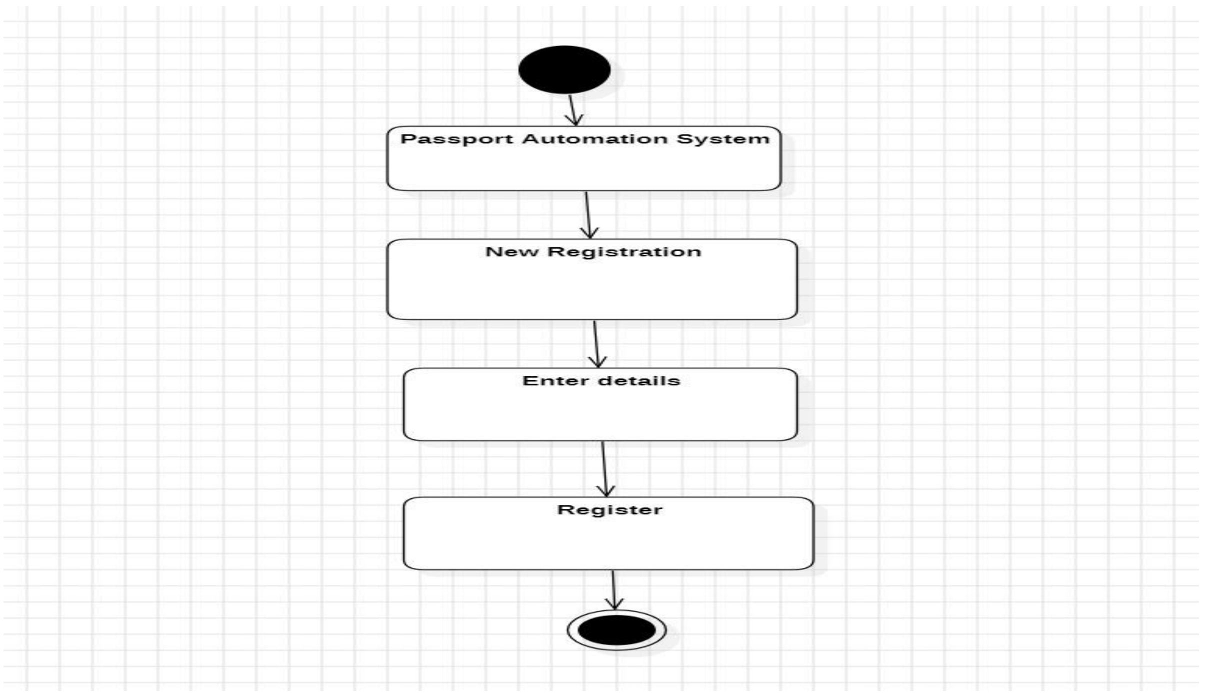
8. Time Event:



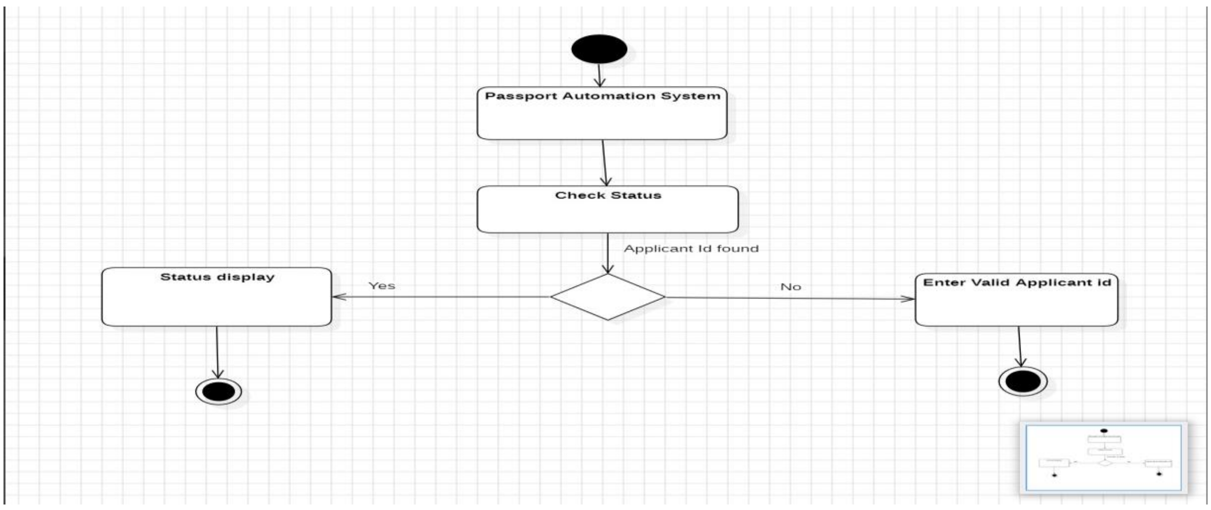
This refers to an event that stops the flow for a time; an hourglass depicts it. We can have a scenario where an event takes some time to complete.

ACTIVITY DIAGRAM FOR PASSPORT AUTOMATION SYSTEM:

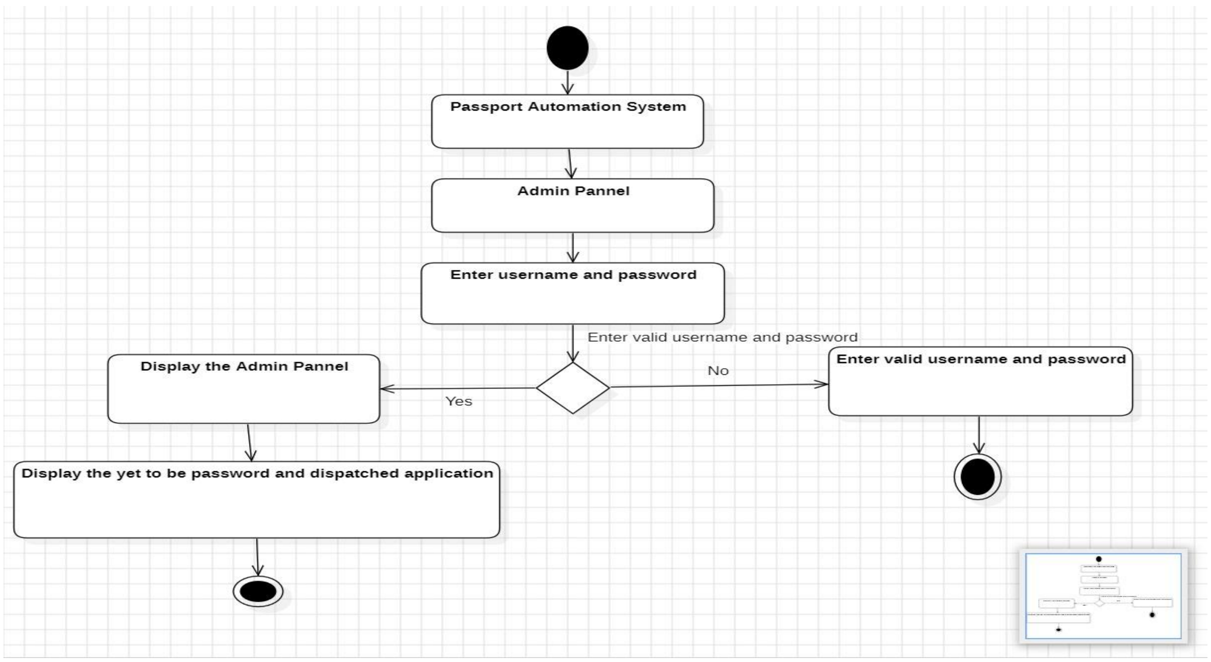
New Registration:



Check Status:



Admin Panel:


Passport Automation System
Admin Pannel
Enter username and password
Enter valid username and password
Display the Admin Pannel
No
Enter valid username and password
Yes
Display the yet to be password and dispatched application

RESULT:

Thus to draw State Chart diagram and Activity diagram for Passport Automation System was done successfully.

Ex. No: 07 Date:	Implement the system as per the detailed design

Aim:

To implement Book Bank System as per the detailed design.

Introduction:

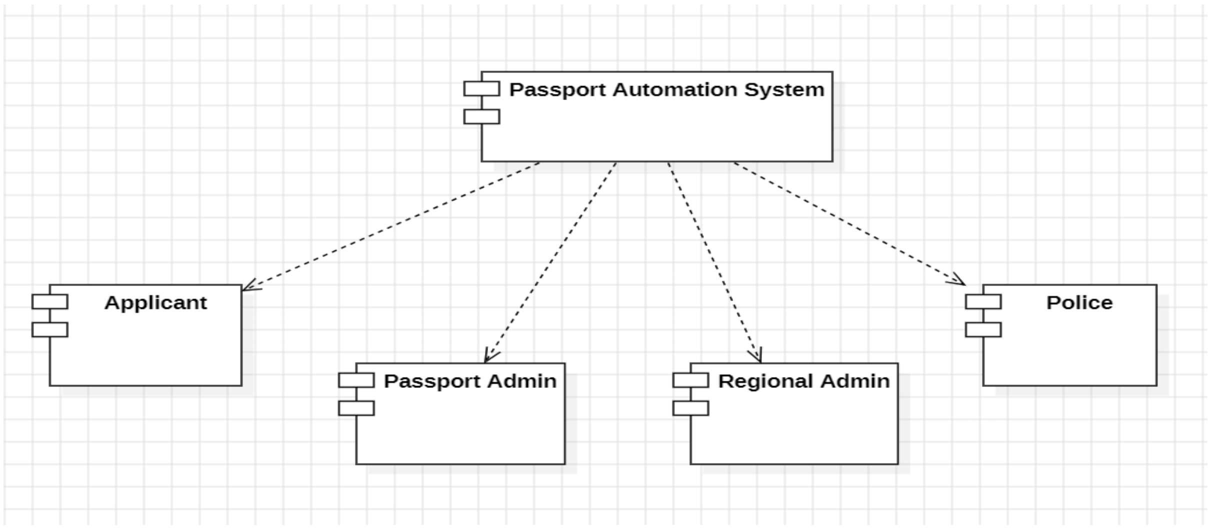
The Passport Automation System implementation plan details a strategic approach to modernize and optimize passport issuance processes. It covers technology selection, development steps, coding standards, documentation, version control, security, and continuous integration practices. With a focus on enhancing efficiency, scalability, and user experience, the plan aims to create a secure and reliable system that streamlines passport management operations.

Implementation Diagram:

Component Diagram:

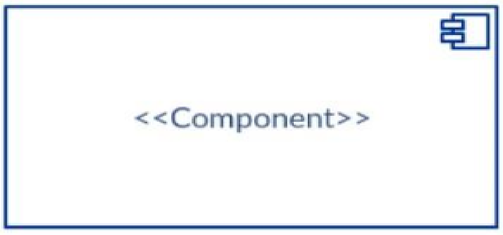
A component diagram is a type of UML (Unified Modeling Language) diagram used to illustrate the structure of a software system by showing the system's components and their relationships. It helps in visualizing the modular structure of a system and the interactions between its components.

Component Diagram for Passport Automation System:



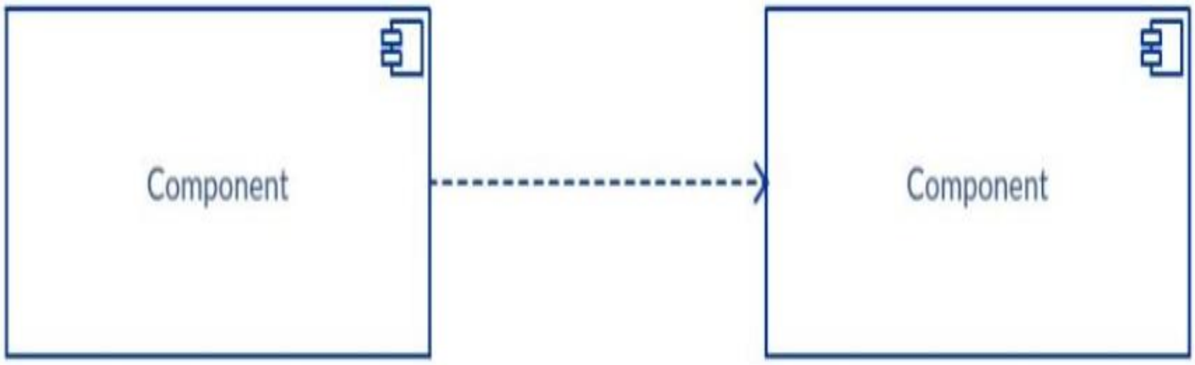
COMPONENT DIAGRAM NOTATION:

1. Component:



Rectangle with the component icon in the top right corner and the name of the component.

2. Dependencies:

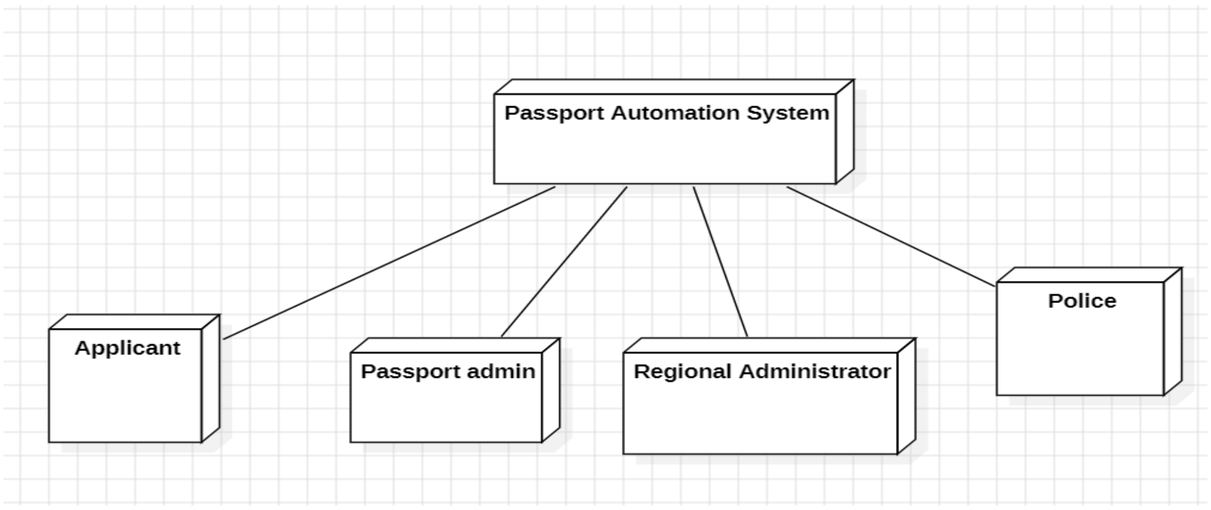


Although you can show more detail about the relationship between two components using the ball-and-socket notation (provided interface and required interface), you can just as well use a dependency arrow to show the relationship between two components.

DEPLOYMENT DIAGRAM:

A deployment diagram is a type of UML (Unified Modeling Language) diagram that illustrates the physical deployment of software components in a system, showing how software artifacts are deployed onto hardware nodes and interconnected through networks. It depicts the configuration and arrangement of system elements such as servers, computers, devices, and communication channels, providing insight into the system's deployment architecture and infrastructure.

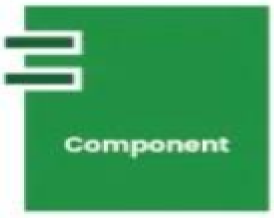
Deployment Diagram for Passport Automation System:



DEPLOYMENT DIAGRAM NOTATIONS:

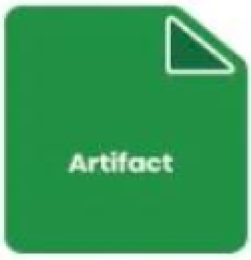
1. Component:

A component represents a modular and reusable part of a system, typically implemented as a software module, class, or package. It encapsulates its behaviour and data and can be deployed independently.



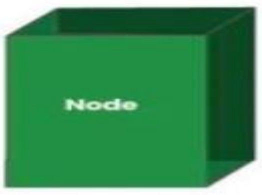
2. Artifact:

An artifact represents a physical piece of information or data that is used or produced in the software development process. It can include source code files, executables, documents, libraries, configuration files, or any other tangible item.



3. Node:

A node represents a physical or computational resource, such as a hardware device, server, workstation, or computing resource, on which software components can be deployed or executed.



Code generation from class diagram:

Applicant:

```
import java.io.*;

import java.util.*;

public class Applicant {


   /**

    * Default constructor

    */

   public Applicant() {

   }

   public String Name;

   public String father_name;

   public String dateofbirth;

   public String permanent_address;

   public String temporaray_address;

   public String email_id;

   public Integer phone_no;

   public Integer pan_no;
```

```
    public Integer Aadhar_no;

    public Integer Application_no;

    public String username;

    public String password;

    public void login() {

        // TODO implement here

    }

    public void checkstatus() {

        // TODO implement here

    }

public void submitdetails() {    }

}
```

Database:

```
import java.io.*;

import java.util.*;

public class Database {

public Database() {

    }

    public String name;

    public void store() {

        // TODO implement here

    }

}
```

Passport Admin:

```
import java.io.*;
import java.util.*;
public class Passport Administrator {
   public Passport Administrator() {
   }
   public String username;
   public String password;
   public void login() {
      // TODO implement here
   }
   public void verify() {
      // TODO implement here
   }
   public void update() {
      // TODO implement here
   }
   public void issuePassport() {
      // TODO implement here
   }}
```

Police:

```
import java.io.*;
import java.util.*;
public class police {
    public police() {

    }

    public String username;

    public String password;

    public void login() {

        // TODO implement here

    }

    public void verify() {

        // TODO implement here

    }

    public void update() {

        // TODO implement here

    }

    public enum Enumeration1 {

    }


}
```

Regional Admin:

```
import java.io.*;

import java.util.*;

public class Regional administrator {

    public Regional administrator() {

    }

    public String username;
    public String password;

    public void login() {

        // TODO implement here

    }

    public void verify() {

        // TODO implement here

    }

    public void update() {

        // TODO implement here

    }

}
```

Result:

Thus the Passport Automation system was implemented successfully.

Ex. No: 08 Date:	**Test the software system for all scenarios identified as per the use case diagram**

Aim:

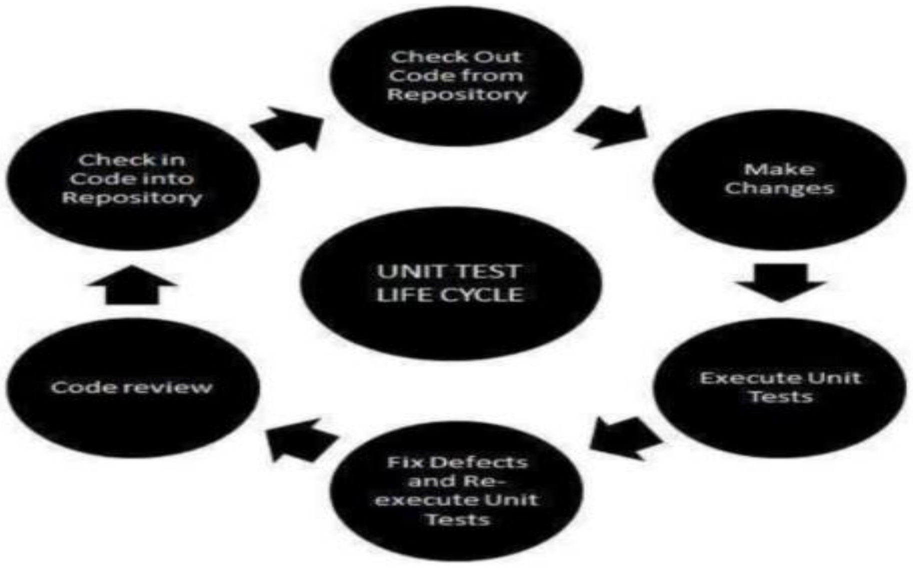
To test the software system for all scenarios identified in the use case diagram.

Passport Automation System:

Passport automation systems represent a significant advancement in streamlining the process of issuing passports, ensuring efficiency, security, and convenience for applicants and governmental authorities alike. These systems leverage modern technologies, including biometrics, secure databases, and automated workflows, to simplify the application, verification, and issuance procedures.

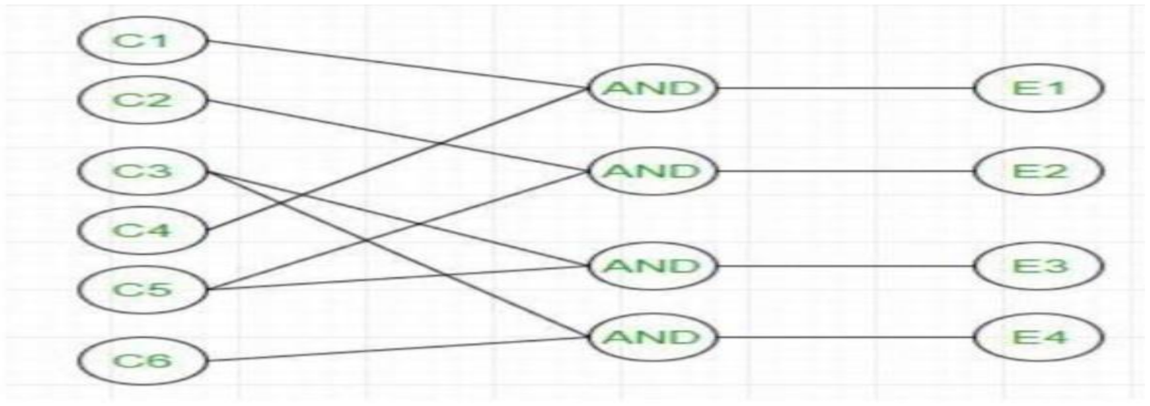
Unit Testing:

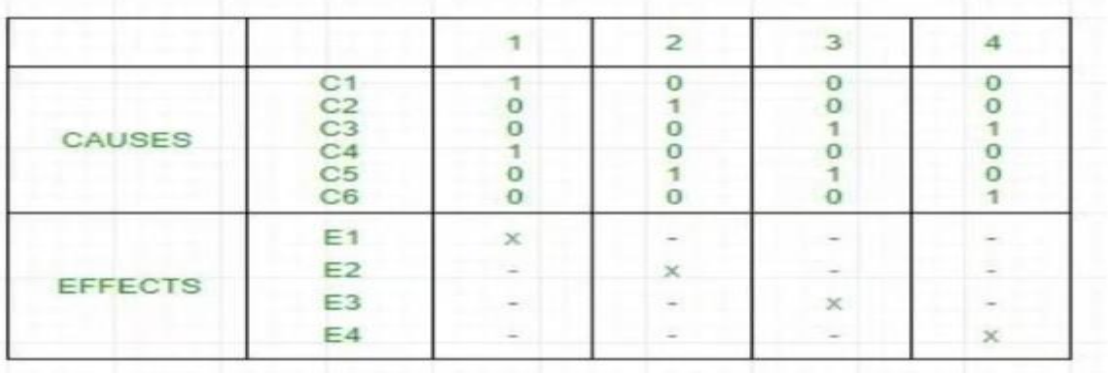
- **Purpose:** Unit testing ensures that individual modules or functions of the passport automation system work correctly in isolation.
- **Process:** Developers write automated tests to validate the behaviour of specific functions or modules, such as creating a new passport record, updating personal information, or validating passport data.
- **Example:** Unit tests for the passport automation system might include verifying the functionality of methods for capturing biometric data, checking passport expiration dates, or validating input fields.


Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review
Check in Code into Repository
UNIT TEST LIFE CYCLE

Black Box Testing:

- **Purpose:** Black box testing assesses the functionality of the passport automation system without knowing its internal workings.
- **Process:** Testers interact with the system's user interface and provide inputs to simulate real-world usage scenarios. They verify outputs to ensure they align with expected results without knowledge of the system's internal logic.
- **Example:** Black box testing for the passport automation system could involve simulating the application process by entering personal details, uploading documents, and verifying that the system correctly generates a passport application.

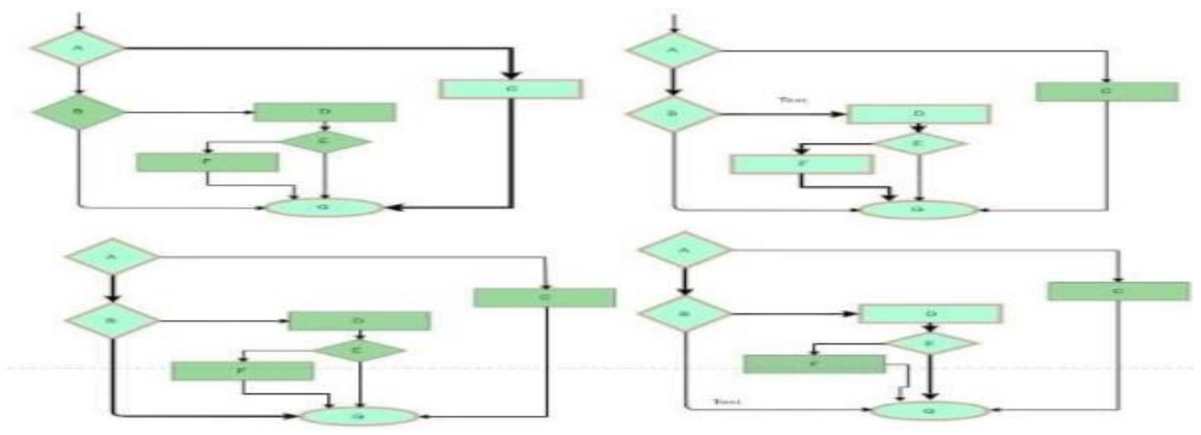




		1	2	3	4
CAUSES	C1	1	0	0	0
	C2	0	1	0	0
	C3	0	0	1	1
	C4	1	0	0	0
	C5	0	1	1	0
	C6	0	0	0	1
EFFECTS	E1	x	-	-	-
	E2	-	x	-	-
	E3	-	-	x	-
	E4	-	-	-	x

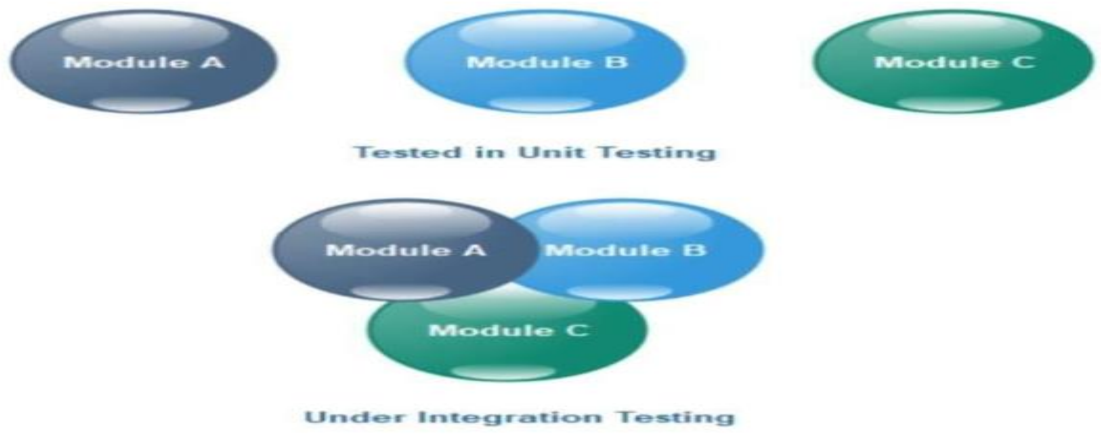
White Box Testing:

- **Purpose:** White box testing examines the internal logic, code structure, and paths within the passport automation system.
- **Process:** Testers have access to the system's source code and design test cases to exercise specific code paths, ensuring thorough coverage. They aim to identify any flaws or errors in the implementation.
- **Example:** White box testing for the passport automation system might involve examining code branches related to identity verification, encryption algorithms for sensitive data, or error handling during application processing.



Integration Testing:

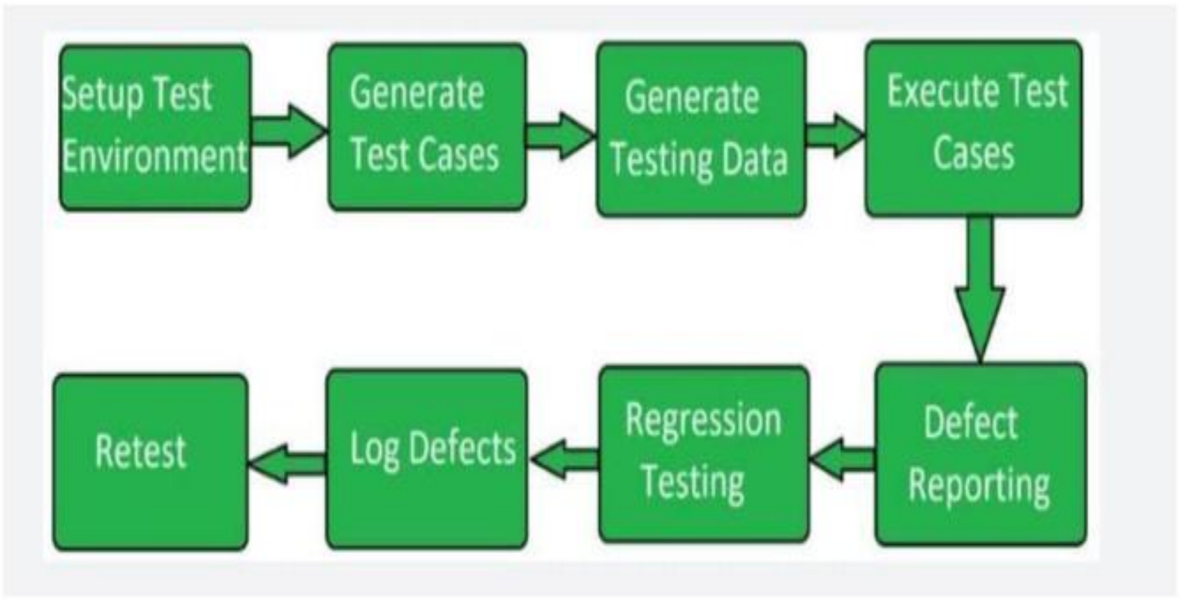
- **Purpose**: Integration testing validates that different modules or components of the passport automation system work together seamlessly.
- **Process:** Testers combine modules and assess their interactions to ensure smooth integration, focusing on interfaces between components.
- **Example:** In the passport management system, integration testing could involve testing the interaction between the user interface, database, and backend logic, while system testing could involve testing scenarios such as applying for passport, issuing of passport and updating user information.



System Testing:

- **Purpose:** System testing involves testing the entire system as a whole to verify that it meets the specified requirements.
- **Process:** Testers would perform end-to-end testing of the entire system, including all modules and their interactions.

- **Example:** Testing the entire process from client negotiation to project delivery and payment to ensure that it functions as expected.



Regression Testing:

- **Purpose:** Regression testing ensures that new updates or changes to the passport automation system do not introduce defects or regressions in existing functionality.
- **Process:** Testers re-run previously executed test cases to verify that existing features still operate correctly after updates. This helps maintain system stability and reliability.
- **Example:** Regression testing for the passport automation system could involve re testing functionalities like applying, verifying and issuing of passports after implementing new features or bug fixes.

Result:

Thus the software system for all scenarios identified in the use case diagram was tested successfully.

Ex. No: 09 **Date:**	**Improve the reusability and maintainability of the software system by applying appropriate design pattern**

Aim:

To improve the reusability and maintainability of the software by applying appropriate design patterns.

Passport Automation System:

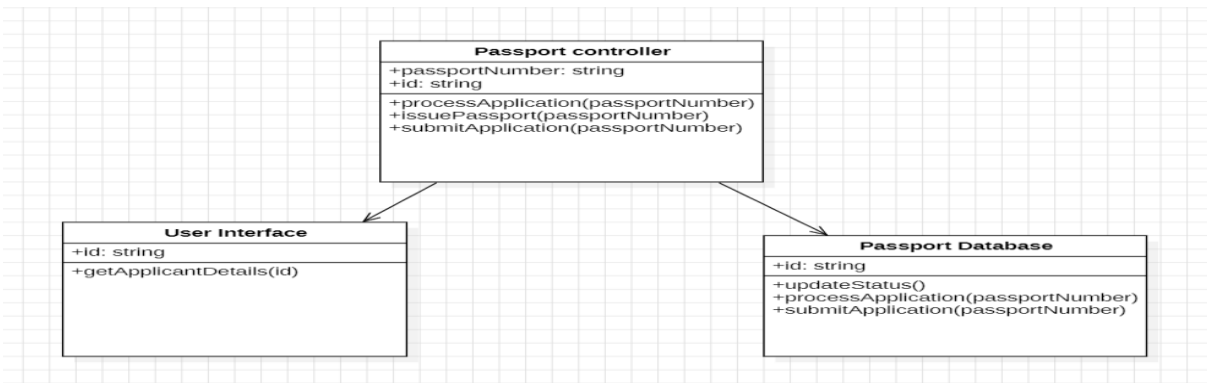
In today's global landscape, efficient passport automation systems are indispensable for managing passport issuance and renewal seamlessly. Enhancing reusability and maintainability in such systems necessitates a thoughtful integration of software design patterns. These patterns offer proven solutions to common design challenges, enabling the creation of flexible, scalable, and easily maintainable architectures. By exploring design patterns such as Factory Method, Singleton, Facade, and Strategy, among others, within the context of passport management, we aim to develop a system that not only fulfils immediate needs but also adapts seamlessly to future enhancements.

DESIGN PATTERNS:

Model-View-Controller Pattern:

Context/Problem: A passport automation system needs to manage the process of applying for, verifying, and issuing passports. The system should allow applicants to submit their applications online, track their application status, and receive notifications. It should also help administrators verify documents, process applications, and issue passports.

Solution: Implementing the MVC (Model-View-Controller) pattern to separate the concerns of the user interface, business logic, and data access.

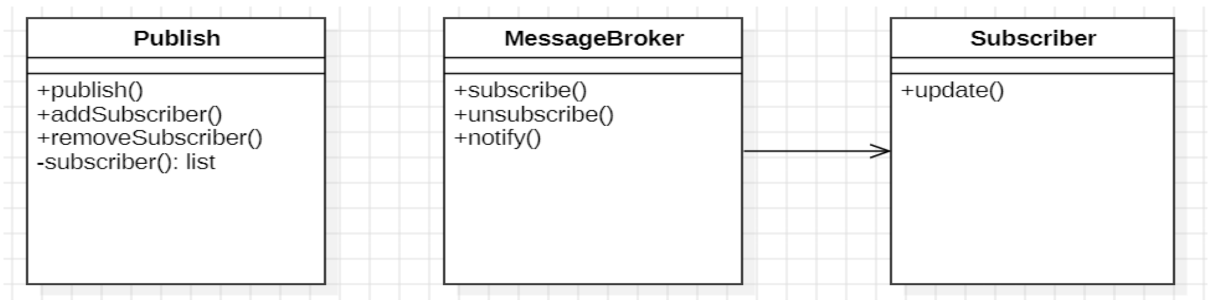

Passport controller
+passportNumber: string
+id: string
+processApplication(passportNumber)
+issuePassport(passportNumber)
+submitApplication(passportNumber)
User Interface
+id: string
+getApplicantDetails(id)
Passport Database
+id: string
+updateStatus()
+processApplication(passportNumber)
+submitApplication(passportNumber)

Publish-Subscribe Pattern:

Context/Problem: The system needs to send notifications to applicants about their application status updates (e.g., submitted, in review, approved).

Solution: Implementing the Publish-Subscribe pattern to decouple the notification mechanism from the core application processing logic.

Visual:

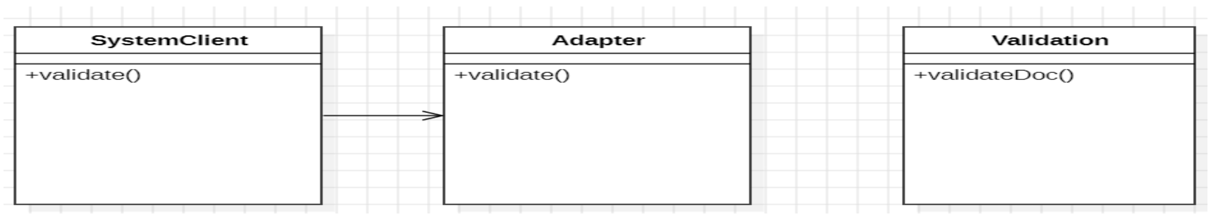


Adapter Pattern:

Context/Problem: The system needs to integrate with an external service for validating applicant documents, but the service interface is different from the system's expected interface.

Solution: Implementing the Adapter pattern to adapt the interface of the external document validation service to the system's expected interface.

Visual:

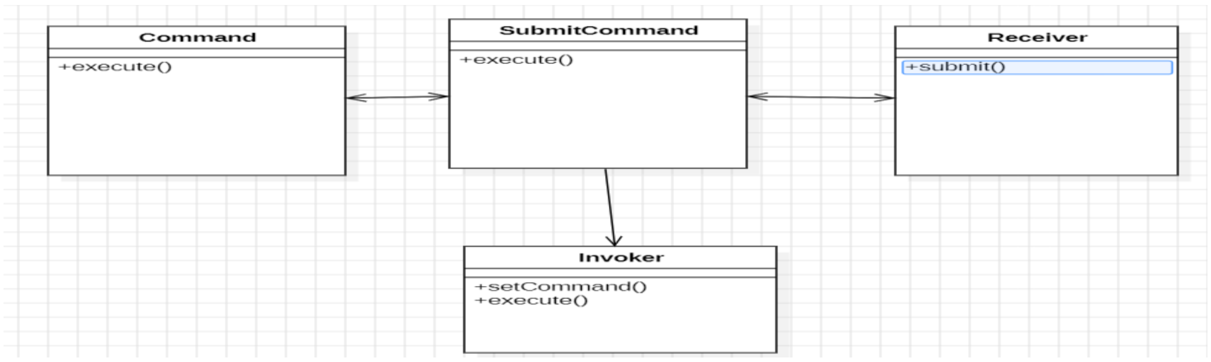


Command Pattern:

Context/Problem: The system needs to handle various operations (commands) like submit application, verify documents, issue passport, which should be executed, queued, or logged.

Solution: Implementing the Command pattern to encapsulate these operations as command objects.

Visual:

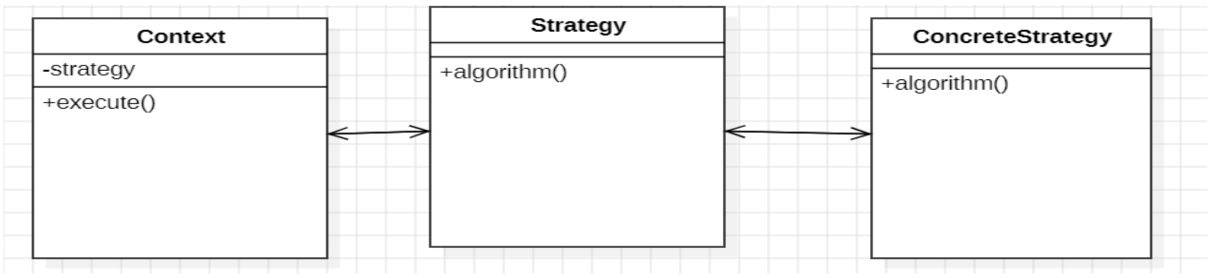


Strategy Pattern:

Context/Problem: The system needs to verify applicant documents using different strategies (e.g., local database, external API) based on the applicant's country or other criteria.

Solution: Implementing the Strategy pattern to encapsulate the different document verification algorithms.

Visual:

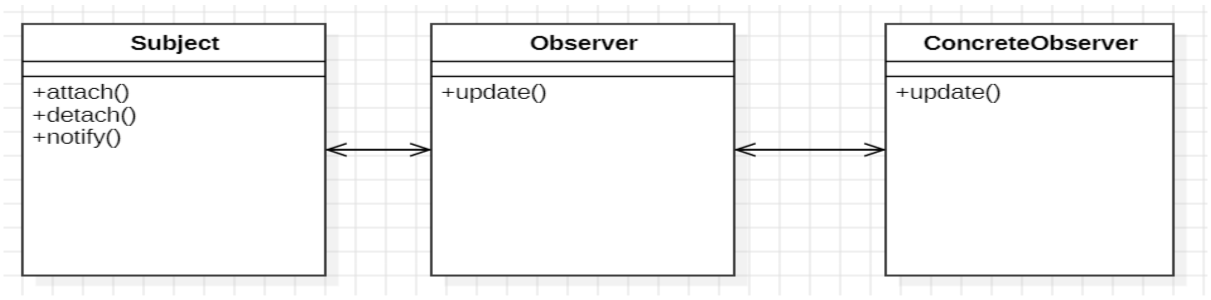


Observer Pattern:

Context/Problem: The system needs to notify multiple components (e.g., different modules, external systems) whenever an applicant's status changes.

Solution: Implementing the Observer pattern to define a one-to-many dependency between the status change event and its observers.

Visual:

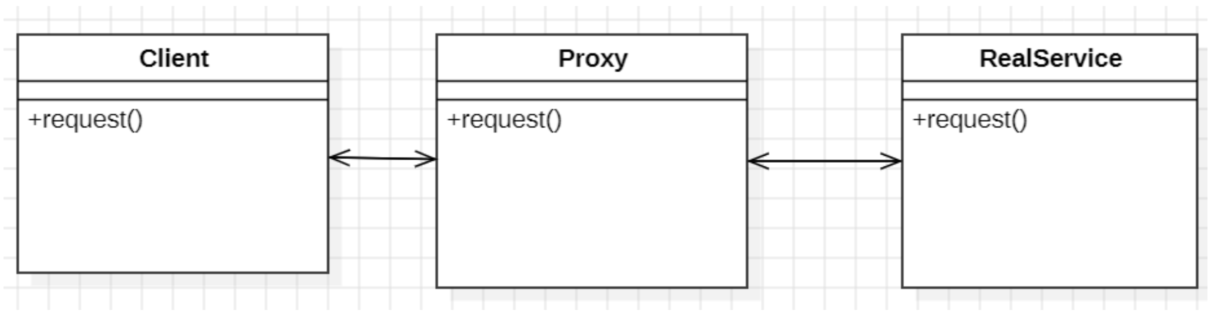


Proxy Pattern:

Context/Problem: The system needs to control access to a sensitive service, such as a passport issuance service, by adding an additional layer of validation and logging.

Solution: Implementing the Proxy pattern to provide a surrogate or placeholder for the passport issuance service.

Visual:

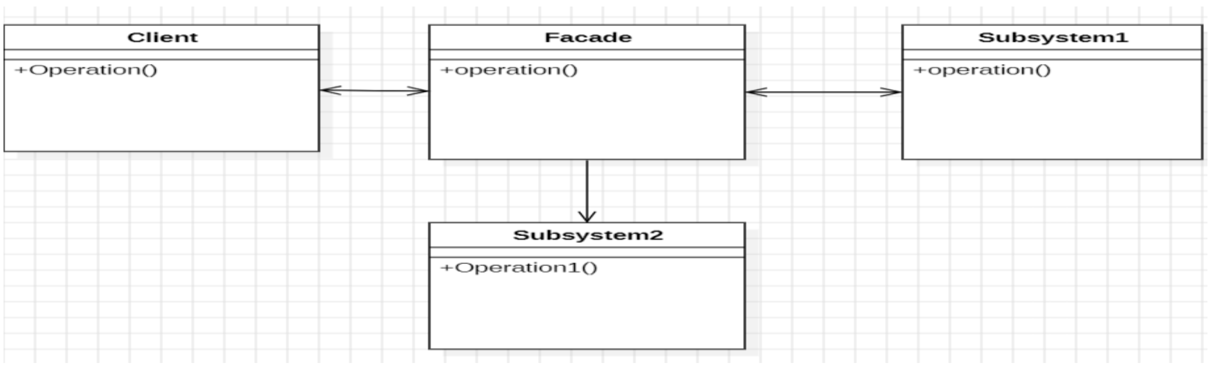


Façade Pattern:

Context/Problem: The system needs to simplify interaction with multiple subsystems (e.g., application processing, document verification, notification).

Solution: Implementing the Facade pattern to provide a unified interface to these subsystems.

Visual:



Result:

Thus the reusability and maintainability of the software system by applying appropriate design patterns was improved successfully.