PROJECT REPORT

Submitted by

Aura Bhattacharyya
RA2211003010364

Under the Guidance of

Dr. M. Suresh Anand

Department of Computing Technology

Associate Professor, Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Project Report titled **"TRAIN TICKETING SYSTEM"** is the bonafide work done by Aura Bhattacharyya RA2211003010364 who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE

Dr. M. Suresh Anand

OODP - Course Faculty

Department of Computing Technologies

SRMIST

SIGNATURE

Dr. Pushpalatha

Head of the Department

Department of Computing Technologies

SRMIST

TABLE OF CONTENTS

PROBLEM STATEMENT

The domain of the train ticketing system would encompass all the technology and processes needed to enable users to search for trains, make reservations, and pay for their bookings. The user interface would allow users to search for trains, view schedules, choose seats, and make reservations .The system would store information about trains, including their schedules, routes, available seats, and fares .The system would allow users to pay for their reservations securely, using credit cards or other payment methods .The system would provide customer support for users who have questions or issues with their reservations .The system would manage and store all data related to reservations, trains, and users .The system would ensure the security of user data and payment information .Integration with other systems: The system would integrate with other systems, such as railways reservation systems and payment gateways, to provide a seamless experience for users.

MODULES OF THE PROJECT

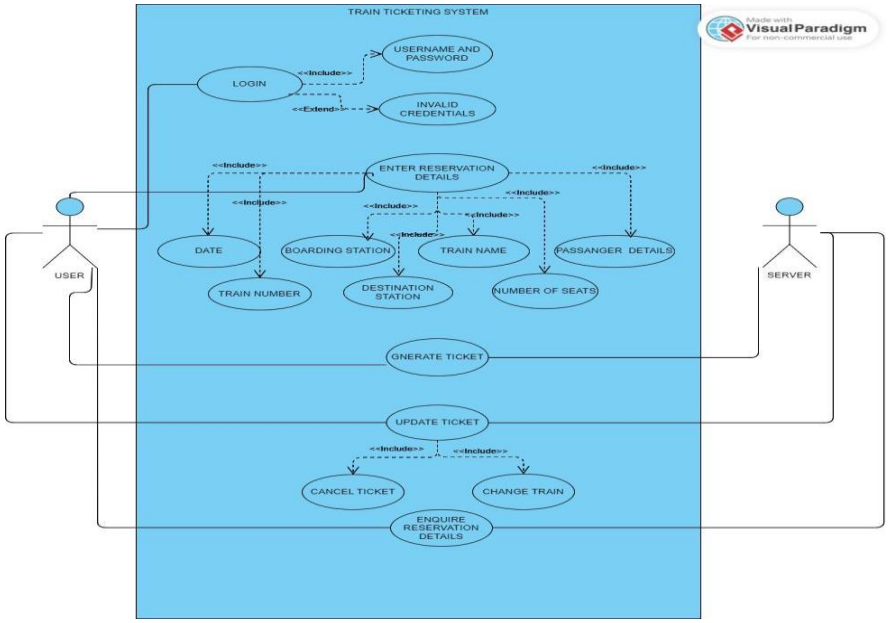
For journeys of longer distances though we have airways most of the people use the railways, which is the most convenient, affordable means of transport in India. So, keeping this in view, the reservation of railways is a most important task and it must be faster and efficient as the demand (travellers) is very high. In order to meet this demand, manual reservation is completely ruled out and it requires an efficient program to implement the online reservation. This program helps us to reserve seats of a train, cancel the previous reserved seats and also to enquire for your train.

There is choice for the users to reserve or cancel their ticket and also to enquire for their ticket. In the beginning it will ask the user to give the username and password if the details match then only the users are allowed to enter the remainder of the program .It is completely developed in C++ language without using graphics.

Research is needed in train ticketing systems for several reasons, the main objective of our project is to improve user experience, efficiency and security. Train ticketing system must be easy to use and provide a smooth user experience. Research can help identify user pain points and areas for improvement in the system's design .It should be efficient and fast, allowing customers to quickly find and book the services they need. Research can help identify bottlenecks and areas for optimization in the booking process .It must be secure and protect customer data. Research can help identify potential security vulnerabilities and recommend solutions to ensure that customer data is protected. Train Ticketing systems are becoming increasingly important for businesses in many industries. Research can help identify trends and best practices in the industry, giving businesses a competitive advantage.

UML DIAGRAMS

(a) USE CASE DIAGRAM



Use-case diagrams model the behaviour of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

During the analysis and design phases, we can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.

During the testing phase, you can use use-case diagrams to identify tests for the system. Actors used are: User(Customer, Travel agent, Airline counter staff), bank payment portal

User in use-case diagram can login with username and password into the reservation system.

To book ticket online :-

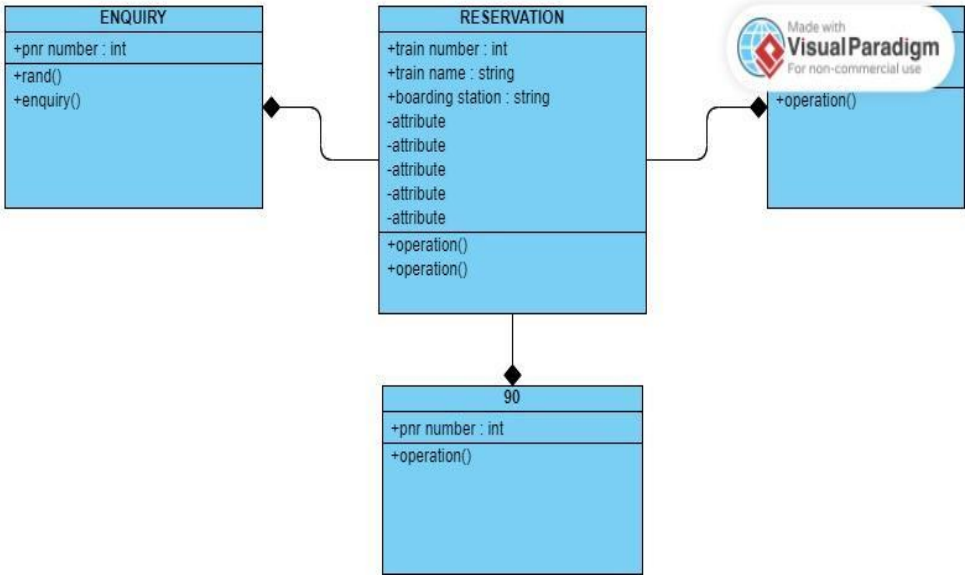
i. User has to fill train details , personal details

ii. Reserve seat

iii. Make payment through Net Banking

iv. Once payment completed , a ticket is generated with a unique ticket id for later cancellation and update of ticket.

(b) CLASS DIAGRAM



It is the most common diagram type for software documentation .It contain classes with their attributes (data members) and their behaviour(member functions)The standard class diagram is composed of three sections:

Upper section: Contains the name of the class. This section is always required, to know whether it represents the classifier or an object.

Middle section: Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.

Bottom section: Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

In the program classes are namely:

1) Login : Used to authenticate in website with help of username and password.

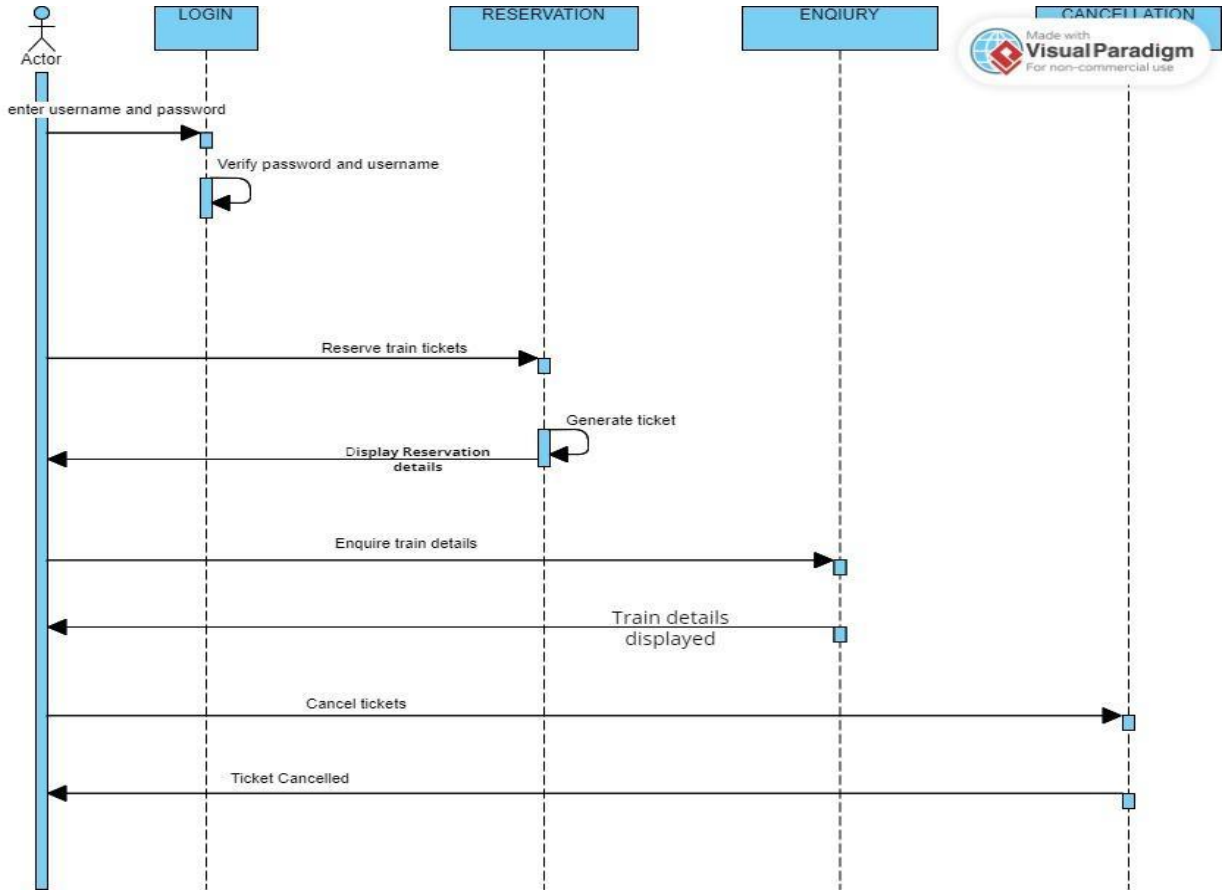
2) Passenger : Used to get personal details like name , gender of passenger.

3) BookTicketOnline : Used to select flight name , destination , date and time of train.

4) Make Payment : Allows user to select mode of payment like that of card.

5)Ticket : User can generate ticket , cancel, and change tickets.

(c) SEQUENCE DIAGRAM



Following things are to be identified clearly before drawing the UML diagram .Objects taking part in the interaction .Message flows among the objects .The sequence in which the messages are flowing .Object organization: this is the UML sequence diagram representing Train Ticketing System. The instance of the class objects involved in this UML diagram are

Login Success

Fill train Details

Update ticket

1.Actors:

An actor in a UML diagram represents a type of role where it interacts with the system and its objects. In the above UML the actor is the admin.

2.Lifelines:

A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline .Lifeline elements are located at the top in a sequence diagram .Lifeline follows the following format:

Instance Name: Class Name

E.g. Login Success

Train Details

Update train

Book ticket: It is represented with a dotted arrow and create word labelled on it to specify that it is the create Message symbol.

3.Synchronous Message:

A synchronous message waits for a reply before the interaction can move forward.

The sender waits until the receiver has completed the processing of the message.

The caller continues only when it knows that the receiver has processed the previous message i.e., it receives a reply message. A large number of calls in object oriented programming are synchronous. We use a solid arrow head to represent a synchronous message.The first message is from customer to login success(Enter username and password). Then from customer to book ticket(Book tickets). Then from customer to fill train details(round ticket). Then form book ticket to fill train details (Verify details).Then from customer to update train.

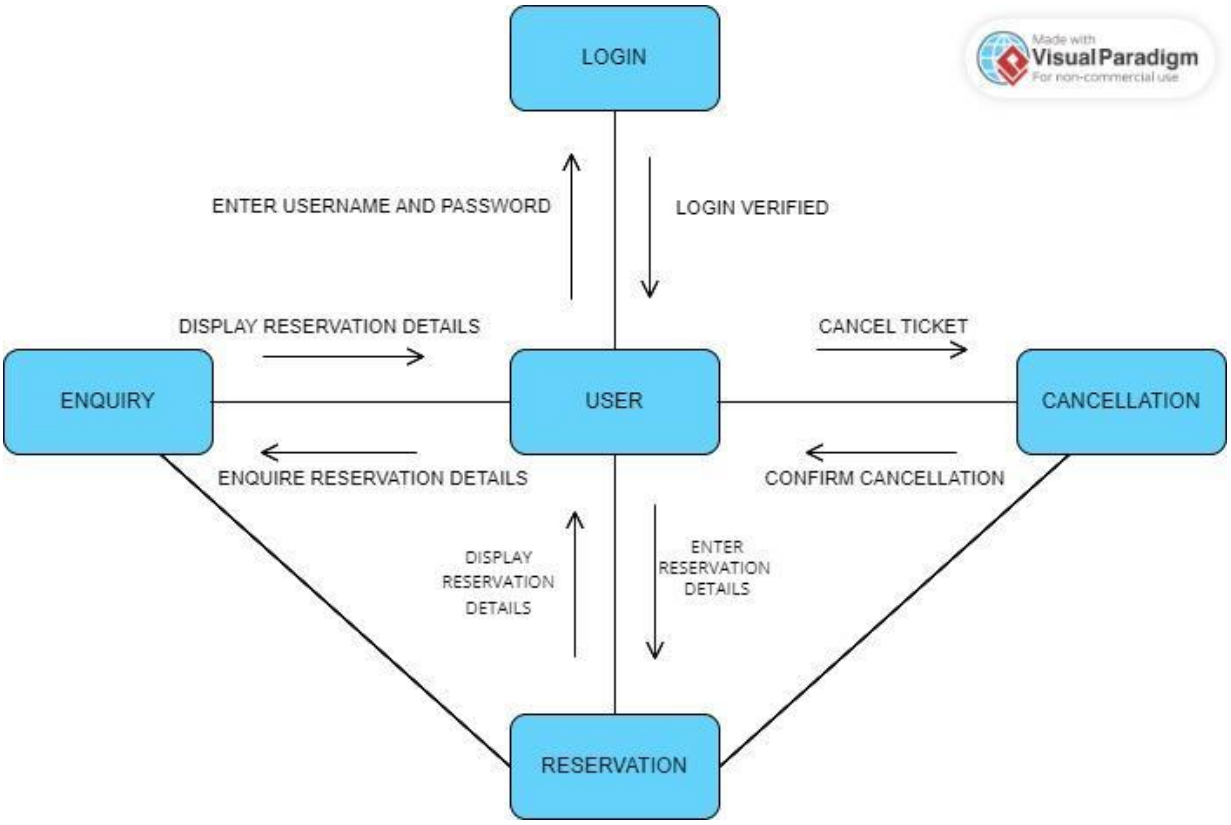
4.Self Message

A message an object sends to itself, usually shown as a U shaped arrow pointing back to itself .Verify password and username.

5.Asynchronous Messages

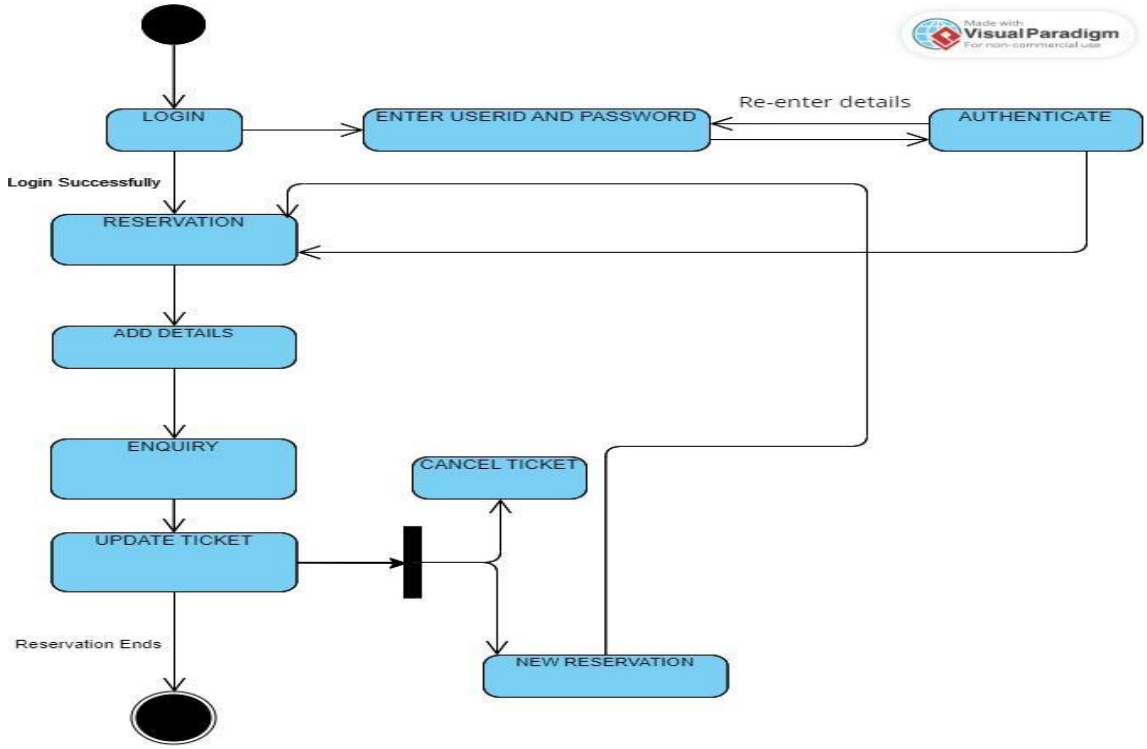
An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message .The first asynchronous message is from fill train details to customer. Then from fill train details to book ticket . Book ticket to customer .

(d) COLLABORATION DIAGRAM



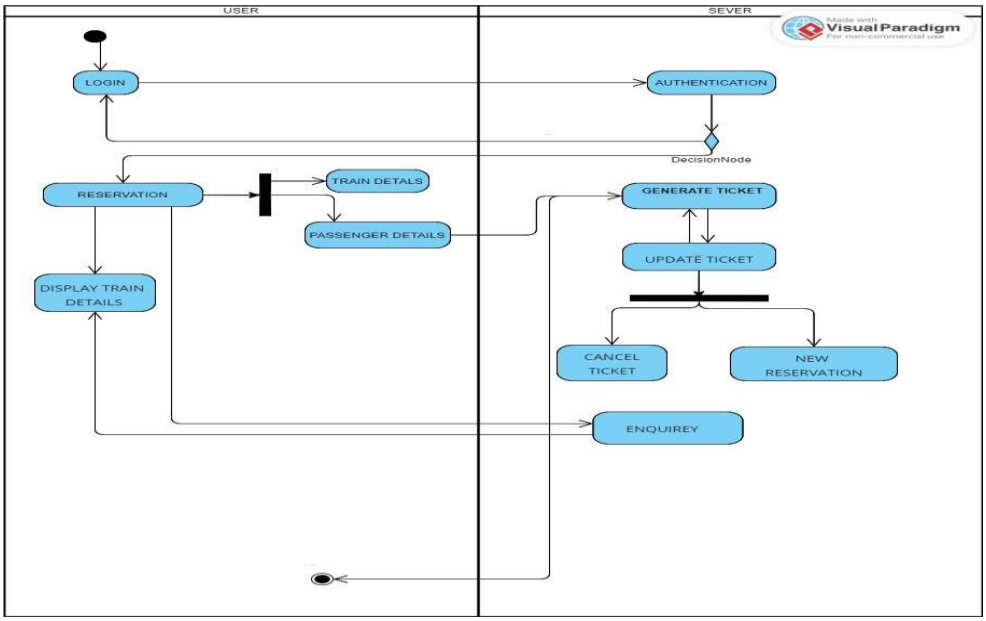
It depicts the relationships and interactions among software objects .They are used to understand the object architecture .They are also known as "Communication Diagrams.". In the collaboration diagram, the method call sequence is indicated by numbering technique which indicates how the methods are called one after another. It emphasizes the structural aspects of an interaction diagram – how lifeline connects. It allows you to focus on the elements rather than focusing on the message flow as described in the sequence diagram.

(e) STATE CHART DIAGRAM



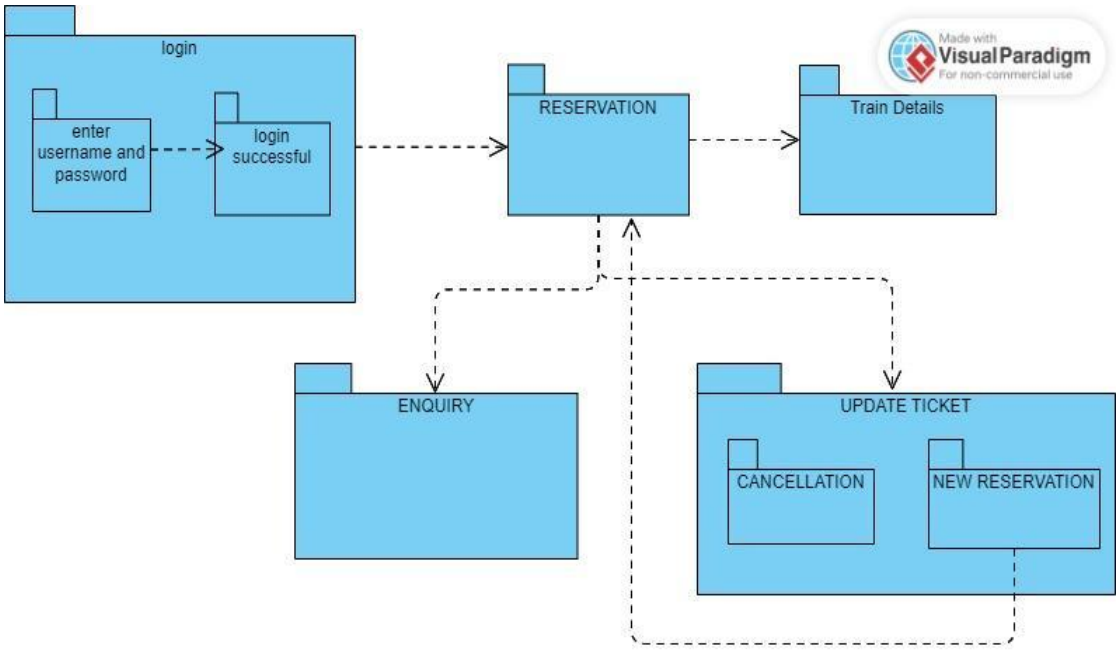
A UML state diagram is a graphical depiction that is used to explain how a system or component will behave in terms of its states, events, and transitions. It is a component of the Unified Modelling Language (UML), which is frequently employed for describing, building, and documenting software systems. A system or component is represented as a state machine in a UML state diagram, which is made up of a number of states, events, and transitions. A state depicts a circumstance or a possible configuration for the system or component. A trigger is represented as an event when a system or component changes states. A transition denotes a shift in a state .The components of a state diagram include :States: The various states of the system, which are represented by rounded rectangle. It describes the function or purpose within the system. The states in the above diagram are Login, authenticate, enter username and password, Book Ticket Online, reserve seat, Make Payment, generate ticket and update ticket .Transitions: The arrows or lines between states that show the possible transitions or movements from one state to another. The transitions in the above diagram are reservation starts, login successful, re enter details, book food online, make transactions and reservation ends .Start State: The initial state of the system, which is typically represented by a solid circle or ellipse with an arrow pointing to it .End State: The final or accepting state of the system, which is represented by a double circle or ellipse. When the system reaches this state, it has completed its task or process.

(f) ACTIVITY DIAGRAM (SWIMLANE)



Activity diagrams can be used to model a wide range of processes, including business workflows, software algorithms, and system architectures. They are particularly useful for visualizing complex processes and identifying potential bottlenecks or inefficiencies .The components of an activity diagram in UML include: Activity: An activity represents a specific action or process that occurs within the system being modelled. It is represented as an oval. The activities in the above diagram are Login, authenticate, enter username and password, Book Ticket Online, reserve seat, make payment, generate ticket and update ticket. Initial node: This represents the starting point of an activity diagram. It is represented as a small filled circle .Final node: This represents the end of an activity diagram. It is represented as a larger filled circle .Control flow: This is used to show the flow of control between different activities and actions in the diagram. It is represented as an arrow .Decision node: This is used to show a decision point in the process, where the flow of the process can take one of two or more possible paths. It is represented as a diamond shape .Fork node: This is used to show a split in the process where multiple activities can occur simultaneously. It is represented as a horizontal bar with multiple outgoing arrows .Join node: This is used to bring multiple activities back together after a fork node. It is represented as a horizontal bar with a single incoming arrow and multiple outgoing arrows .These components can be combined and arranged in various ways to create activity diagrams that accurately represent the processes or workflows of a system.

(g) PACKAGE DIAGRAM

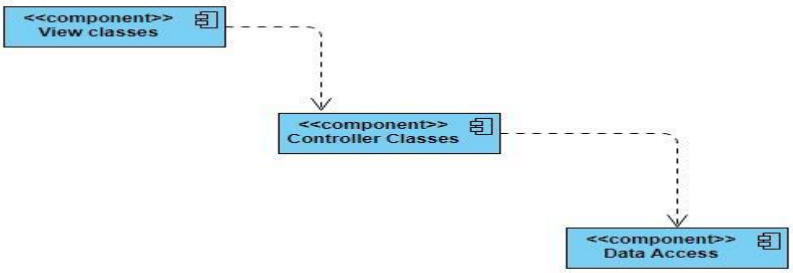


A package is a grouping of model elements. Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. Packages themselves may contain other packages. A package may contain both subordinate packages and ordinary model elements. All UML model elements and diagrams can be organized into packages.A package is represented as a folder, shown as a large rectangle with a tab attached to its upper left corner. If contents of the package are not shown, then the name of the package is placed within the large rectangle. If the contents of the package are shown, then the name of the package may be placed on the tab. The contents of the package are shown within the large rectangle. The dotted arrows joining the packages are called dependencies.

The major of the components in our package diagram are as follows:

- Login
- Book Ticket online
- Passenger Details
- Bank Portal
- Tickets

(h) COMPONENT DIAGRAM



Component diagram is a special kind of diagram in UML. It does not describe the functionality of the system but it describes the components used to make those functionalism. So, the component diagrams are used to visualize the physical components. The component diagram are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. The combinations of packages or individual entities forms a component. Various components and their dependencies are depicted using a component diagram. Component diagram contains: component package, components, interfaces, and dependency relationship A component is represented using a rectangle box, with two rectangles protruding (sticking out) from the left side. A dependency is used to model the relationship between two components. The notation for a dependency relationship is a dotted arrow, pointing from a component to the component it depends on.

The major components in our component diagram are as follows:

- User

- Login

- Railway server

- Bank server

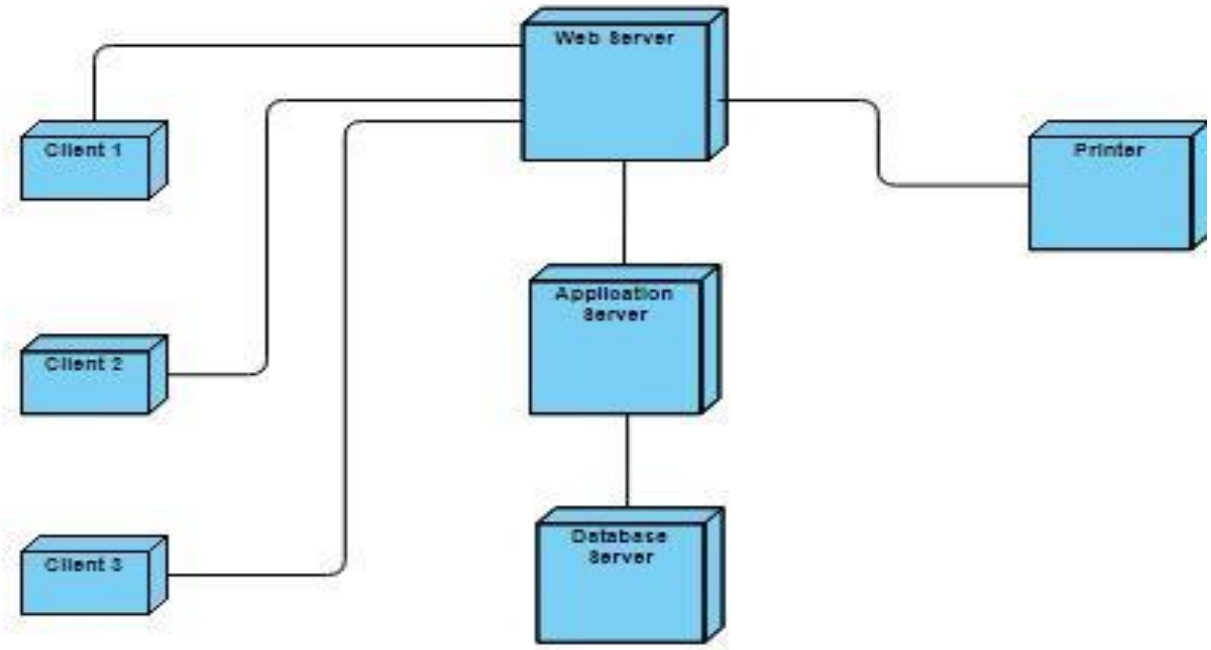
- Train details

- Personal details

- Payment Details

- Ticket

(i) DEPLOYMENT DIAGRAM



Deployment diagram is a type of diagram that shows the physical deployment of a system, including its hardware and software components, on nodes .In other words, it is a diagram that models the architecture of a system and how the system components are distributed across different hardware nodes, such as servers or devices. Deployment diagrams are useful for planning, designing, and understanding the physical infrastructure needed to support a system .The main components of a deployment diagram include:

Nodes: These represent the physical hardware devices or software execution environments on which the system components are deployed. Nodes can represent servers, workstations, laptops, routers, firewalls, or any other device that can host software.

Components: These represent the software components that are deployed on nodes. Components can be software applications, libraries, or modules that perform specific functions. The components in the above diagram are Login, authenticate, enter username and password, Book Ticket Online, reserve seat, Make Payment, generate ticket and update ticket.

Artifacts: These represent the physical files or data that are stored on nodes. Artifacts can include configuration files, databases, log files, or any other data that is needed by the system. The artifacts in the above diagram are enter username and password, name, age, gender and pre order food.

Dependencies: These represent the relationships between the different components and artifacts in the system. Dependencies can include communication links, database connections, or any other type of interaction between components.

Associations: These represent the relationships between nodes and components. Associations can show which components are deployed on which nodes, and how they are connected.

Deployment diagrams are particularly useful for understanding the physical layout of a system, and for identifying potential issues or bottlenecks in the deployment process. They can also help developers and system architects to plan for scalability, redundancy, and fault tolerance in their systems

SOURCE CODE

```
//Header files
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<fstream>
#include<cstring>
//#include<conio.h>
#include<ctime>
using namespace std;
//FSTREAM VARIABLE
fstream fin;

/*---------------------------------------------------------------------------------------------------- CLASS
CANCELL----------------------*/

class cancell {
public:
        long pnr;
        long tno;
        char tname[100];
        char bp[20];
        char dest[20];
        char name[10][200];
        int age[10];
        int d, m, y;
        int nos;
        int amt;
       char clas;
       void get();
        void screen();

};
//MEMBER FUNCTION TO GET PNR DETAILS
void cancell::get()
{
        cout << "\nEnter the details as follows: ";
        cout << "\nEnter PNR number: ";
        cin >> pnr;
}
//MEMBER FUNCTION TO SHOW CANCELL DETAILS
void cancell::screen()
{
        cout << "\nPNR number: " << pnr;
        cout << "\nTrain number: " << tno;
        cout << "\nTrain Name: " << tname;
        cout << "\nBoarding station: " << bp;

        cout << "\nDestination station: " << dest;
```

```
	cout << "\nNumber of seat: " << nos;
	cout << "\n Your Class: "
	if (clas == 'f')
	{
		cout << "\tFirst Class";

	}
	else if(clas=='s')
	{
		cout << "\tSecond class";
	}

	for (int i = 0;i < nos;i++) {
		cout << "\nName of Passenger: " << name[i];
		cout << "\n Age: " << age[i];
	}
	cout << "\nDate of cancellation: " << d << "/" << m << "/" << y;
	cout << "\nYou can collect: " << amt-250;
}

/*---------------------------------------------------------------------------------------------------------- CLASS
RESERVE ---------------------------
-----------------------------------------------------------------------------*/

class reserve
{
public:
	long tno;
	int nof;
	long pnr;
	char tname[100];
	char bd[100];
	char dest[100];
	int age[100];
	char name[10][200];
	int amc;
	char clas;
	int d, m, y;
	int sno;
	int ret()
	{
		return pnr;
	}
```

```
//MEMBER FUNCTION TO COPY DETAILS
        void ca()
        {
                cancell a;
                a.pnr = pnr;
                a.tno = tno;
                strcpy(a.tname,tname);
                strcpy(a.bp, bd);
                strcpy(a.dest, dest);
                a.d = d;
                a.m = m;
                a.y = y;
                a.clas = clas;
                a.amt = amc;

                a.nos = nof;
                for (int i = 0;i < nof;i++)
                {
                        strcpy(a.name[i], name[i]);
                        a.age[i] = age[i];
                }

                a.screen();
        }
     //MEMBER FUNCTION TO SHOW DETAILS
              void show()
        {

                int i;
                ofstream f2;
                f2.open("cancell.dat", ios::out|ios::app);


                f2 << tname << endl << tno << endl << bd << endl <<

       dest << endl << d << " " << m << " " << y << "\n";
                for (i = 0;i < nof;i++)
                {
                        f2 << name[i] << "\t" << age[i] << "\t" << pnr <<

          "\t" << sno << "\n";

                }
                f2.close();
                ca();

        }

 //MEMBER FUNCTION TO SEARCH IN FILE
        void search(int x)
```

```
{
        cancell k;
        int i;
        ifstream f1;


        f1.open("rseservation.dat", ios::in);
                if (!f1)
                {
                        cout << "Ticket Cancelled\n";
                }
                else
                {
                        fin.read((char*)this, sizeof(*this));
                        while (!f1.eof())
                        {
                                if (x == pnr)
                                {
                                        show();


                                }
                                else {
                                        cout << "\nyou have not done your reservation till: ";
                                }
                                f1.read((char*)this, sizeof(*this));
                        }


                        f1.close();
                }
}
//MEMBER FUNCTION TO GET DETAIL FOR RESERVATION
void getdata()
{
        cout << "~~~~~RESERVATION~~~~~";
        cout << "\nEnter Train Number: ";
        cin >> tno;
        cin.ignore();
        cout << "\nEnter train name: ";
        cin.getline(tname, 20);
        cout << "\nEnter Boarding Station: ";
        cin.getline(bd, 20);
        cout << "\nEnter Destination: ";
        cin.getline(dest, 20);
        cout << "\nEnter number of seats to be booked: ";
        cin >> nof;

       cout << "\nEnter the date(dd mm yyyy)";
        cin >> d >> m >> y;
```

```
            cin.ignore();
            for (int i = 0;i < nof;i++)
            {
                        cout << "\nEnter Name: ";
                        cin.getline(name[i], 20);
                        cout << "\nEnter age: ";
                        cin >> age[i];
                        cin.ignore();
            }
            cout << "\nEnter first-class or second class(f/s): ";

            cin >> clas;

            cout << "-------------------------------------------";
            cout << "end of reservation";

            cout << "-------------------------------------------";
            if ((clas == 'f')||clas=='F')
            {
                        amc = 2500 * nof;
            }
            else if ((clas == 's'||clas=='S'))


            {
                        amc = 1200 * nof;
            }
  }
  //MEMBER FUNCTION TO SHOW RESERVATION DETAILS
  void putu()
  {

            int i;
            srand(time(NULL));
            fin.open("reservation.dat", ios::app|ios::out);
            cout << "\nTrain Name: " << tname;
            cout << "\nTrain number: " << tno;
            cout << "\nboarding station: " << bd;
            cout << "\ndestination: " << dest;
            cout << "\nperson information: ";
            cout << "\nTravelling date" << d << "/" << m << "/" <<
y;
            pnr = rand() % 9000000 + 1000000;
            cout << "\tPNR: " << pnr;
            for ( i = 0;i < nof;i++)
            {


                        cout << "\nName: " << name[i];
                         cout << "\tage" << age[i];
```

```
                    sno = rand() % 64 + 1;
                    cout << "\tseat number: " <<sno ;
                }

                cout << "\nTotal Fare" << amc;
                fin << tname << endl << tno << endl << bd << endl <<
        dest << endl << d << " " << m << " " << y<<"\n";
                for (i = 0;i < nof;i++)
                {
                    fin << name[i] << "\t" << age[i] << "\t" << pnr <<
          "\t" << sno<<"\n";
                }
                fin.close();
        }

 //MEMBER FUNCTION FOR ENQUIRY

        void enquiry(int n)
        {

                int pl;
                int h, m;
                int t1, t2,m1,m2;
                t1 = rand() % 24 + 1;

                m1 = rand() % 59 + 1;
                m2 = rand() % 59 + 1;

                pl = rand() % 6 + 1;
                h = rand() % 2 + 0;
                m = rand() % 60 + 0;
                cout << "\ntrain will arrive in Platform Number" << pl;
                cout << "\nTrain is late by " << h << " hours and " <<
            m << " minutes";
                cout << "\nActual Arrival time :- " << t1 << ":" << m1;
                cout << "\nArrival time" << h + t1 << ":" << m;
                cout << "\nDeparture time" << t1 + h + 20 << m2;


        }
};




//MAIN FUNCTION
int main()
```

```
{
        reserve r2;
        cancell o;
        int z=0;
        ifstream f1;
        ofstream f2;

        char *id="ADMIN";
        char *password = "12345678";
        char n[50];

        char pass[50];
        int ch;

        cout << "~~~~~~~~~~~~~Welcome TO INDIAN RAILWAY~~~~~~~~~~~~~~~~~~~~";


h:
        cout << "\n Enter your id";
        cin >> n;
        cout << "\nEnter password";

    cin >> pass;
                if ((strcmp("ADMIN", n) != 0) && (strcmp(password,
         pass) != 0))
                {
                        cout << "\n\nWrong Password\n\nPlease Enter The Correct password";
                        goto h;
                }
                else
                {
                        k:
                        cout << "\n\n\n\nENTER YOUR
CHOICE\n1)RESERVATION\n2)CANCELLATION\n3)ENQUIRY\n4)EXIT";

          cin >> ch;

          switch (ch)
                        {
                        case 1:
                                r2.getdata();
                                r2.putu();

                                goto k;
                                break;
                        case 2:
                                cout << "---------------------------- CANCELLATION----------------------
-----------------"<<endl;
                                cout << "Enter the pnr : ";
                                cin >> z;
```

```
                    r2.search(z);
                    goto k;

                    break;
                case 3:
                    cout << "Enter the pnr : ";
                    cin >> z;
                    r2.enquiry(z);
                    goto k;
                    Break;


                case 4:
                    exit(0);
                }
            }
        return 0;
}
```

OUTPUT SCREENSHOT

RESERVATION:

```
~~~~~RESERVATION~~~~~
Enter Train Numebr: 12345

Enter train name: SD Express

Enter Boarding Station: Chennai

Enter Destination: Pune

Enter number of seats to be booked: 2

Enter the date(dd mm yyyy)05 06 2023

Enter Name: Anjali

Enter age: 18

Enter Name: Aura

Enter age: 18

Enter first-class or second class(f/s): f
-----------------------------------------------end of reservation-----------------------------------------------
Train Name: SD Express
Train number: 12345
boarding station: Chennai
destination: Pune
person information:
Travelling date5/6/2023 PNR: 7874913
Name: Anjali    age18   seat number: 35
Name: Aura      age18   seat number: 57
Total Fare5000
```

ENQUIRY:

```
ENTER YOUR CHOICE
1)RESERVATION
2)CANCELLATION
3)ENQUIRY
4)EXIT3
Enter the pnr : 7874913

train will arrive in Platform Number2
Train is late by 1 hours and 22 minutes
Actual Arrival time :- 14:3
Arrival time15:22Ndeparture time354
```

CANCELLATION:

```
ENTER YOUR CHOICE
1)RESERVATION
2)CANCELLATION
3)ENQUIRY
4)EXIT2
---------------------------CANCELLATION--------------------------------------
Enter the pnr : 7874913
Ticket Cancelled
```

CONCLUSION AND RESULTS

Train ticketing systems have revolutionized the way people book trains and travel today. They have made it more convenient, accessible, and efficient to book flights from the comfort of your home or office. Here are some of the key benefits and applications of the Train Ticketing System:

Time-Saving: Saves a lot of time for travellers as they can easily search for flights, compare prices, and book tickets online without having to physically visit a travel agency or ticket counter. This has made the booking process more efficient, hassle-free, and convenient.

Cost-Effective: They offer exclusive deals, discounts, and promotions to their customers. This allows travellers to save money on their flights and travel expenses. Additionally, users can easily compare prices and choose the best deal that suits their budget and needs.

Flexibility: The systems offer users the flexibility to choose from a wide range of options, timings, and dates. Users can easily filter their search results based on their preferences and travel requirements. This allows travellers to plan their trips more efficiently and effectively.

Convenience: These systems are available 24/7, which means users can book trains at any time of the day or night. This allows travellers to book trains on their own schedule and convenience.

Customer Service: Many such systems offer customer support and assistance to their users. This means users can easily contact customer service representatives to get help with their bookings, refunds, cancellations, and other queries.

Overall, these systems have transformed the travel industry by making it more accessible, efficient, and cost-effective for travellers. They have also helped railways and travel agencies to streamline their operations, reduce costs, and increase customer satisfaction.

REFERENCES

1.https://www.w3schools.com/cpp/default.asp

2. https://www.freeprojectz.com/uml-diagrams

3. https://www.onlinegdb.com/online_c++_compiler