TABLE OF CONTENT

EXP.NO: 1 **DATE: 31|01|2025**

STUDY OF STAR UML

AIM:

INTRODUCTION:

Star UML is an open-source software modeling tool that supports the UML (Unified Modeling Language) framework for system and software modeling. It is based on UML version 1.4, provides eleven different types of diagrams, and accepts UML 2.0 notation, It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing togenerate code for multiple languages.

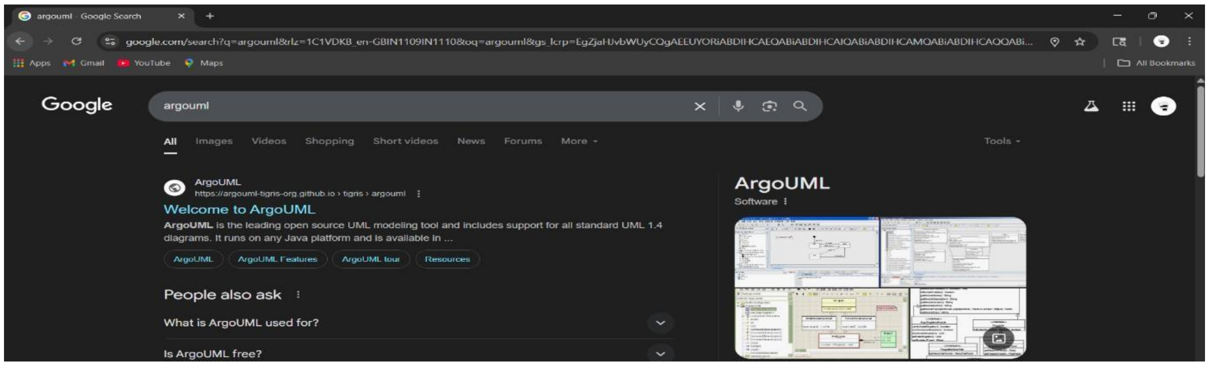
SYSTEM REQUIREMENTS:

Windows 2000, Windows XP, or higher; Microsoft Internet Explorer 5.0 or higher; 128 MB RAM(256MB recommended); 110MB hard disc space (150MB space recommended).

INSTALLATION: The installer follows the classic Windows install procedure without issues.

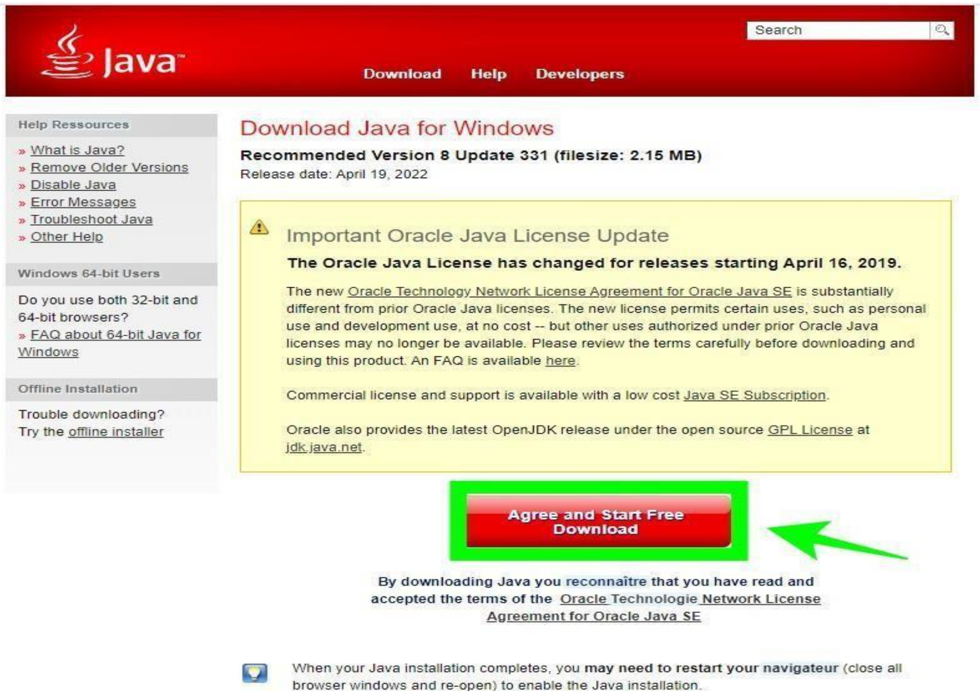
STEPS FOR DOWNLOADING STAR UML

STEP 1: Search Argo UML in google


argouml - Google Search
google.com/search?q=argouml&rlz=1C1VDKB_en-GBIN1109IN1110&oq=argouml&gs_lcrp=EgZjaHJvbWUyCQgAEEUYORiABDIHCAEQABiABDIHCAIQABiABDIHCAMQABiABDIHCAQQABi...
Apps Gmail YouTube Maps All Bookmarks
Google
argouml
All Images Videos Shopping Short videos News Forums More Tools
ArgoUML Features
ArgoUML tour
Resources
ArgoUML
Software

ArgoUML
https://argouml-tigris-org.github.io › tigris › argouml
Welcome to ArgoUML
ArgoUML is the leading open source UML modeling tool and includes support for all standard UML 1.4 diagrams. It runs on any Java platform and is available in ...

People also ask
What is ArgoUML used for?
Is ArgoUML free?



Help Ressources

- What is Java?
- Remove Older Versions
- Disable Java
- Error Messages
- Troubleshoot Java
- Other Help

Windows 64-bit Users

Do you use both 32-bit and 64-bit browsers?
» FAQ about 64-bit Java for Windows

Offline Installation

Trouble downloading?
Try the offline installer

Download Java for Windows

Recommended Version 8 Update 331 (filesize: 2.15 MB)
Release date: April 19, 2022

Important Oracle Java License Update

The Oracle Java License has changed for releases starting April 16, 2019.

The new Oracle Technology Network License Agreement for Oracle Java SE is substantially different from prior Oracle Java licenses. The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle Java licenses may no longer be available. Please review the terms carefully before downloading and using this product. An FAQ is available here.

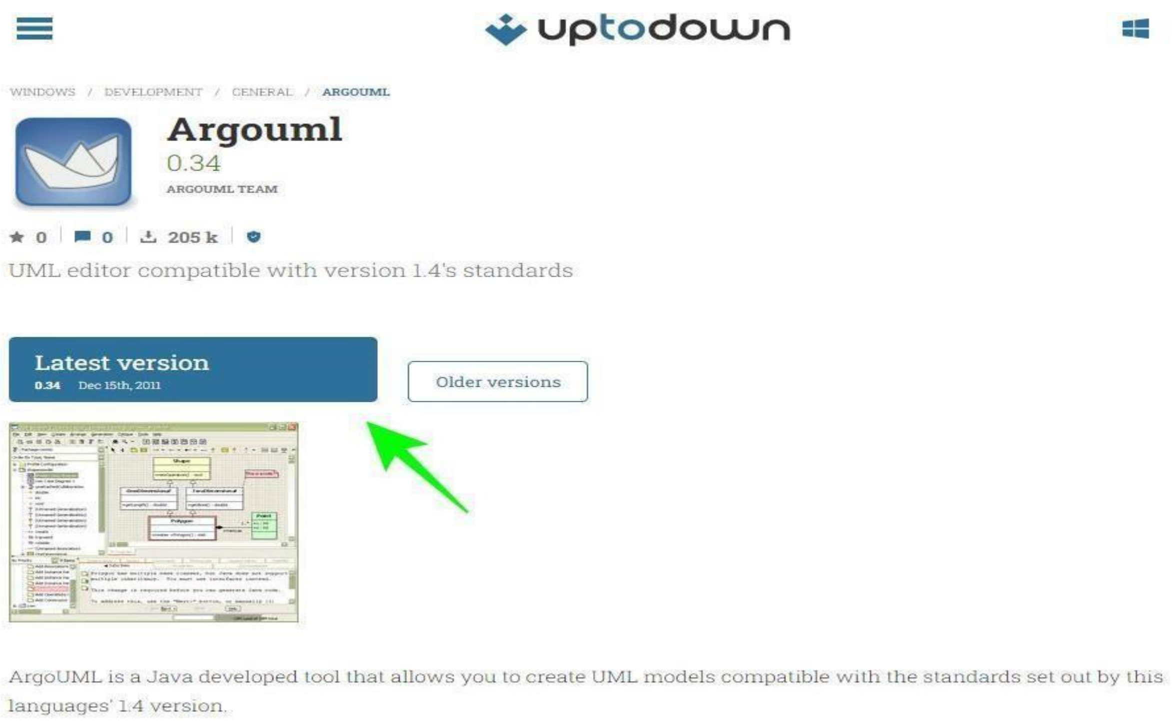
Commercial license and support is available with a low cost Java SE Subscription.

Oracle also provides the latest OpenJDK release under the open source GPL License at jdk.java.net.

By downloading Java you reconnaître that you have read and accepted the terms of the Oracle Technologie Network License Agreement for Oracle Java SE

When your Java installation completes, you may need to restart your navigateur (close all browser windows and re-open) to enable the Java installation.

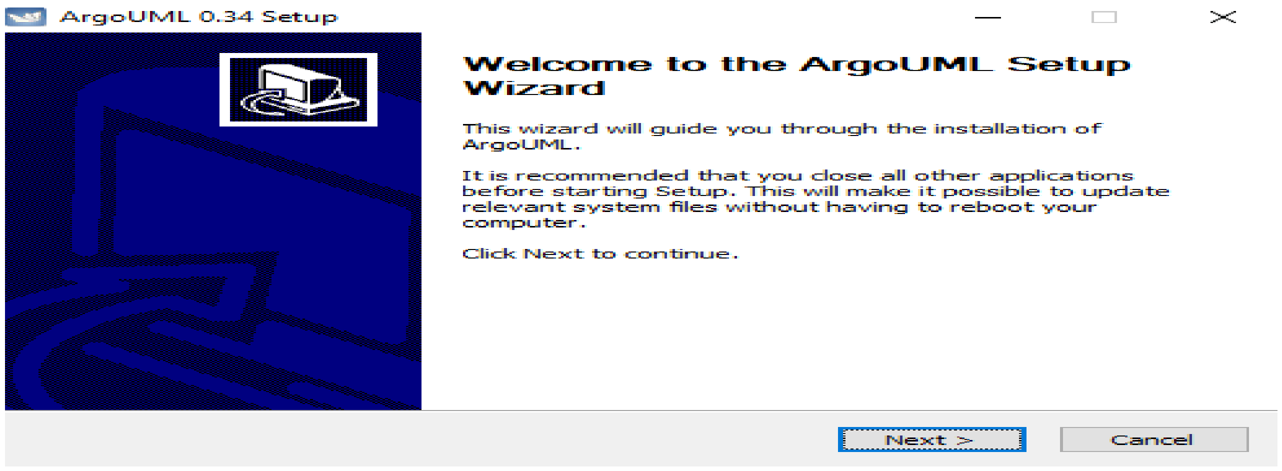
STEP 2 : Download for Mac OS/Windows/.deb, .rpm



UML editor compatible with version 1.4's standards

ArgoUML is a Java developed tool that allows you to create UML models compatible with the standards set out by this languages' 1.4 version.

STEP 3 : Install it



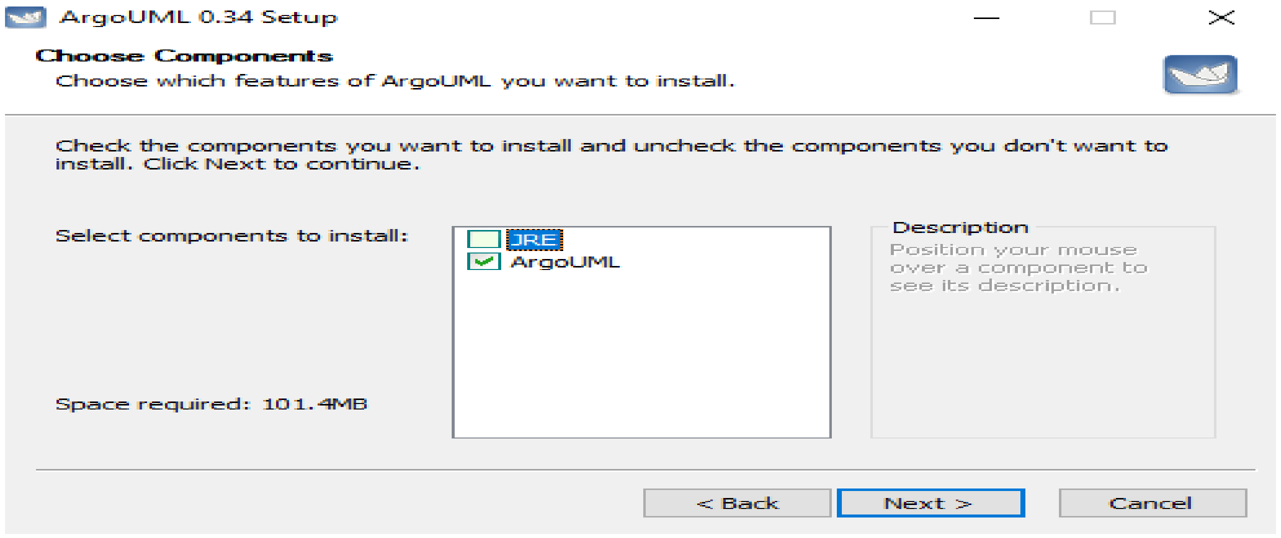
ArgoUML 0.34 Setup

Welcome to the ArgoUML Setup Wizard

This wizard will guide you through the installation of ArgoUML.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



ArgoUML 0.34 Setup

Choose Components

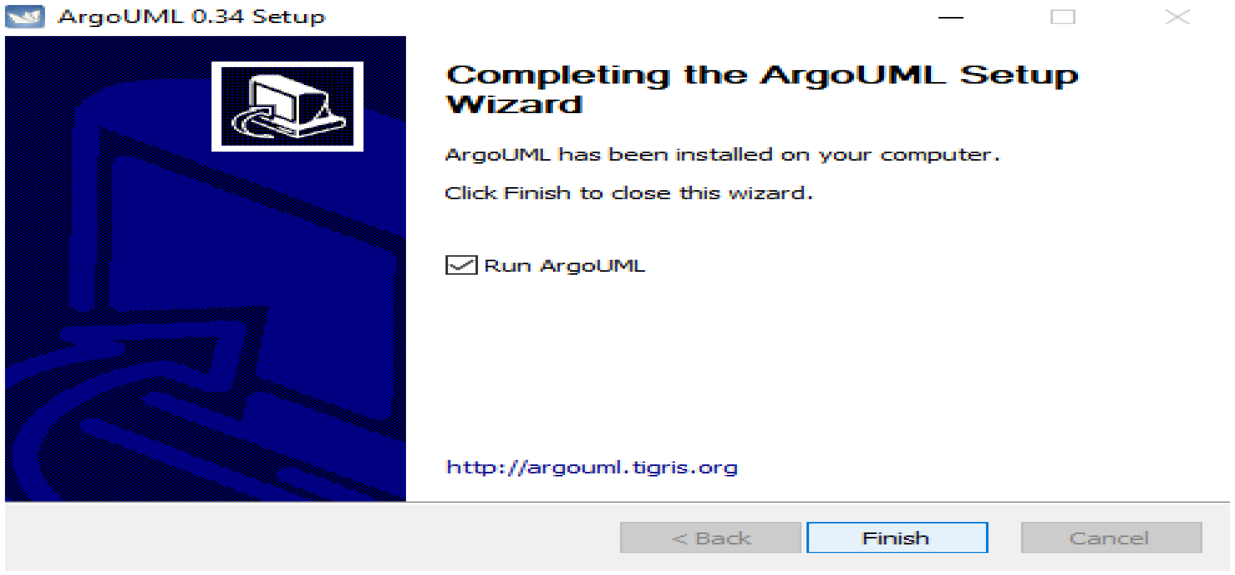
Choose which features of ArgoUML you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [ ] JRE
- [✓] ArgoUML

Space required: 101.4MB

Description
Position your mouse over a component to see its description.



ArgoUML 0.34 Setup

Completing the ArgoUML Setup Wizard

ArgoUML has been installed on your computer.

Click Finish to close this wizard.

[✓] Run ArgoUML

http://argouml.tigris.org

DOCUMENTATION:

The same help that could be browsed on the StarUML website is available with the tool on your desktop Documentation describes the concepts of the tool but on high-level vision. More detailed documentation is available for the diagramming functions. Sample projects are provided with the tooland one of them contains the model of the tool itself, showing that the developers were able to eat their dog food. Besides English, documentation exists in Korean, Japanese, and Russian.

CONFIGURATION:

Some general and diagram configuration options are available from the Tools/Options menu. You willfind in this window also the configuration switches for the code generation. The interface is also very configurable as you can select what part of the tool you would like to view or not.

FEATURES:

When you start a new project, StarUML proposes which approach you want to use:4+1 (Kruchten).Rational, UML, components (from Cheesman and Daniels book), default or empty. Depending on theapproach, profiles and/or frameworks may be included and loaded. If you don't follow a specific approach, the "empty" choice could be used.

Although a project can be managed as one file, it maybeconvenient to divide it into many units and manage them separately if many developers are working on it together. StarUML. makes a clear conceptual distinction between models, views, and diagrams. A Model isanelement that contains information for a software model. A View is a visual expression of the information contained in a model, and a Diagram is a collection of view elements that represent theuser's specific design thoughts.

StarUML is built as a modular and open tool. It provides frameworks for extending the functionality of the tool. It is designed to allow access to all functions of the model/meta-model and tool through COM Automation, and it provides an extension of menu and option items.

Also,users can create theirown approaches and frameworks according to their methodologies. The toolcan also be integrated with any external tools. StarUML supports the following Diagram types:

- Use Case Diagram
- Class Diagram
- Sequence Diagram Collaboration Diagram
- State chart Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- Composite Structure Diagram

RESULT:

EXP.NO: 2 **DATE: 07|02|2025**

STUDENT RESULT MANAGEMENT SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

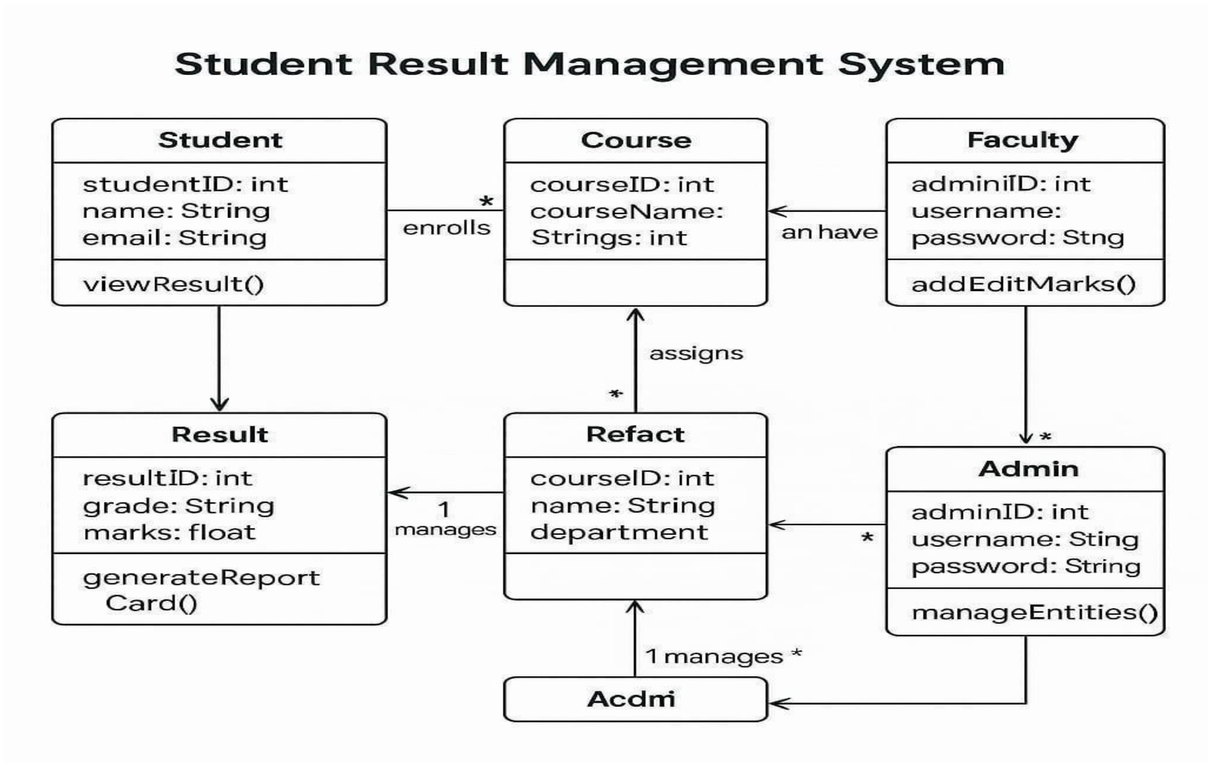
INTRODUCTION:

OBJECTIVES:

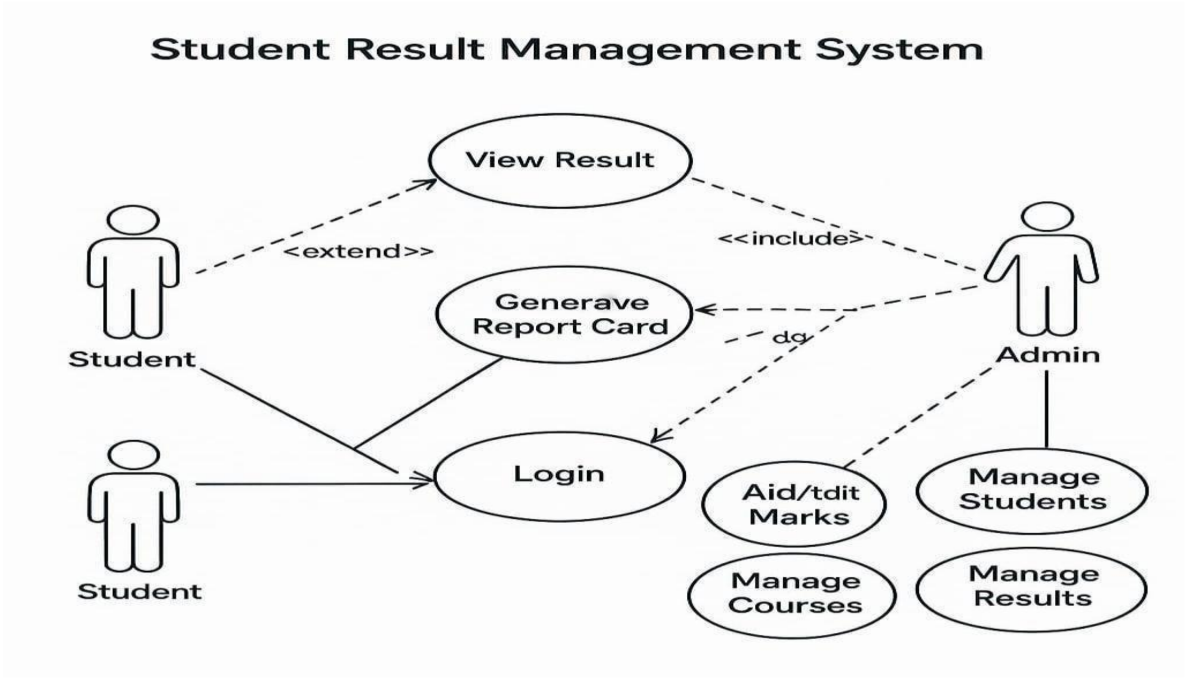
SCOPE:

PROBLEM STATEMENT:

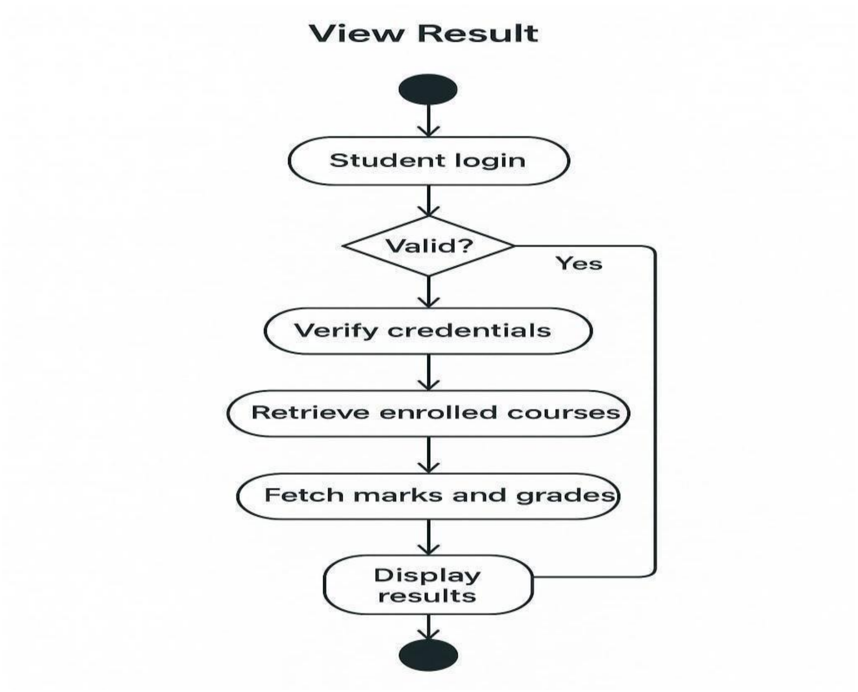
Class Diagram:



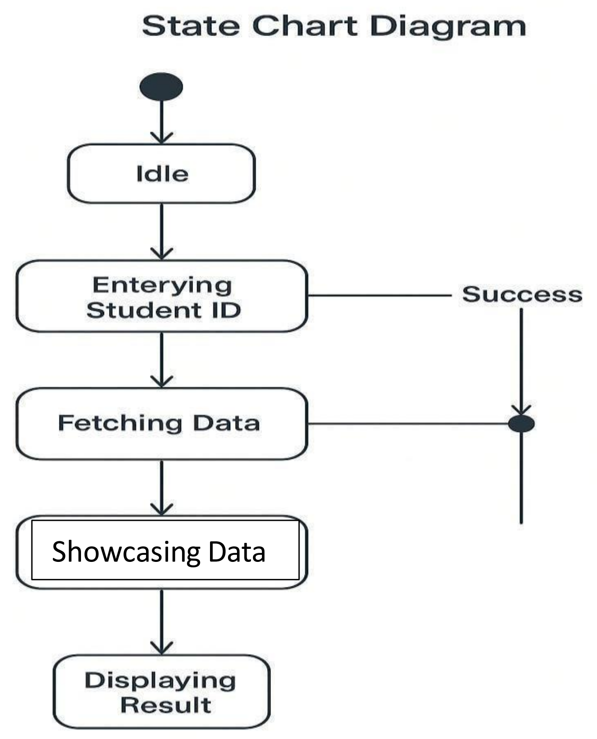
Use-case Diagram:



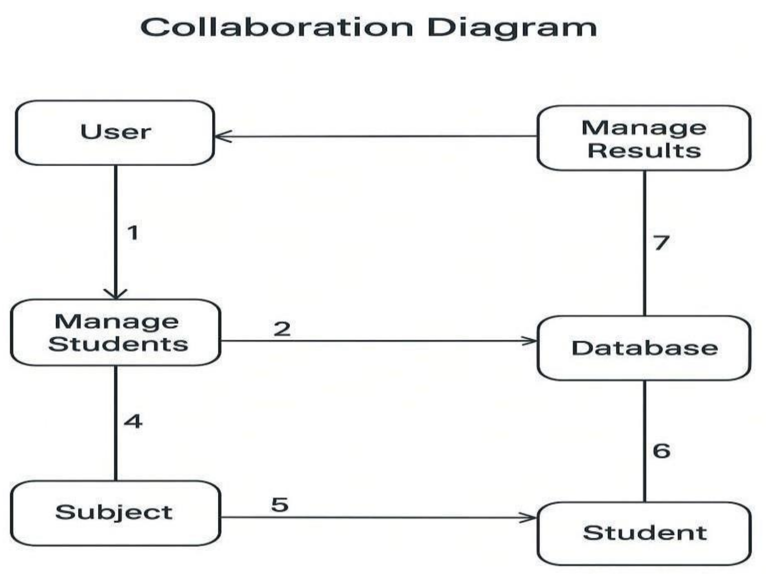
Activity Diagram:



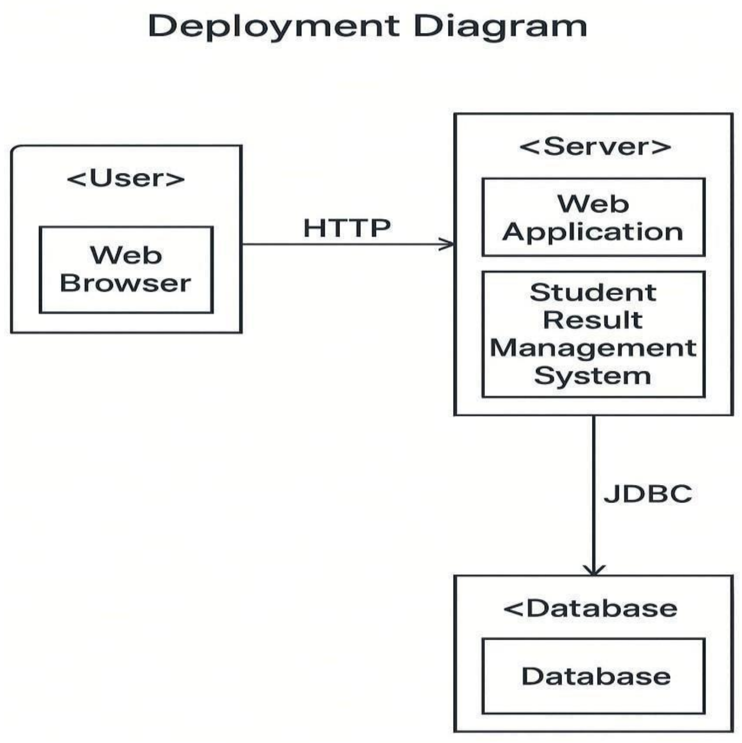
State chart diagram:



Collaboration Diagram:



Deployment Diagram:



STEPS TO GENERATE CODE IN STAR UML:

```
}
```

OUTPUT:

Student.java

```
public class Student {
public int studentID;
public String name;
public String email;
public void viewResult() {
   }
}
```

Course.java

```
public class Course {
public int courseID;
public int strings;
public String courseName;
}
```

Faculty.java

```
public class Faculty {
public int adminID;
public String username;
public String password;
public void addEditMarks() {
   }
}
```

Result.java

```
public class Result {
public  int  resultID;
public String grade;
public float marks;
public void generateReportCard() {
   }
}
```

Admin.java

```
public class Admin {
public int adminID;
public String username;
public String password;
public void manageEntities() {
   }
}
```

Refact.java

```
public class Refact {
public int courseID;
public String name;
public String department;
}
```

RESULT:

EXP.NO: 3 **DATE: 14|02|2025**

INVENTORY CONTROL SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

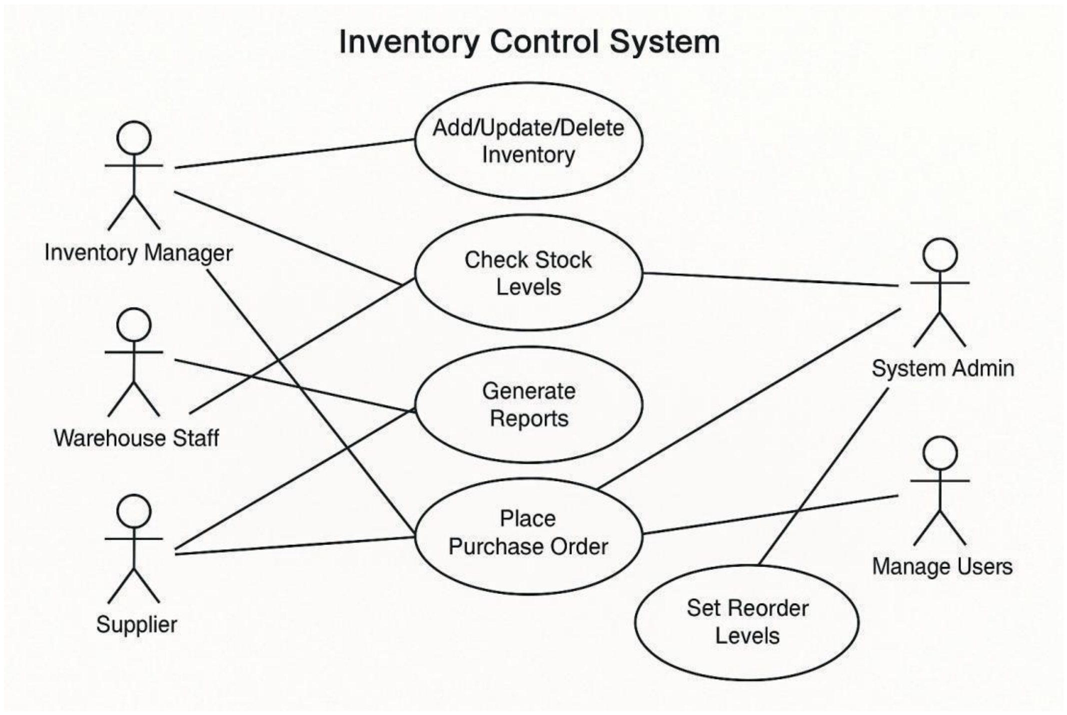
INTRODUCTION:

OBJECTIVES:

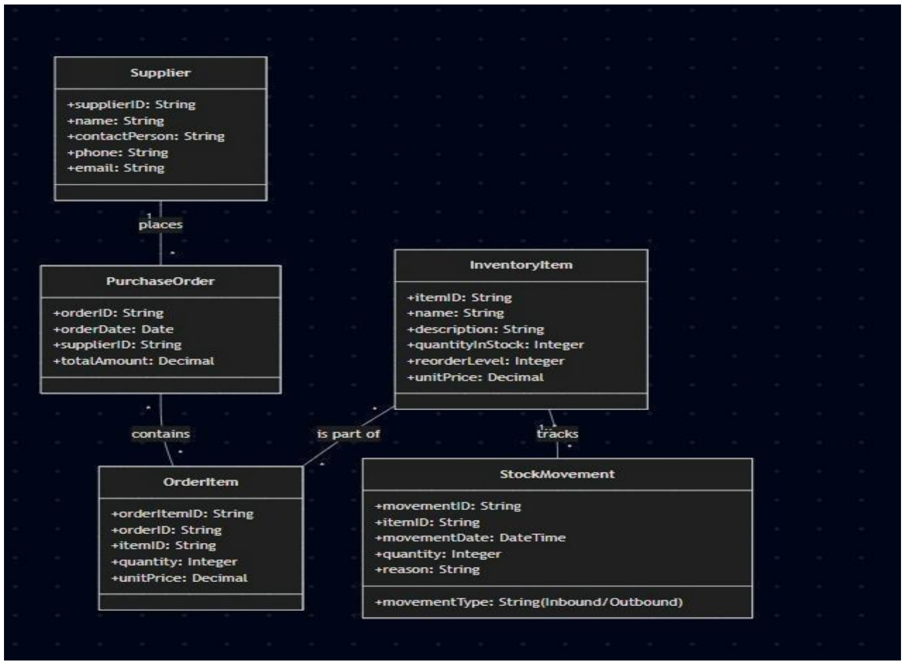
SCOPE:

PROBLEM STATEMENT:

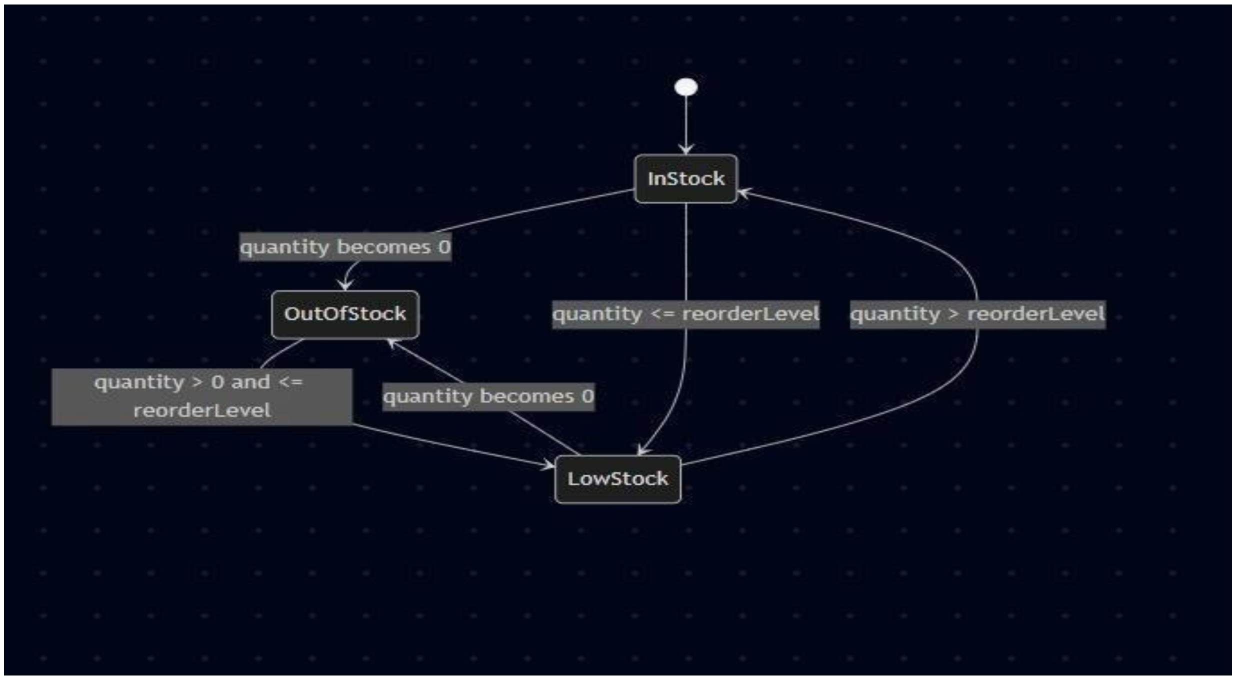
Use Case Diagram:



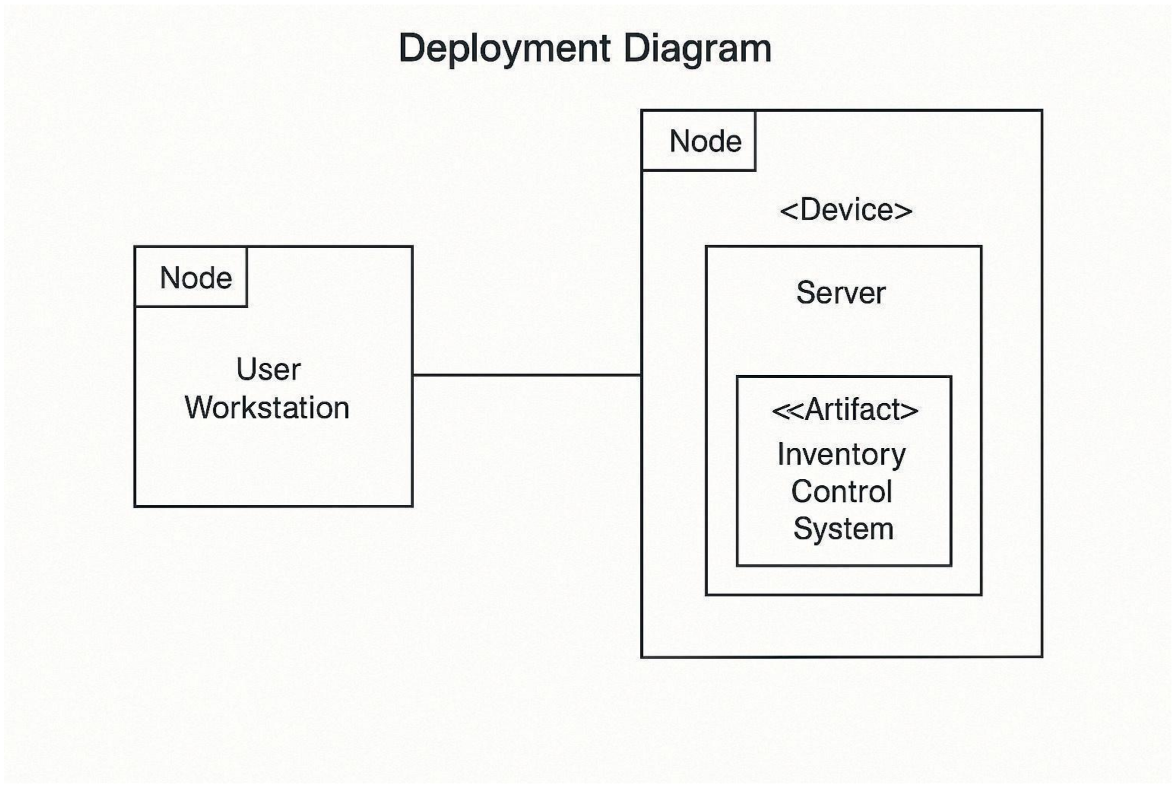
Class Diagram:



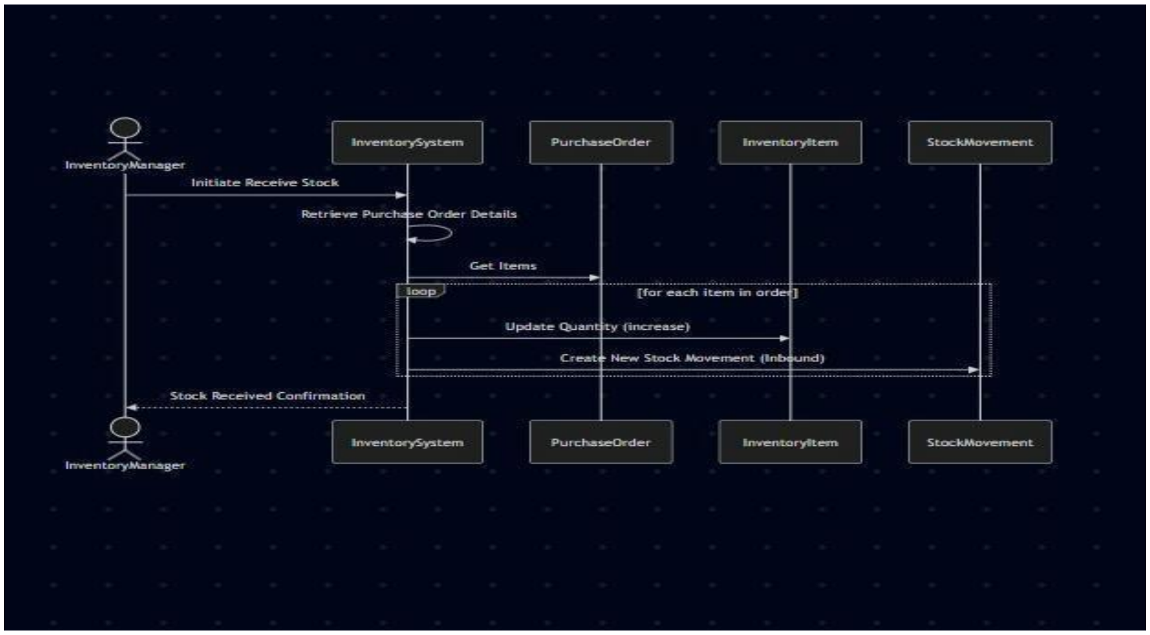
State Chart Diagram:



Deployment Diagram:



Sequence Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Item.java

```
public class Item {

public int itemID;

public String name;

public int quantity;

public double price;

}
```

Supplier.Java

```
public class Supplier {

public int supplierID;

public String name;

public String contactInfo;

}
```

InventoryManagement.java

```
public class InventoryManager {

public int managerID;

public String name;

public String loginCredentials;

}
```

Order.java

```
import java.util.Date;

public class Order {

public int orderID;

public Date date;

public String status;

}
```

Purchase.java

```
import  java.util.Date;

public class Purchase {

public int purchaseID;

public Date date;

public double totalAmount;

}
```

RESULT:

EXP.NO: 4 **DATE: 21|02|2025**

RAILWAY RESERVATION SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

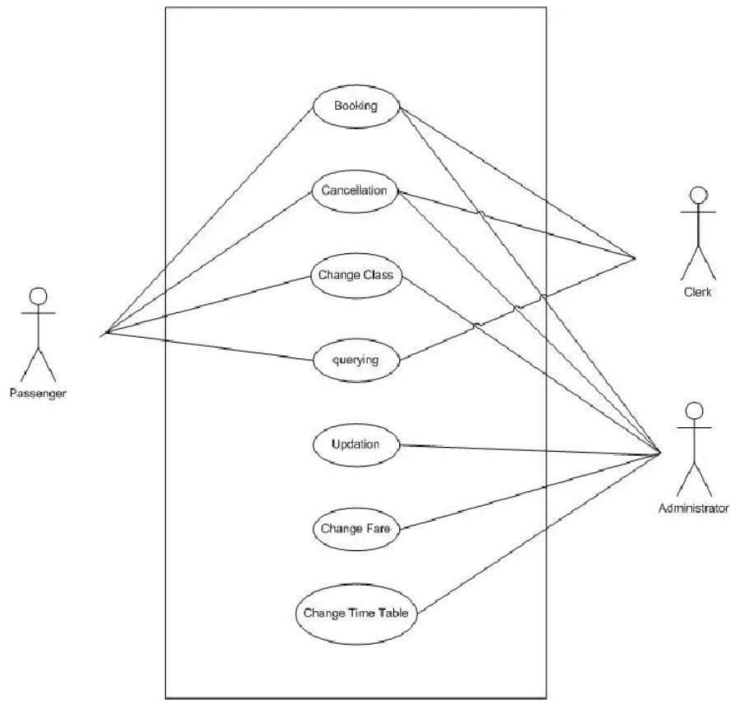
INTRODUCTION:

OBJECTIVES:

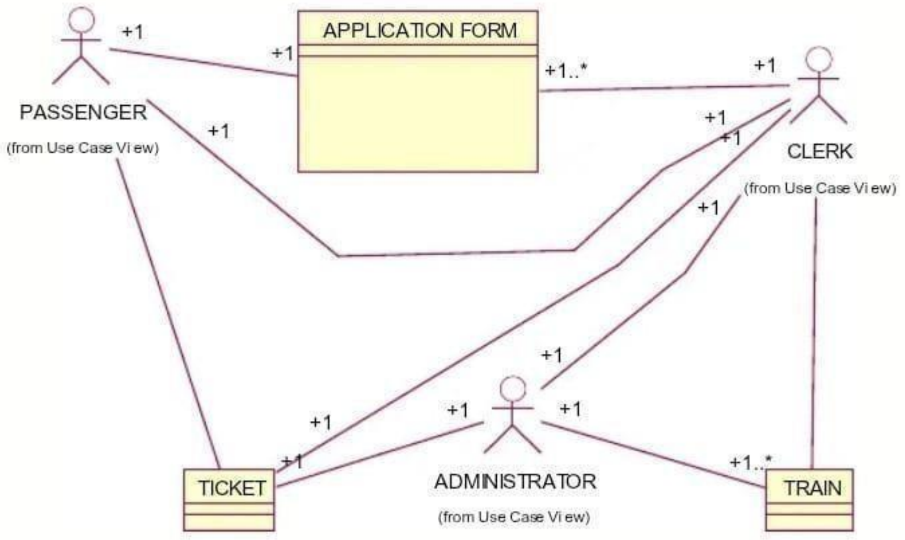
SCOPE:

PROBLEM STATEMENT:

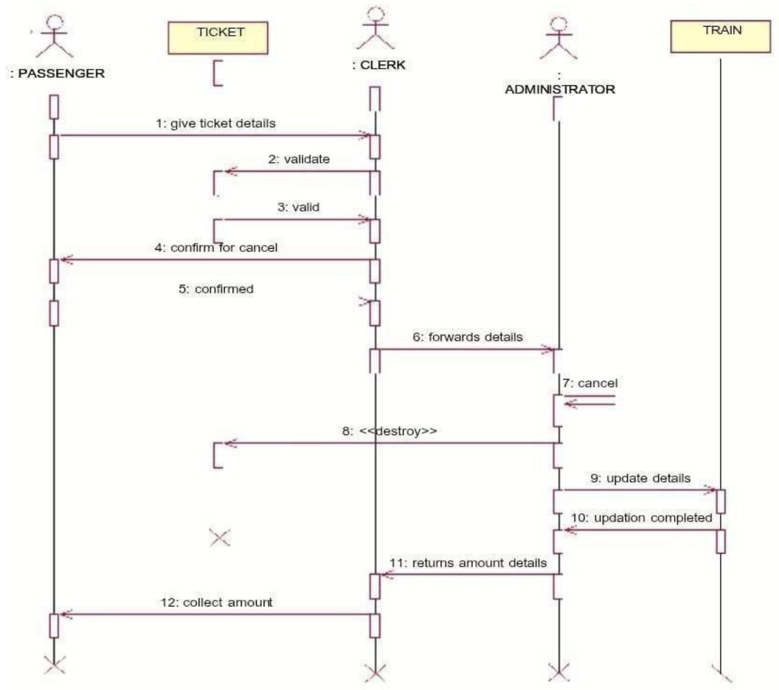
Use Case Diagram:



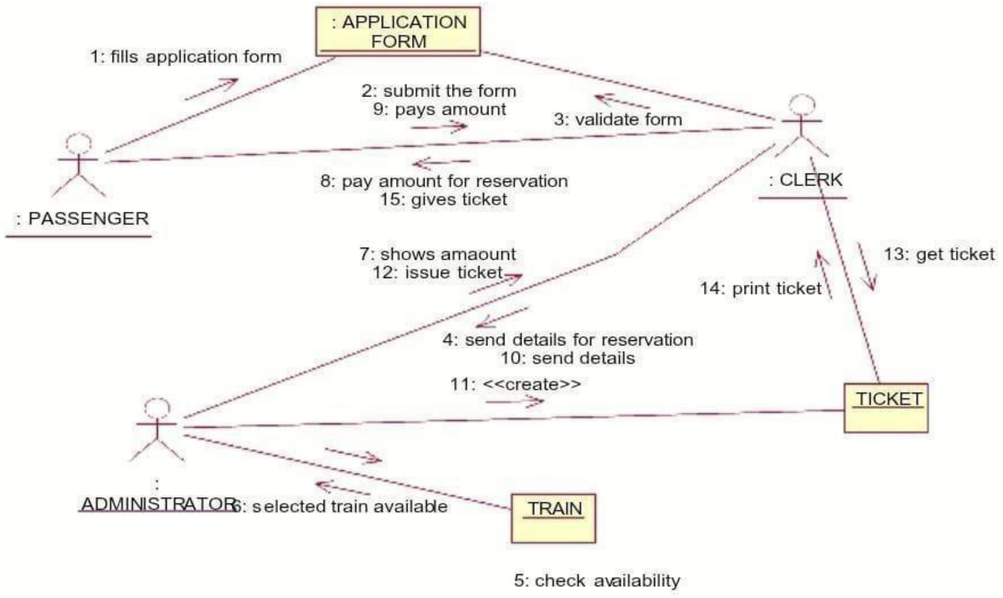
Class Diagram:



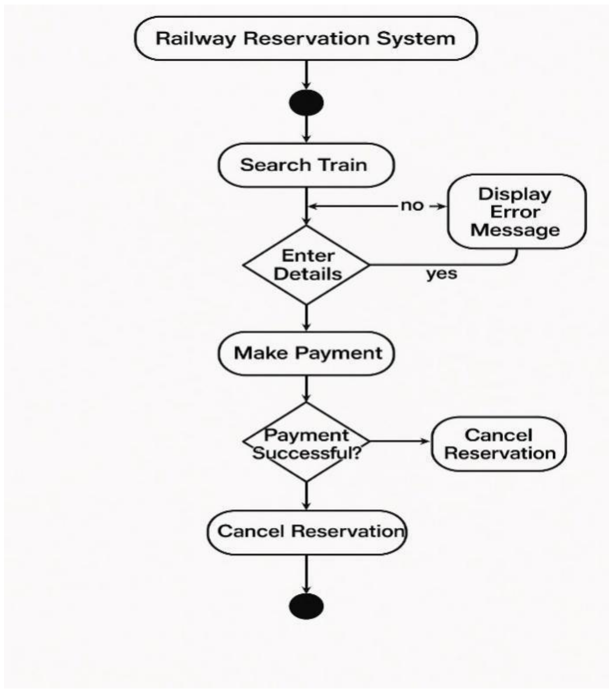
Sequence Diagram:



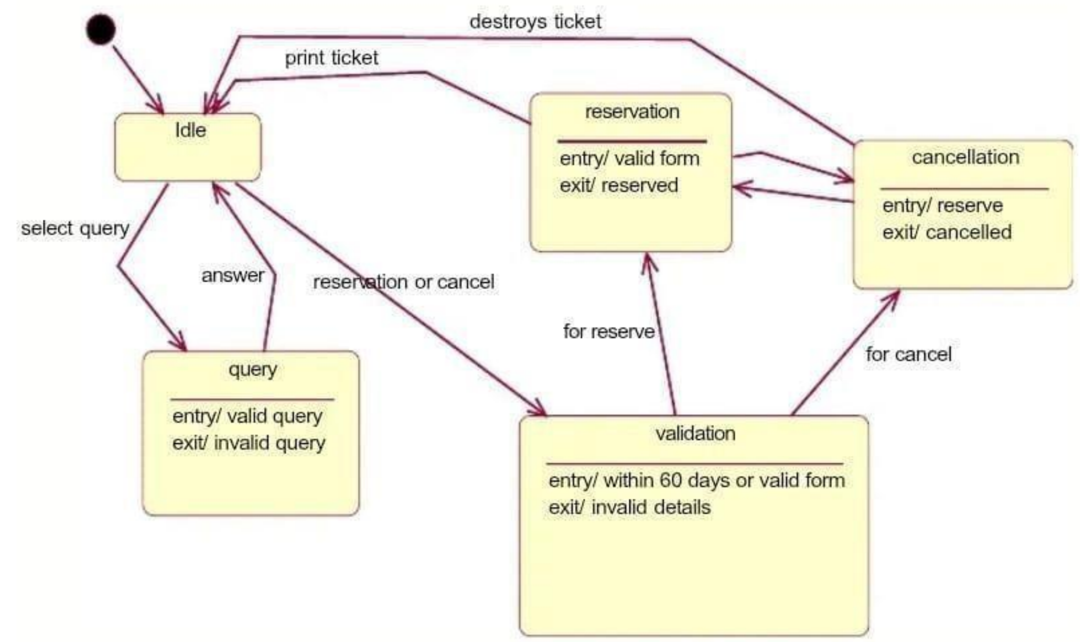
Collaboration Diagram:



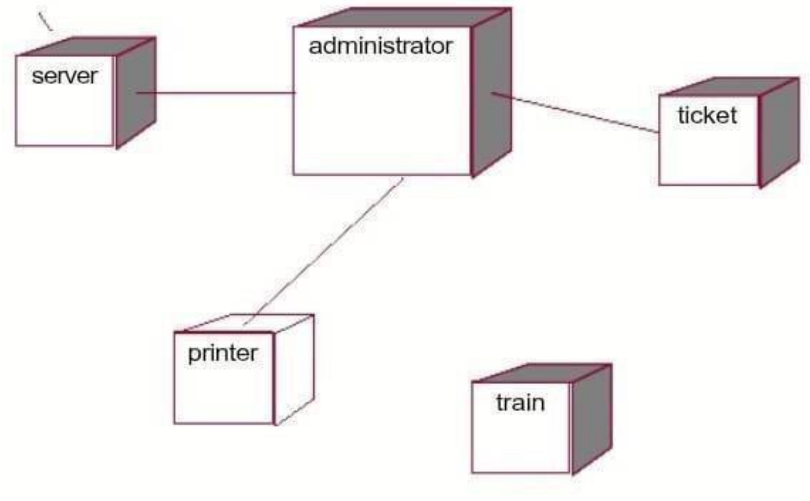
Activity Diagram:



State Chart Diagram:



Deployment Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Train.java

```
public class Train {
public int trainNo;
public String name;
}
```

Clerk.java

```
public class Clerk {
public int ID;
public void verifyDetails() {
   }
public void cancellation() {
   }
}
```

RailwaySystem.java

```
public class RailwaySystem {
public int ID;
public void responses() {
   }
}
```

Payment.java

```
public class Payment {
   public int amount;
}
```

Passenger.java

```
public class Passenger {

public String name;

public String address;

public String gender;

public int date;

public void searchTrain() {

   }

public void bookTrain() {

   }

public void cancelTickets() {

   }

public void payFare() {

   }

}
```

Ticket.java

```
public class Ticket {

public int no;

public String status;

public int noOfPassengers;

public String place;

public void fareAmount() {

   }

public void cancelTickets() {

   }   }
```

RESULT:

EXP.NO: 5 **DATE: 28|02|2025**

HOTEL MANAGEMENT SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

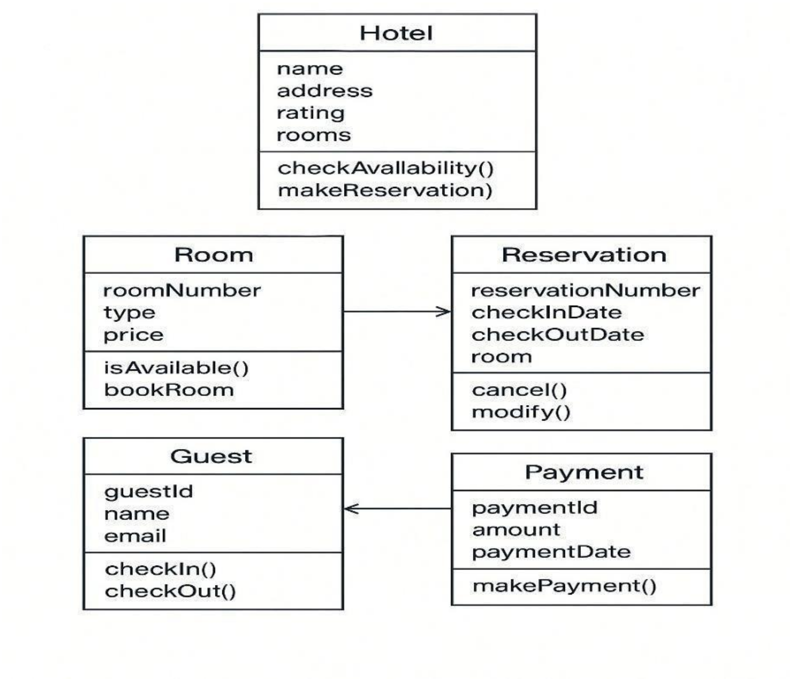
INTRODUCTION:

OBJECTIVES:

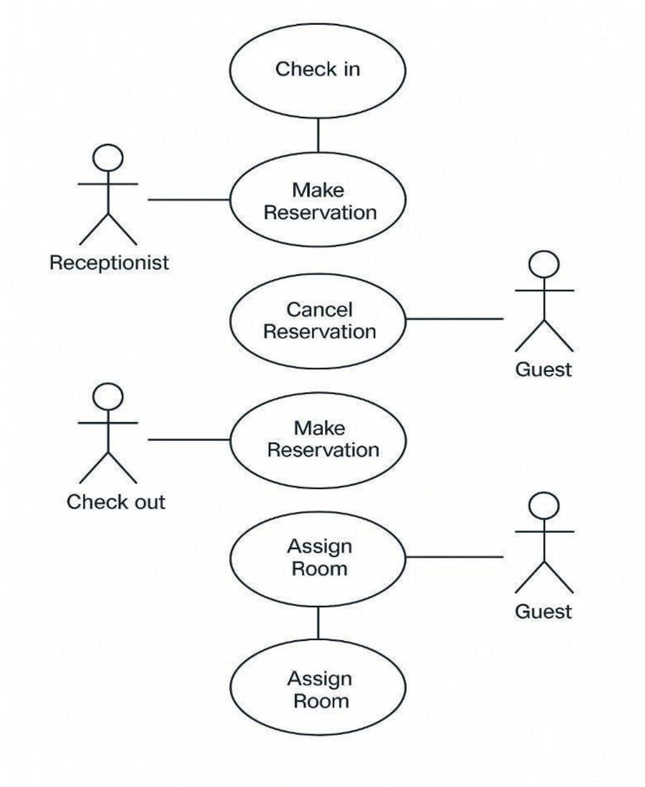
SCOPE:

PROBLEM STATEMENT:

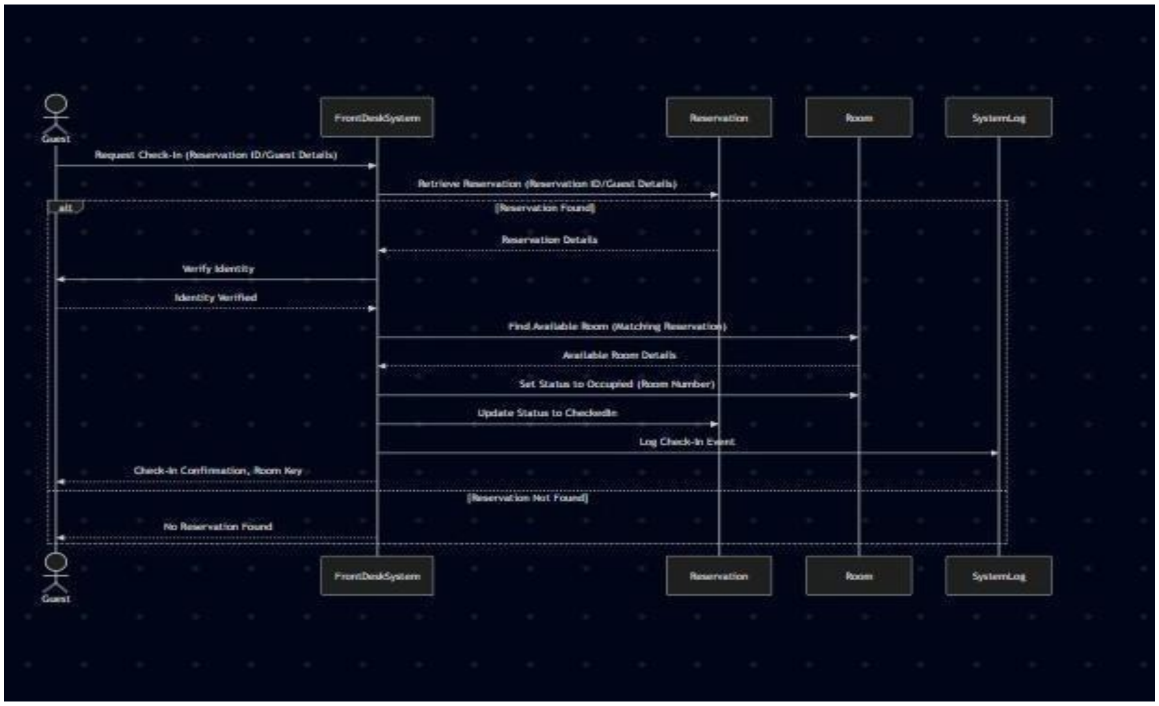
Class Diagram:



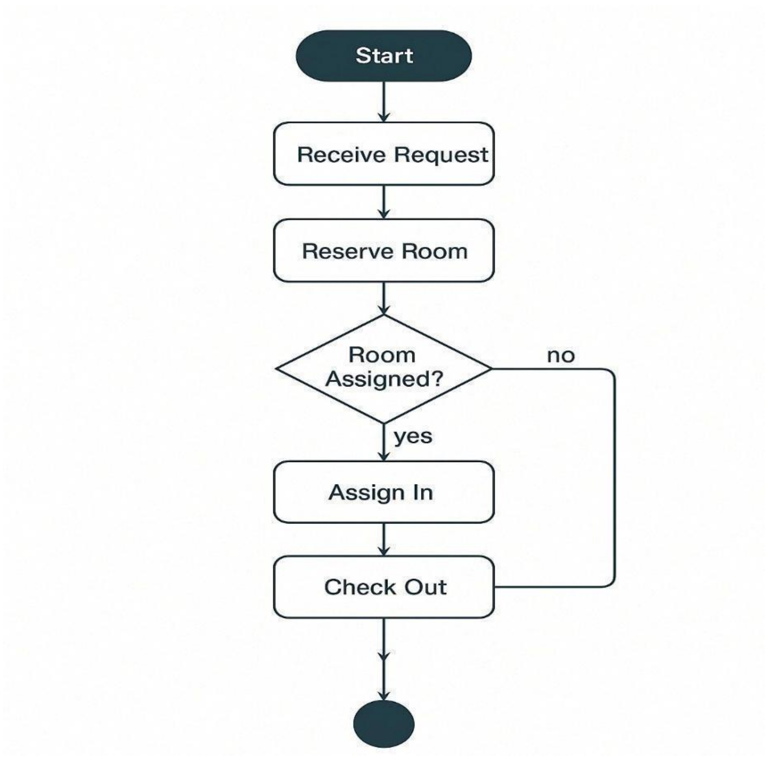
Use Case Diagram:



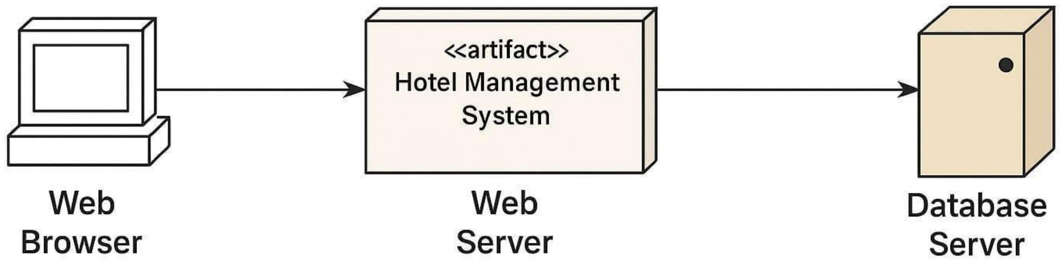
Sequence Diagram:



Activity Diagram:



Deployment Diagram:



Hotel Management System

STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Hotel.java

```
import java.util.List;

public class Hotel {

public String name;

public String location;

public void addRoom() {

   }

public void registerCustomer() {

   }

public void makeBooking() {

   }

}
```

Room.java

```
public class Room {

public int roomNumber;

public String type;

public boolean isAvailable;

public double price;

public void checkAvailability() {

   }

public void updateStatus() {

   }

}
```

Customer.java

```
public class Customer {

public int customerId;

public String name;

public String contact;

public void bookRoom() {

   }

public void cancelBooking() {

   }

}
```

Booking.java

```
import java.util.Date;

public class Booking {

public int bookingId;

public int customerId;

public int roomNumber;

public Date checkInDate;

public Date checkOutDate;

public void confirmBooking() {

   }

public void generateInvoice() {

   }

}
```

Staff.java

```
public class Staff {

public int staffId;

public String name;

public String role;

public void manageRoomService() {

   }

public void assistCustomer() {

   }

}
```

RESULT:

EXP.NO: 6 **DATE: 07|03|2025**

AUTOMATING THE BANKING SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

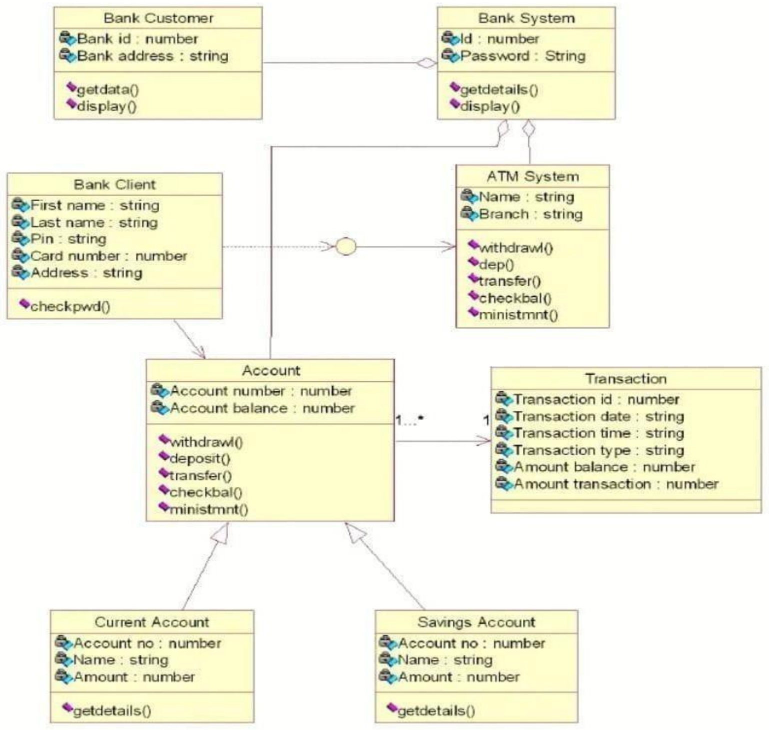
INTRODUCTION:

OBJECTIVES:

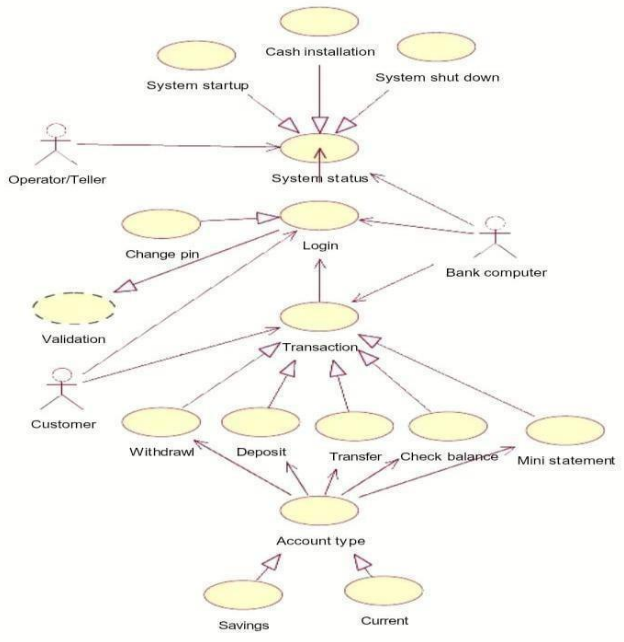
SCOPE:

PROBLEM STATEMENT:

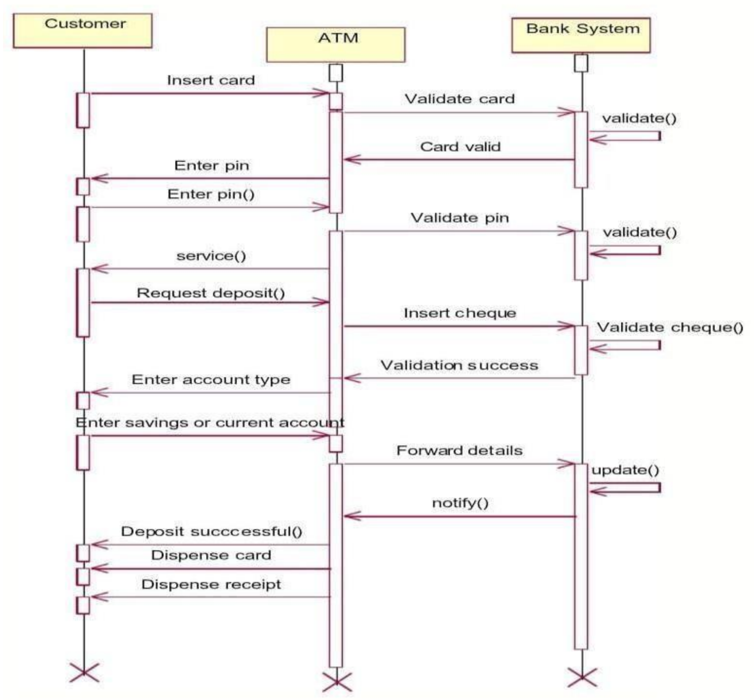
Class Diagram:



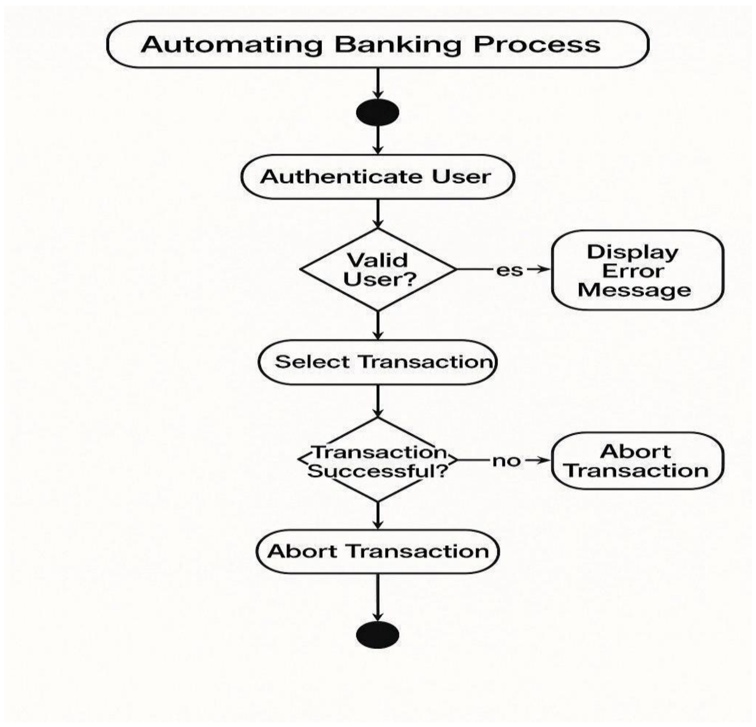
Use case Diagram:



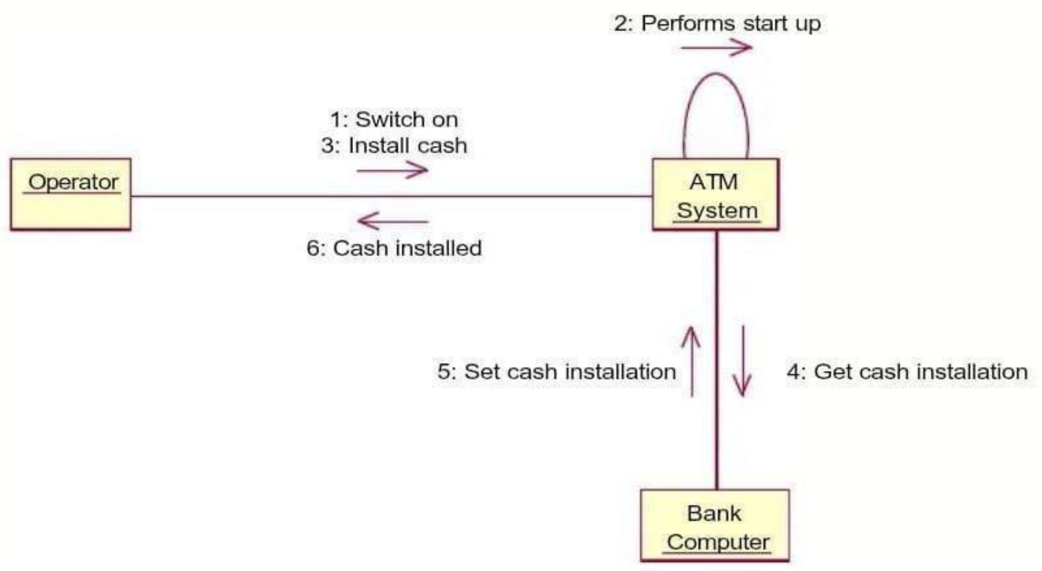
Sequence Diagram:



Activity Diagram:



Collaboration Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

User.java

```
import java.util.Date;

public class User {

public String username;

public String password;

public Date dob;

public int ac_no;

public void read() {

}

public void write() {

}

}
```

ATMmc.java

```
public class ATM_mc {

public int limit;

public int withdraw;

public int tbalance;

public void login_verify() {

   }

public void check_mc() {

   }

public void confirmation() {

   }

}
```

Account.java

```
public class Account {
public int ac_no;
public int balance;
public void withdraw() {
   }
public void deposit() {
   }
public void display_avail() {
   }
}
```

Transaction.java

```
import java.util.Date;
public class Transaction {
public int ac_no;
public Date date;
public String type;
public int amt;
public int balance;
public void ministmt() {
   }
public void display() {
   }
public void update() {
 }
```

RESULT:

EXP.NO: 7 **DATE: 21|03|2025**

LIBRARY MANAGEMENT SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

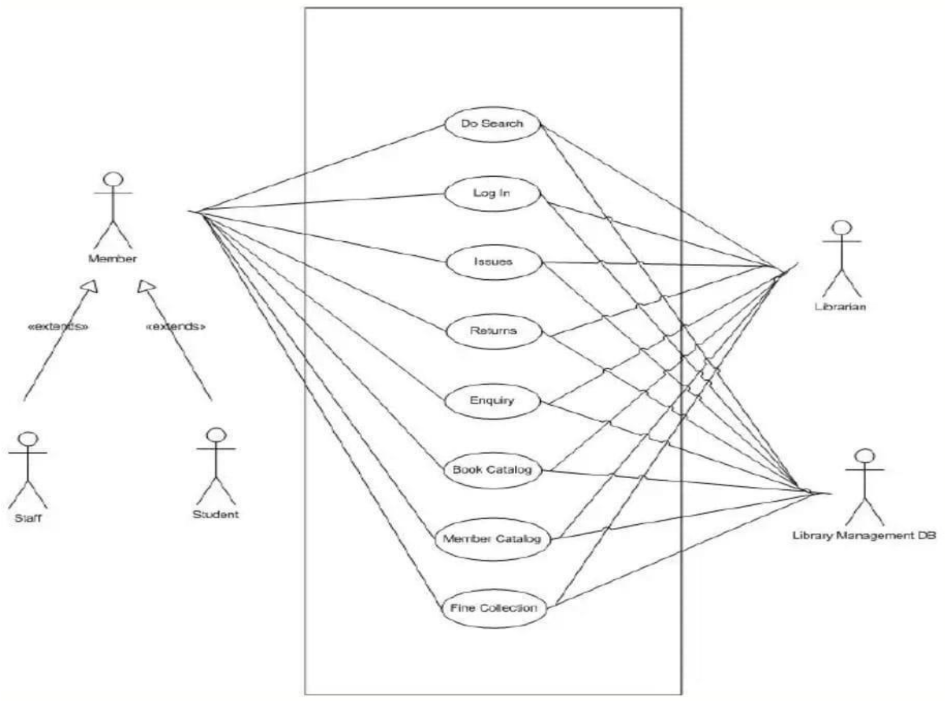
INTRODUCTION:

OBJECTIVES:

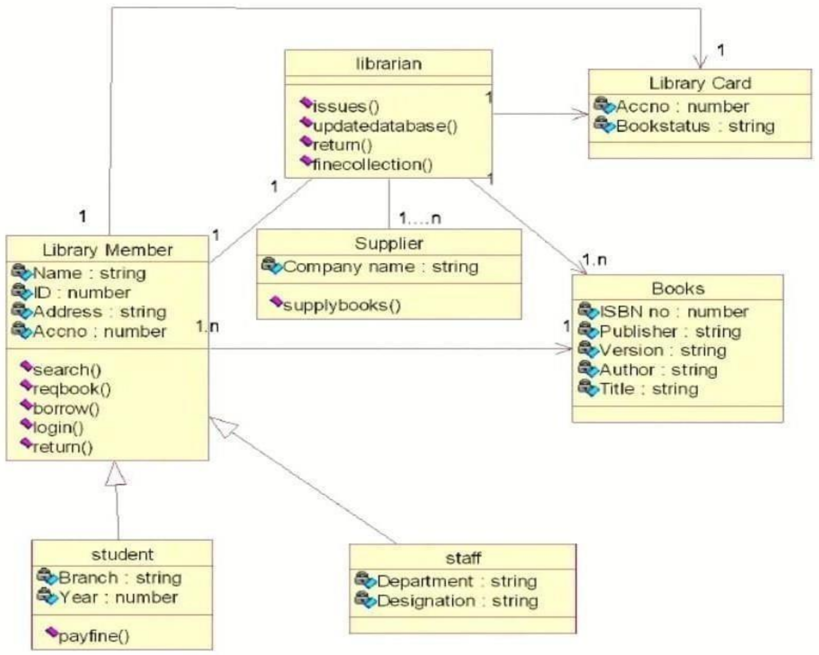
SCOPE:

PROBLEM STATEMENT:

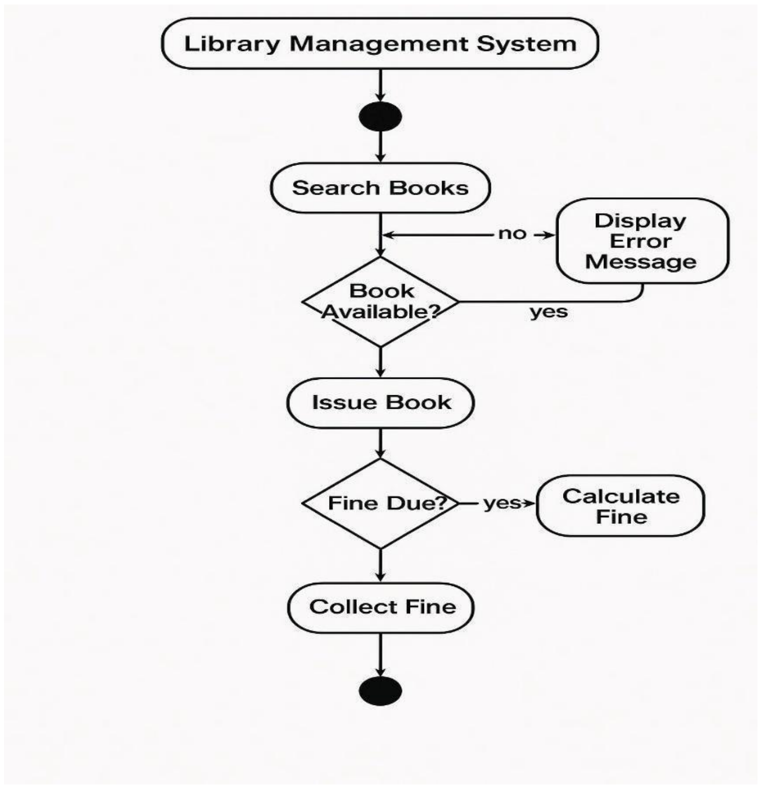
Use case Diagram:



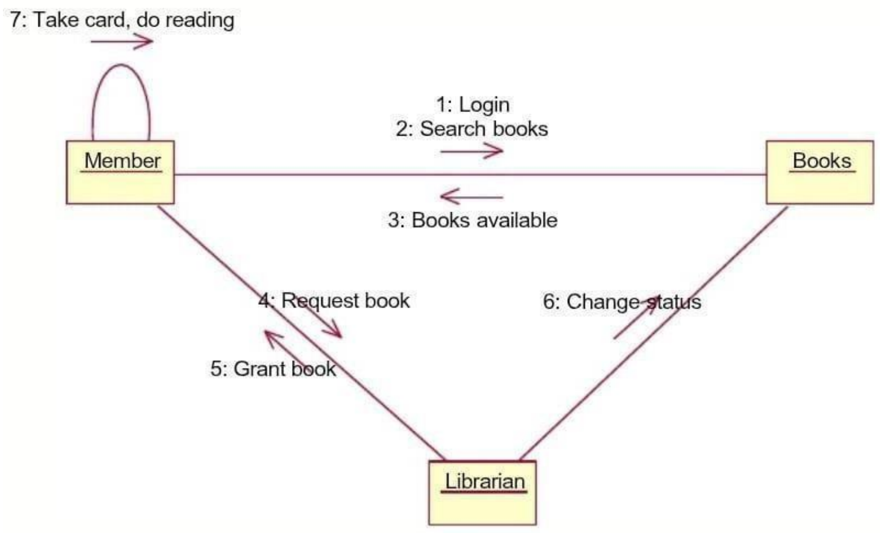
Class Diagram:



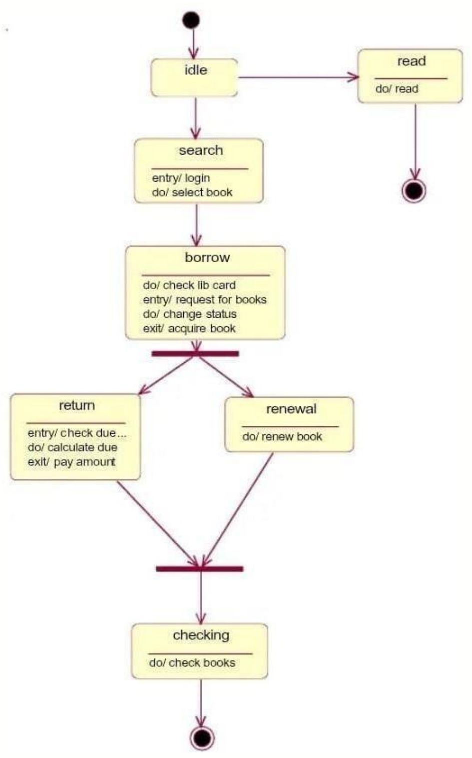
Activity Diagram:



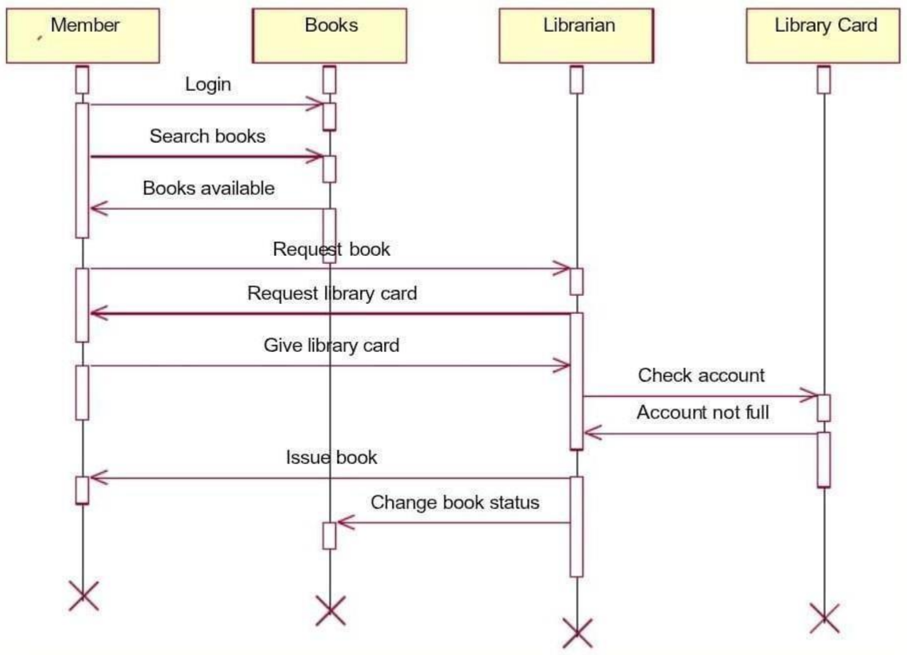
Collaboration Diagram:



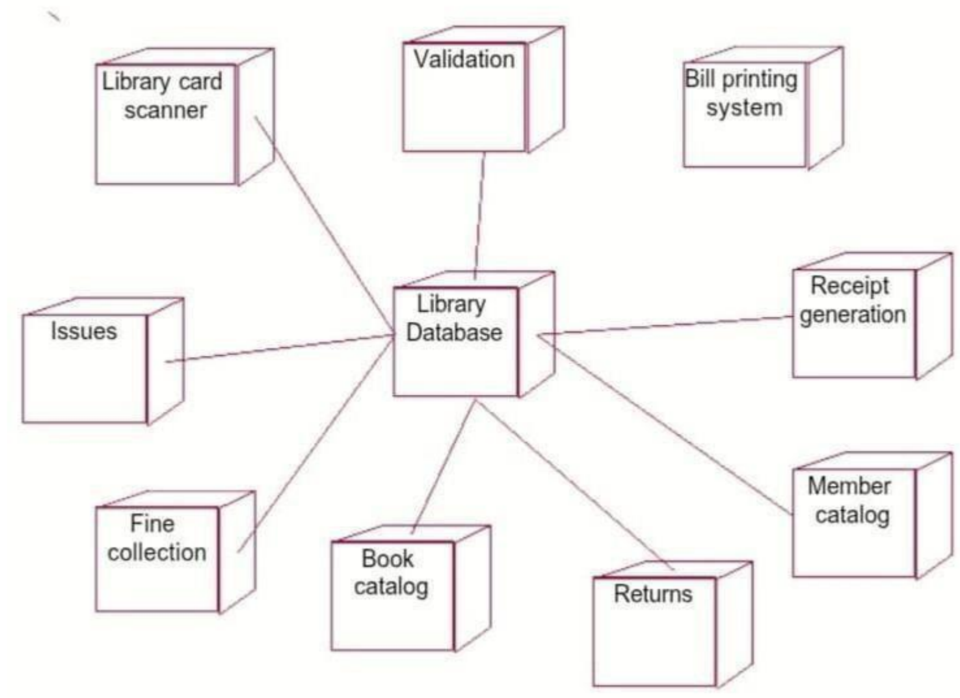
State Chart Diagram:



Sequence Diagram:



Deployment Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Book.java

```
public class Book {

public int bookId;

public String author;

public String name;

public float price;

public String rackNo;

public String status;

public String edition;

public String dateOfPurchase;

public void displayBookDetails() {

   }

public void updateStatus() {

   }

}
```

Journals.java

```
public class Journals extends Book {

}
```

Magazines.java

```
public class Magazines extends Book {

}
```

Librarian.java

```
public class Librarian {
public String name;
public String password;
public void searchBook() {
   }
public void verifyMember() {
   }
public void issueBook() {
   }
public void calculateFine() {
   }
public void createBill() {
   }
public void returnBook() {
   }
}
```

Students.java

```
public class Student extends MemberRecord {
}
```

Faculty.java

```
public class Faculty extends MemberRecord {
}
```

MemberRecord.java

```
public class MemberRecord {
public int memberId;
public String type;
public String dateOfMembership;
public int noOfBooksIssued;
public int maxBookLimit;
public String name;
public String address;
public String phoneNo;
public void retrieveMember() {
   }
public void increaseBookIssued() {
   }
public void decreaseBookIssued() {
   }
public void payBill() {
   }
}
```

Transaction.java

```
public class Transaction {

public int transId;

public int bookId;

public int memberId;

public String dateOfIssue;

public String dueDate;

public void createTransaction() {

   }

public void deleteTransaction() {

   }

public void retrieveTransaction() {

   }

}
```

Bill.java

```
public class Bill {

public int billNo;

public String date;

public int memberId;

public float amount;

public void billCreate() {

   }

public void billUpdate() {

   }

}
```

RESULT:

EXP.NO: 8 **DATE: 28|03|2025**

PASSPORT AUTOMATION SYSTEM

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

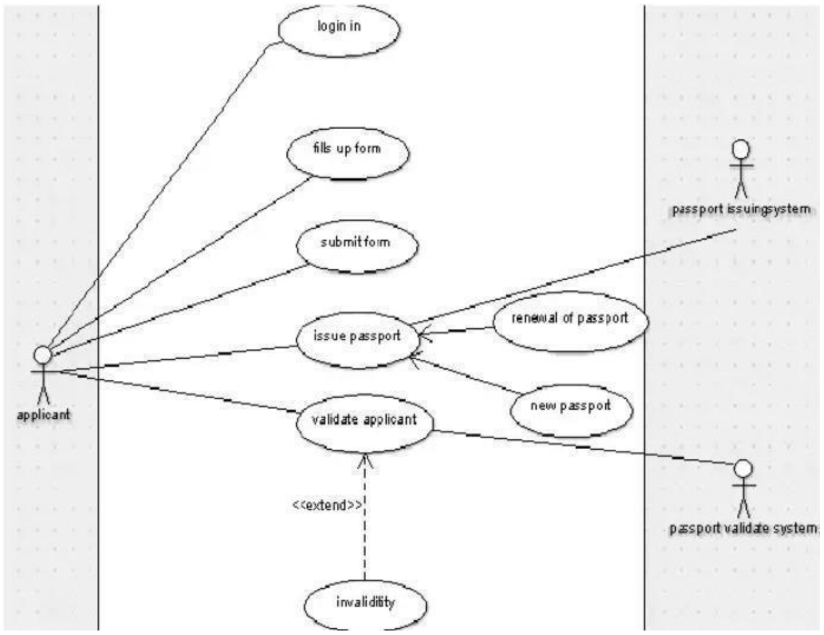
INTRODUCTION:

OBJECTIVES:

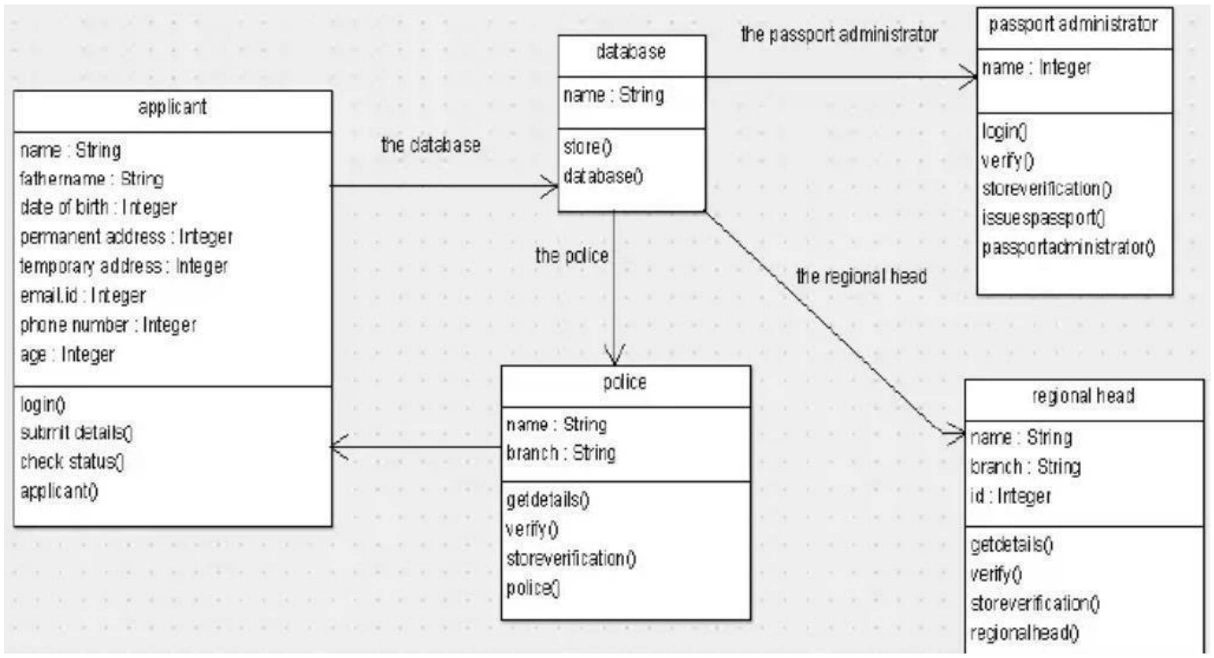
SCOPE:

PROBLEM STATEMENT:

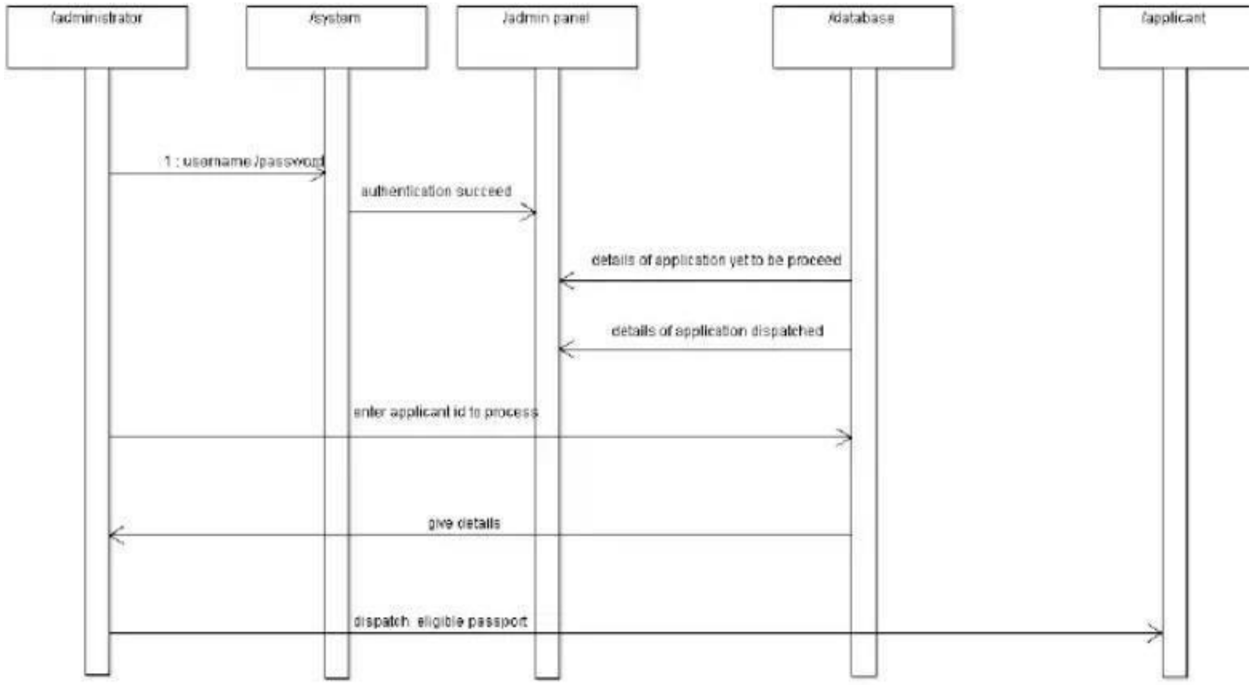
Use-case Diagram:



Class Diagram:

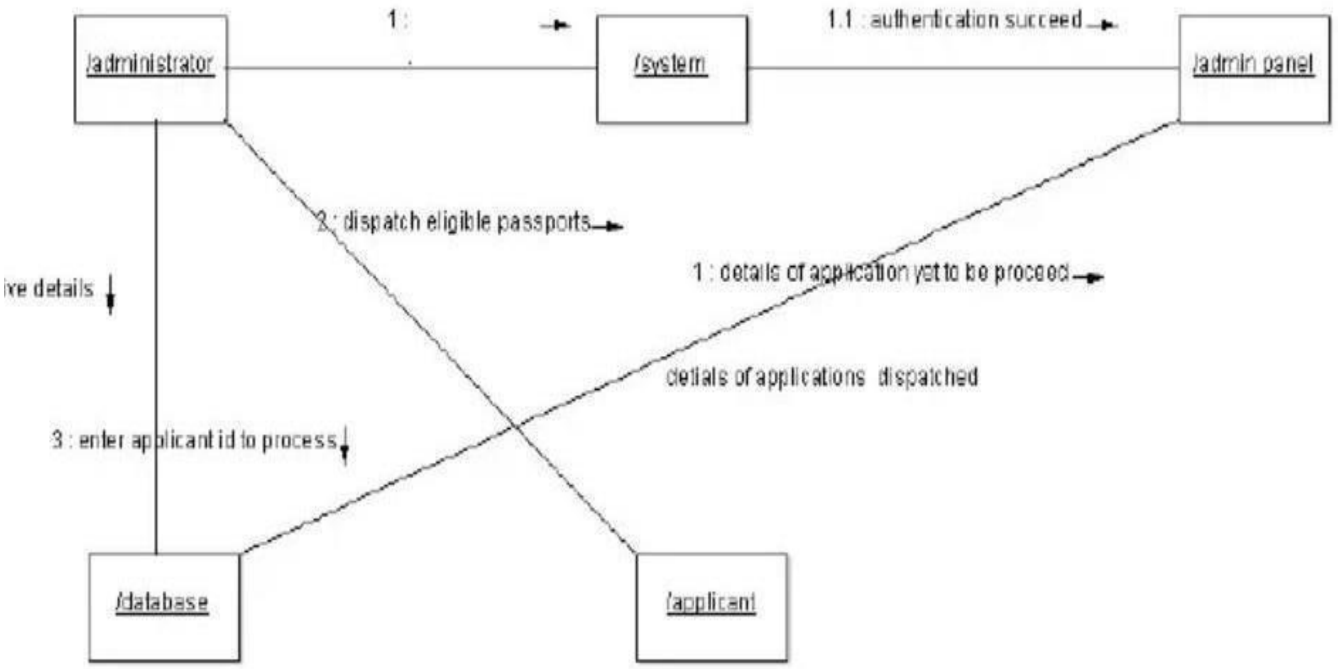


Sequence Diagram:



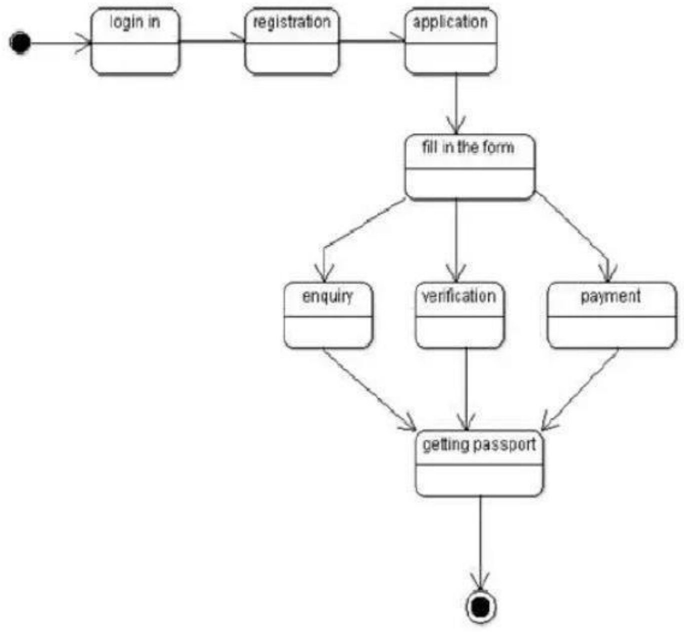
/administrator
/system
/admin panel
/database
/applicant
1 : username /password
authentication succeed
details of application yet to be proceed
details of application dispatched
enter applicant id to process
give details
dispatch eligible passport

Collaboration Diagram:

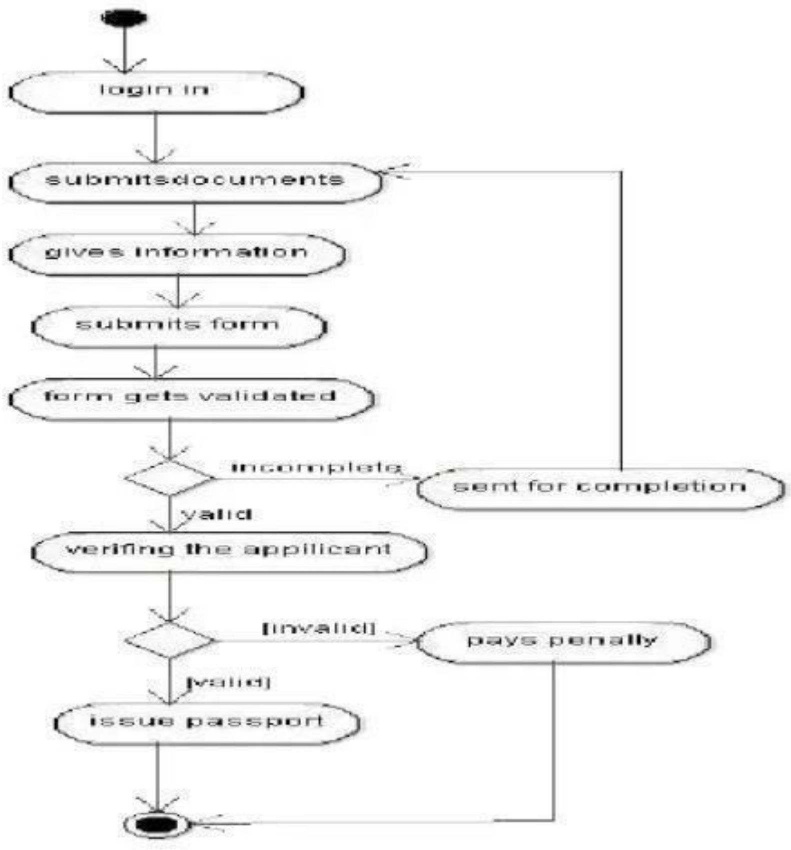


/administrator
/system
/admin panel
/database
/applicant
1 :
1.1 : authentication succeed
2 : dispatch eligible passports
1 : details of application yet to be proceed
detials of applications dispatched
ive details
3 : enter applicant id to process

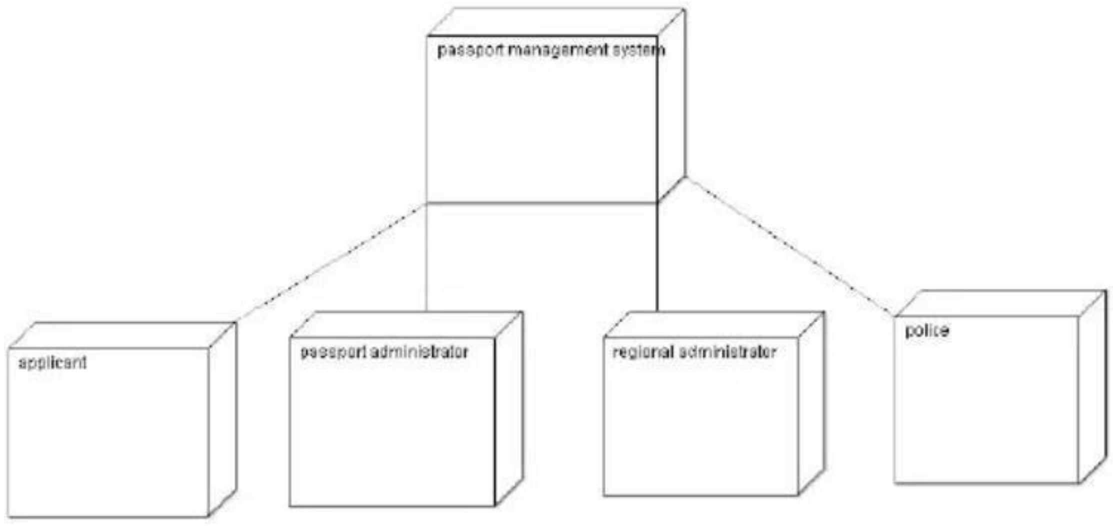
State Chart Diagram:



Activity Diagram:



Deployment Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Applicant.java

```
public class Applicant {
private String name;
private String fatherName;
private String dateOfBirth;
private String permanentAddress;
private String temporaryAddress;
private String emailID;
private String phoneNumber;
private String panNo;
private String applicationNo;
private String userName;
private String password;
public void login() {
}
public void submitDetails() {
}
public void checkStatus() {
}
}
```

Database.java

```
public class Database {
private String name;
public void store() {
  }
}
```

PassportAdministrator.java

```
public class PassportAdministrator {
private String username;
private String password;
public void login() {
    }
public void verify() {
    }
 public void update() {
    }
}
```

RegionalAdministrator.java

```
public class RegionalAdministrator {
private String username;
private String password;
public void login() {
    }
public void verify() {
    }
public void update() {
    }
}
```

Police.java

```
public class Police {
private String username;
private String password;
public void login() {
   }
public void verify() {
   }
public void update() {
   }
}
```

RESULT:

EXP.NO: 9 **DATE: 04|04|2025**

E-TICKETING

AIM:

PROBLEM ANALYSIS AND PROJECT PLANNING

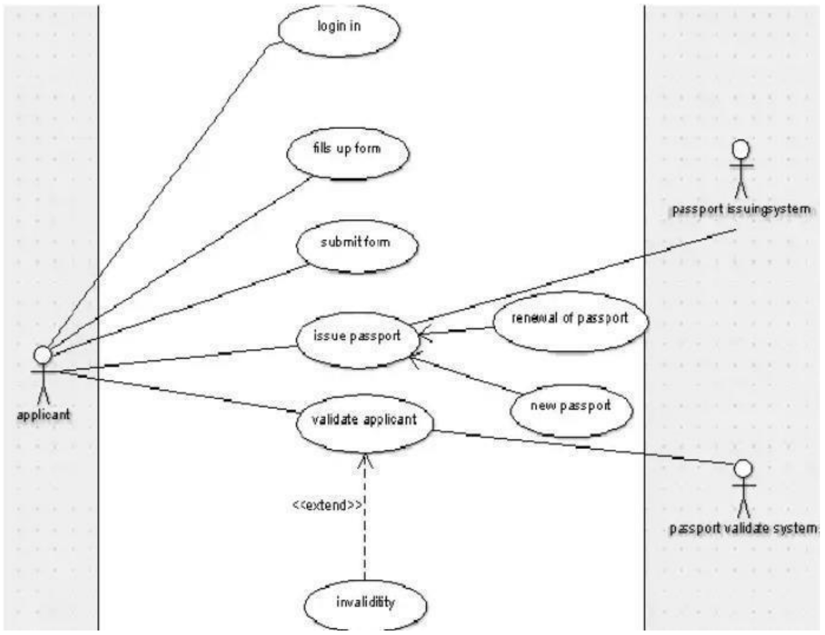
INTRODUCTION:

OBJECTIVES:

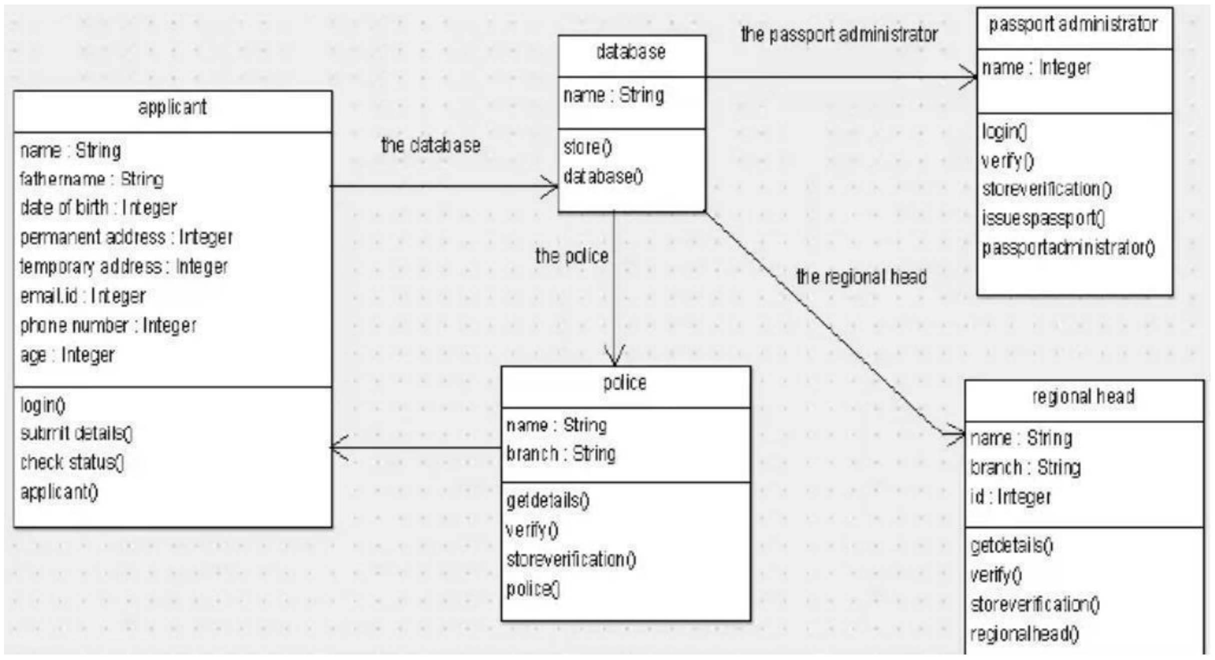
SCOPE:

PROBLEM STATEMENT:

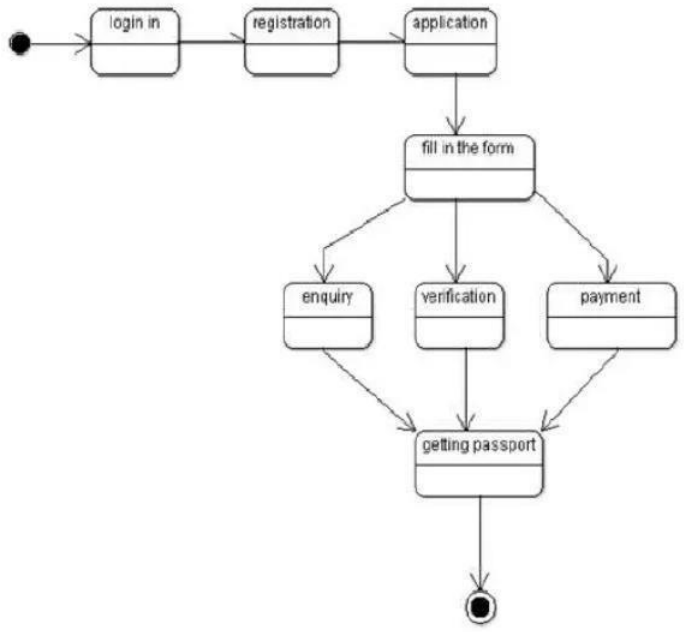
Use-case Diagram:



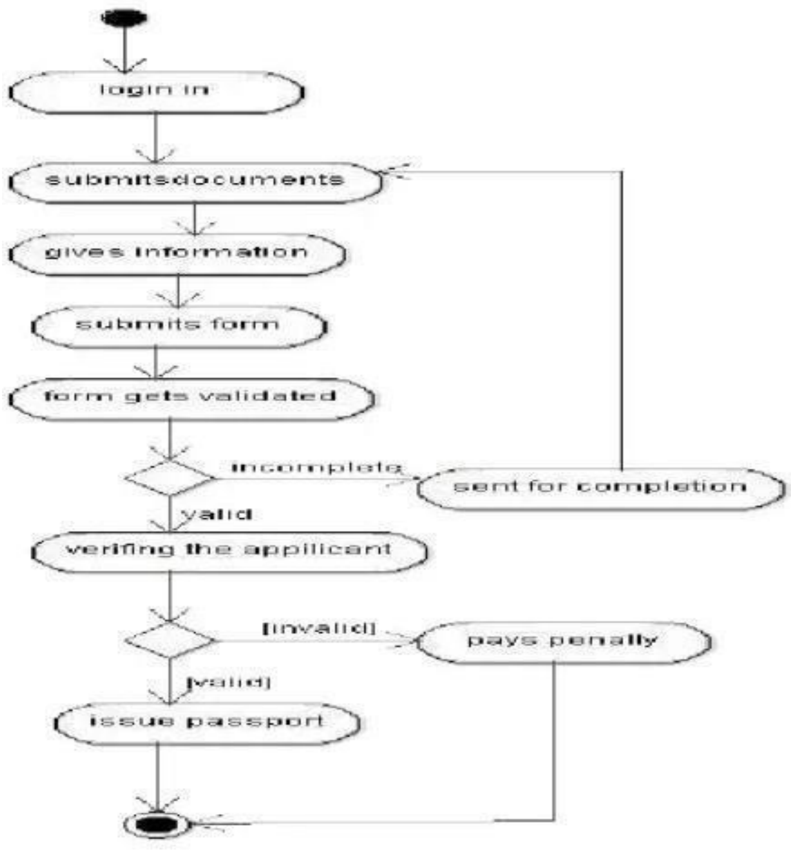
Class Diagram:



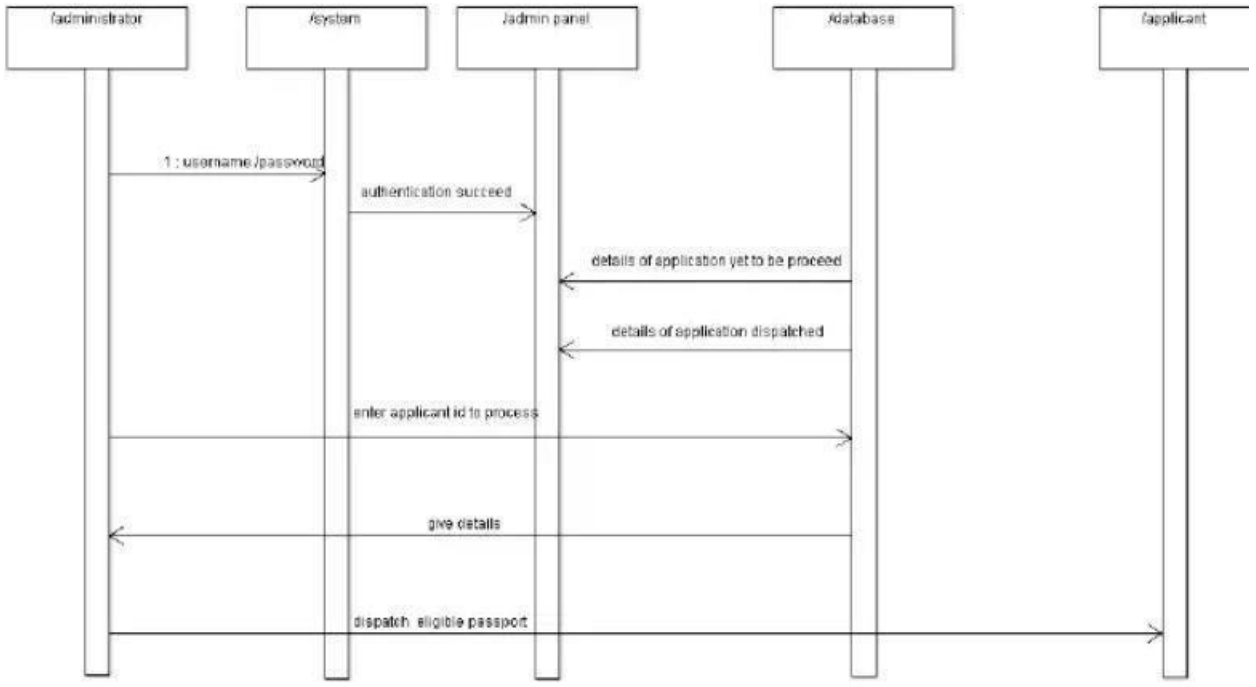
State Chart Diagram:



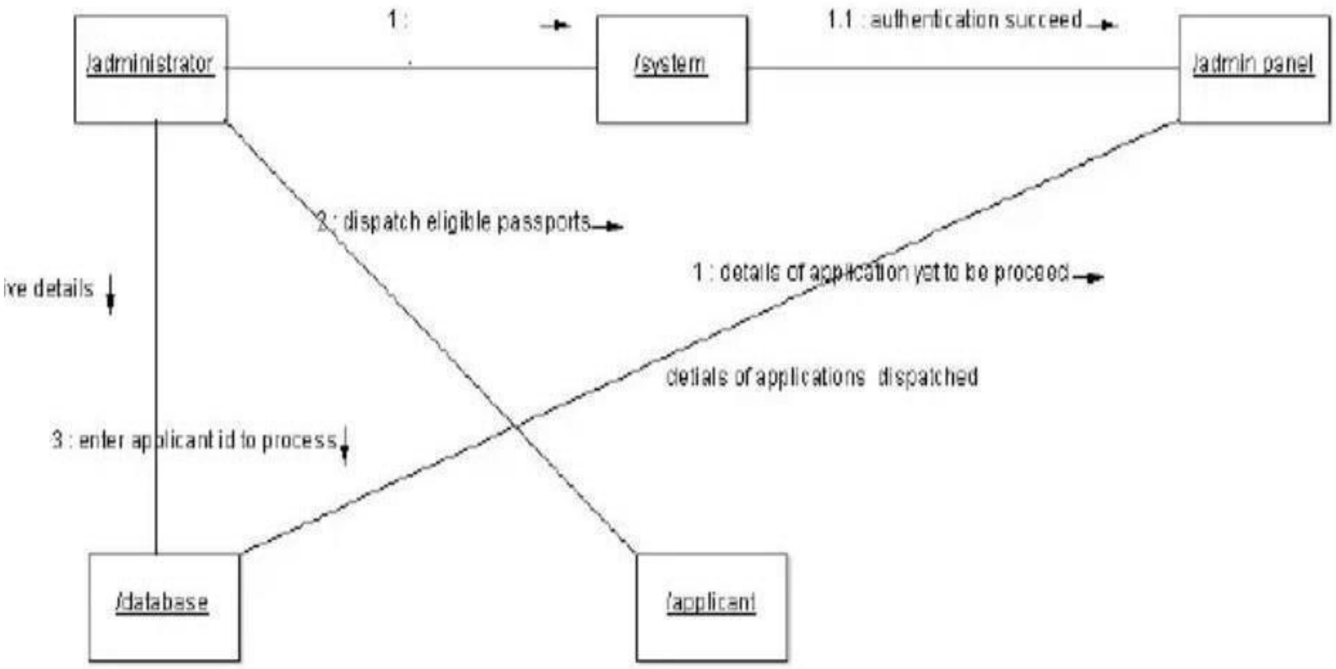
Activity Diagram:



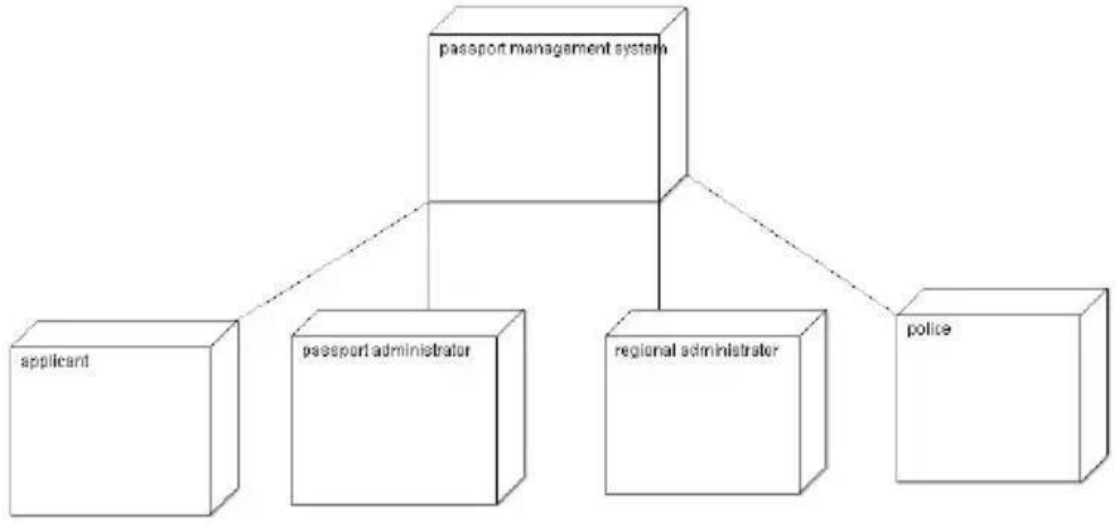
Sequence Diagram:


/administrator
/system
/admin panel
/database
/applicant
1 : username /password
authentication succeed
details of application yet to be proceed
details of application dispatched
enter applicant id to process
give details
dispatch eligible passport

Collaboration Diagram:


/administrator
/system
/admin panel
/database
/applicant
1 :
1.1 : authentication succeed
2 : dispatch eligible passports
1 : details of application yet to be proceed
detials of applications dispatched
ive details
3 : enter applicant id to process

Deployment Diagram:



STEPS TO GENERATE CODE IN STAR UML:

OUTPUT:

Airline.java

```
import java.util.List;
public class Airline {
public String name;
public String code;
public void addFlight() {
   }
public void manageBooking() {
   }
}
```

Flight.java

```
import java.util.Date
public class Flight {
   public String flightNumber;
   public String source;
public String destination;
public Date departureTime;
public Date arrivalTime;
public void checkAvailability() {
   }
public void updateSchedule() {
   }
}
```

Passenger.java

```
public class Passenger {
    public int passengerId;
    public String name;
    public String passportNumber;
    public String contact;
public void bookTicket() {
    }
public void cancelTicket() {
    }
}
```

Ticket.java

```
public class Ticket {
    public int ticketId;
    public String flightNumber;
    public int passengerId;
    public Date bookingDate;
    public double fare;
     public void generateTicket() {
    }
public void viewTicket() {
    }
}
```

Payment.java

```
public class Payment {
    public int paymentId;
    public int ticketId;
    public double amount;
    public String paymentMode;
public void makePayment() {
    }
public void refund() {
    }
}
```

Admin.java

```
public class Admin {
    public int adminId;
    public String username;
    public String password;
 public void manageFlights() {
    }
public void managePassengers() {
    }
}
```

RESULT: