WESTERN RAILWAY TICKET BOOKING SYSTEM

A Project by:

Mrinalini Vettri

INDEX

INTRODUCTION

In the fast forward world of technology, everyone is running behind time. Thus the main motivation of technology is to produce a time and cost efficient product. The mobile devices are becoming more and more popular and are providing a new notion of communication that we could once only imagine. With respect to Mumbai, one of the major problems faced by the seventy lakh people, who travel by local trains every day, is standing in the long queues for an average of 10-15 minutes to buy a ticket. This often leads to people traveling without tickets at all. This project aims to find a remedy for these seventy lakh people by using an online application to book tickets on their phone. It will reduce the average minimum 3170 minutes they spend standing in the line annually.

The railway department online ticket booking or e-ticketing was introduced for facilitating the users to book ticket on internet via a governmental website. The printout of the ticket may be used for validation. Later M-ticketing (Mobile ticketing) was introduced which sends user messages of tickets for validation purposes after booking the tickets through online ticket portals.

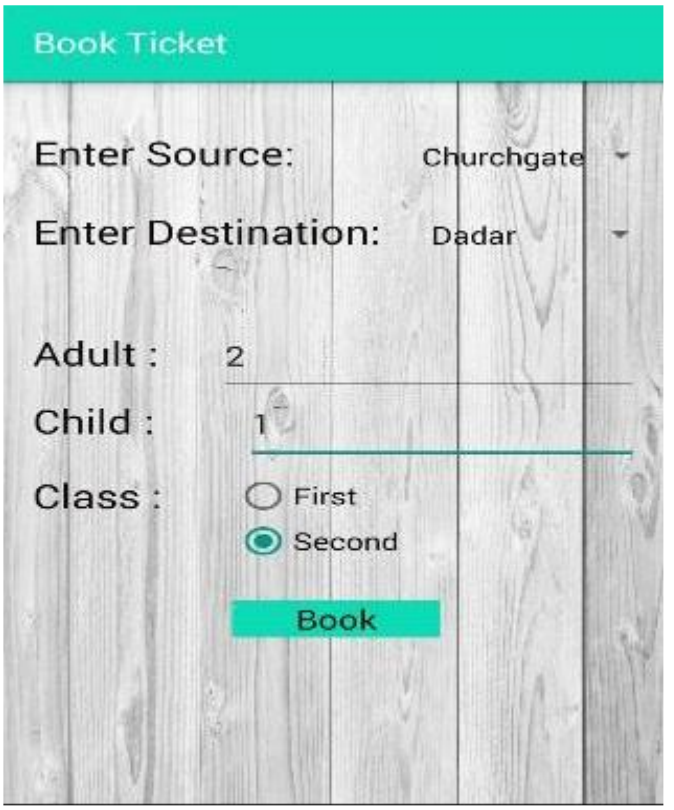
The Western Railway Ticket Booking System is basically a way of buying tickets for Mumbai's Western Line, which is an extremely easy task. WRTBS is a simple application which enables users to buy tickets in an efficient manner, with the help of a smart application. The Western Railway Ticket Booking System project for local trains enables passengers to book local train tickets and gets ticket receipt online. This local train project gives login rights to normal clients and administrator. A normal client may login and get a ticket on the web, print it and travel by train. The ticketing procedure comprises of a ticket booking form.

The user will have to enter the train details like train source and destination, number of tickets indicating adult or child passengers, and First Class or Second Class. The train details will be stored in the database which will be accessed by the application. The project database is already

filled with stations on Mumbai western local line. The system likewise comprises of an alternative to choose if the passenger wants a single journey or a return ticket.

This project aims to provide an incredible and much needed solution, which will benefit more than half of the population of Mumbai and make their daily routine easy and enjoyable.

A sample image of how the booking application will look like



SYSTEM ANALYSIS

a) **Technical Feasibility:** The new system is at par with the existing hardware and technical capability of the Indian Railways. The internal technical capability is sufficient to support the project requirements. The current technical resources need no upgradations or additional hardware. However, it might require some additional, minor hardware upgradations if additional functionalities to the local train ticket reservation process are to be added in the software, in future.

A thorough system analysis has been done in order to understand the problem, which is to be solved. That is very important for the development of the database system. The requirements and the collection analysis phase produce both data requirements and functional requirements. The data requirements are used as a source of database design. The functional requirements of the application consist of user-defined operations that will be applied to the database (retrievals and updates).The functional requirements are used as a source of application software design.

Software Requirements

- Java Development Kit (JDK)
- Notepad
- Windows XP

Hardware Requirements

- Hard Disk – 80 GB
- RAM – 1 GB
- Processor – Dual Core or Above
- Monitor
- Keyboard
- Mouse
- Processor (Pentium 4)
- Mother Board (Intel dual or quad core)
- Device (Smart Phones)

b) **Economical Feasibility:**

Existing System: The system should be updated regularly on the railway fares and tariff. Security and ticket duplication issues are considered as major drawback of the existing system. The passengers are required to wait in the queue for booking the tickets. Sometimes instead of excessive waiting in the queue, people prefer to travel by their own vehicle or sometimes they just cancel down their journey due to the problems they have to face for buying tickets only. Many travel without buying a ticket because of the delay that might be caused by excessive waiting in the queue. Booking ticket via ticket counter after killing time on the ticket window, may sometimes lead to missing up of the train as well.

Proposed System: The user need to select the source and destination of the travel. The user can also opt whether it is single journey or double-way journey. The user can select the class for the travel. The main advantage of using this application in your devices is that you can book your tickets online, of your own choice and you do not need to waste your time just waiting in the queue for your number to come for buying tickets.

Conclusion: The new software is not only cost effective but also time saving and labour intensive. The short-term costs might be overshadowed by long-term gains but the new system has enough capability to produce immediate reduction in operating costs. The working environment of employees and customers is going to change since the new system is more efficient than the existing one and hence eliminates the need of managing various aspects of ticket booking, manually.

The development costs are a one-time investment which will not recur after the completion of the software while the operation costs will reduce considerably after successful execution. Being labour-intensive software, variable costs relating to personnel, appliance and network usage are predicted to reduce over time. The layout of the new system

is user-friendly and hence allows employees as well as users to quickly adapt to the new environment.

It is to be noted that fewer processing errors, increased efficiency, decreased response times, elimination of job steps, reduced credit losses and reduced expenses make the benefits of implementing the new software, tangible.

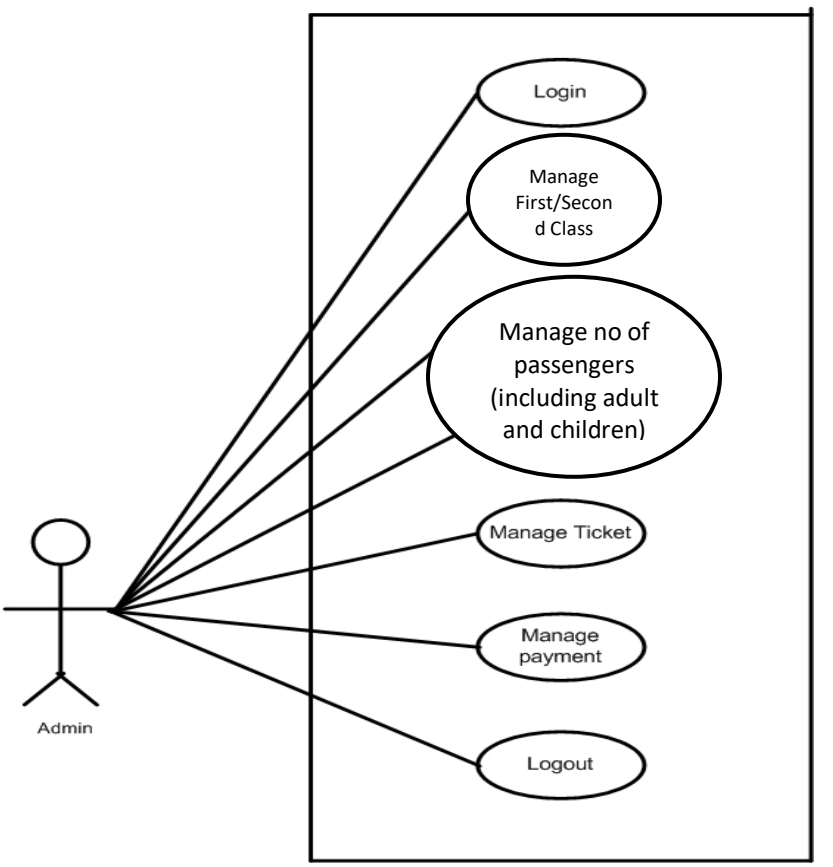
SYSTEM DESIGN

Listed below are the modules for Western Railway Ticket Booking System project:

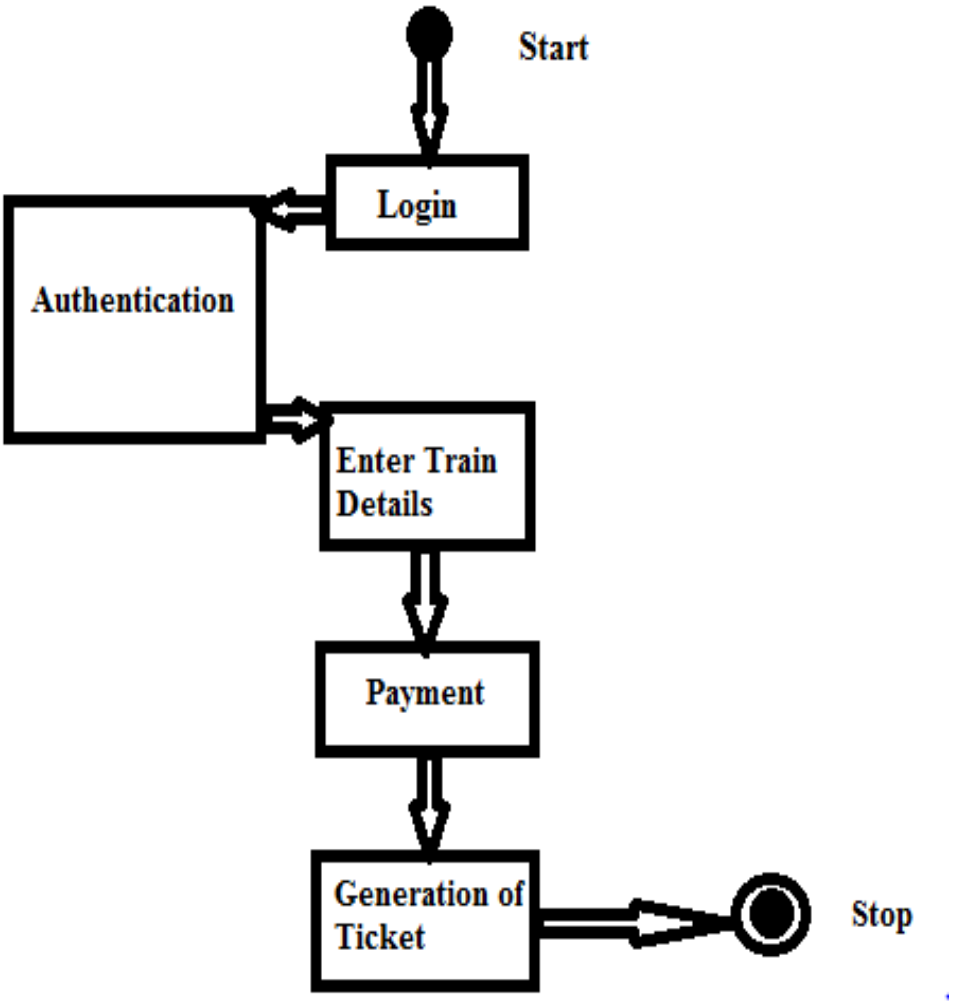
- **Ticket Window Module:** Booking of the tickets is done in this window by the user.

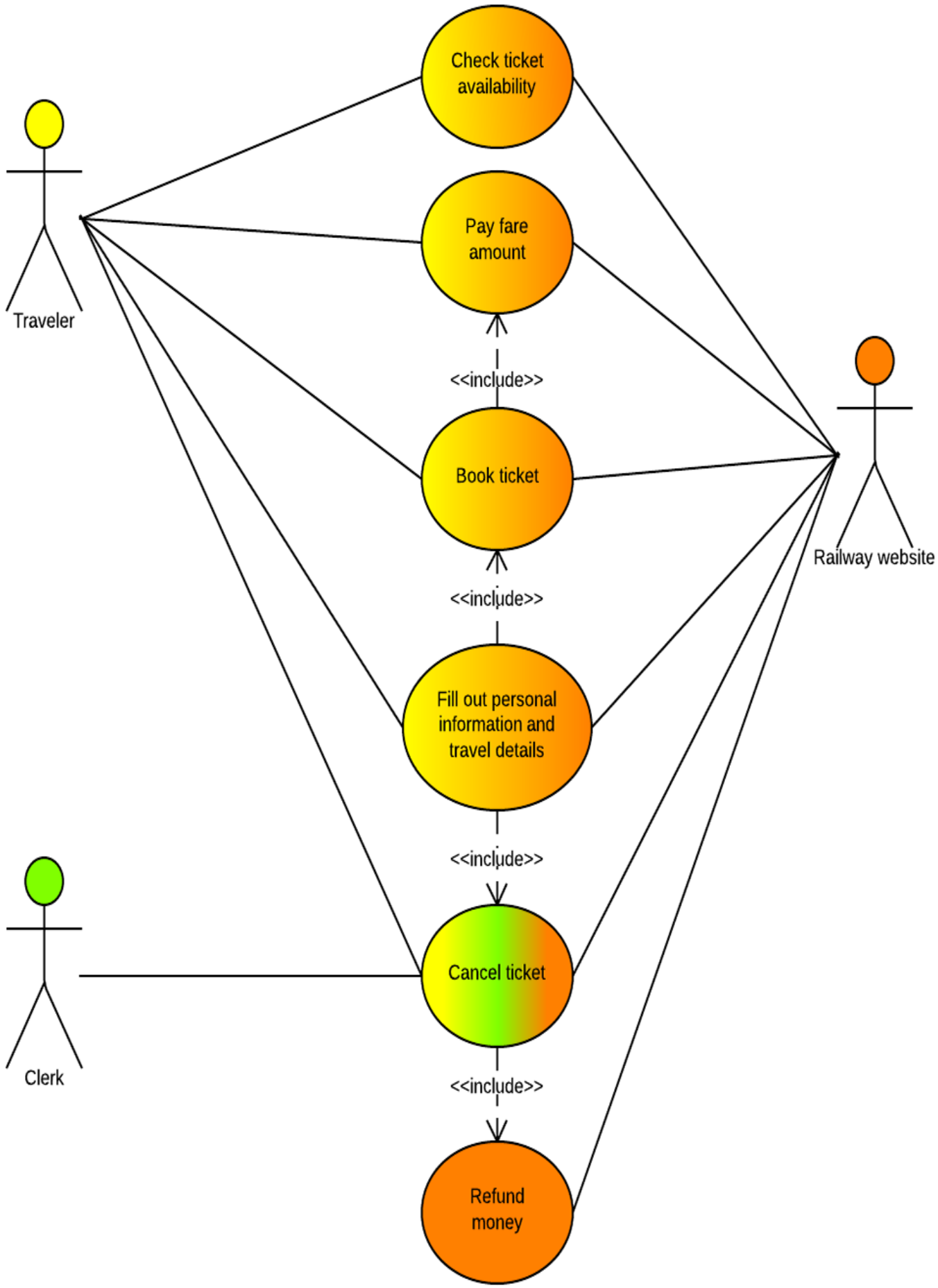
- **Admin Module:** All the updates and issuing of the tickets are done using this module. Admin also updates all the users about their payment and ticket status.

Diagrammatic Representation of Admin Module:

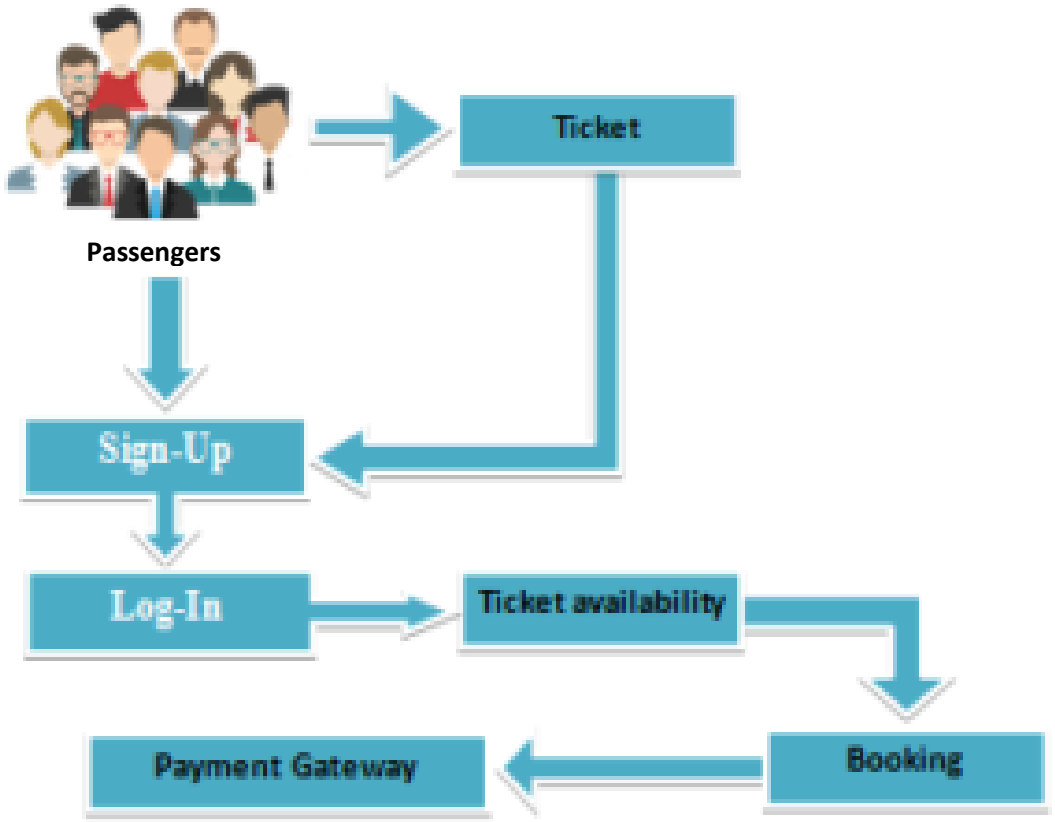


Flow Chart of the working of the project:



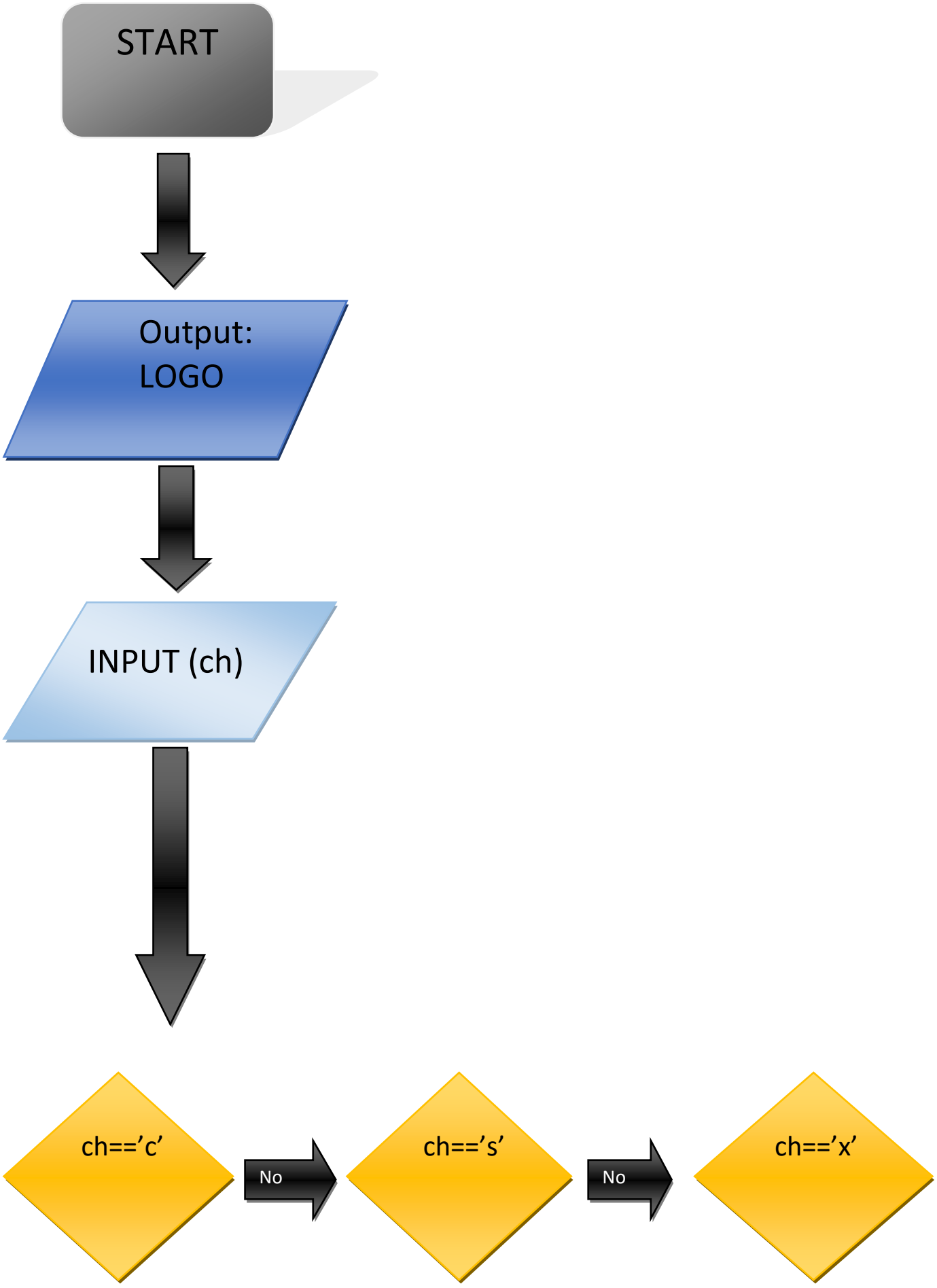

Check ticket availability
Pay fare amount
Book ticket
Fill out personal information and travel details
Cancel ticket
Refund money
<<include>>
<<include>>
<<include>>
<<include>>
Traveler
Clerk
Railway website

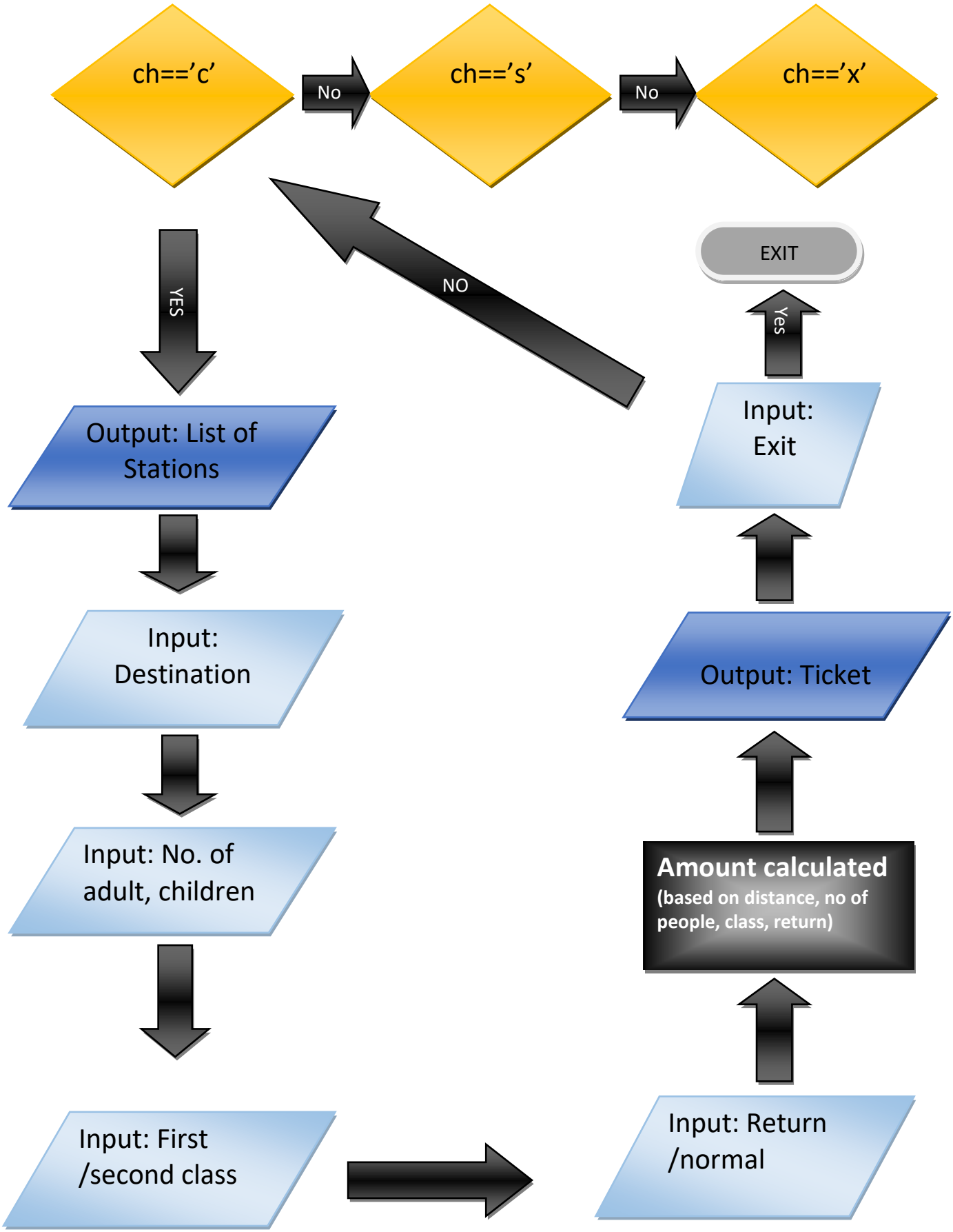
Booking Tickets:

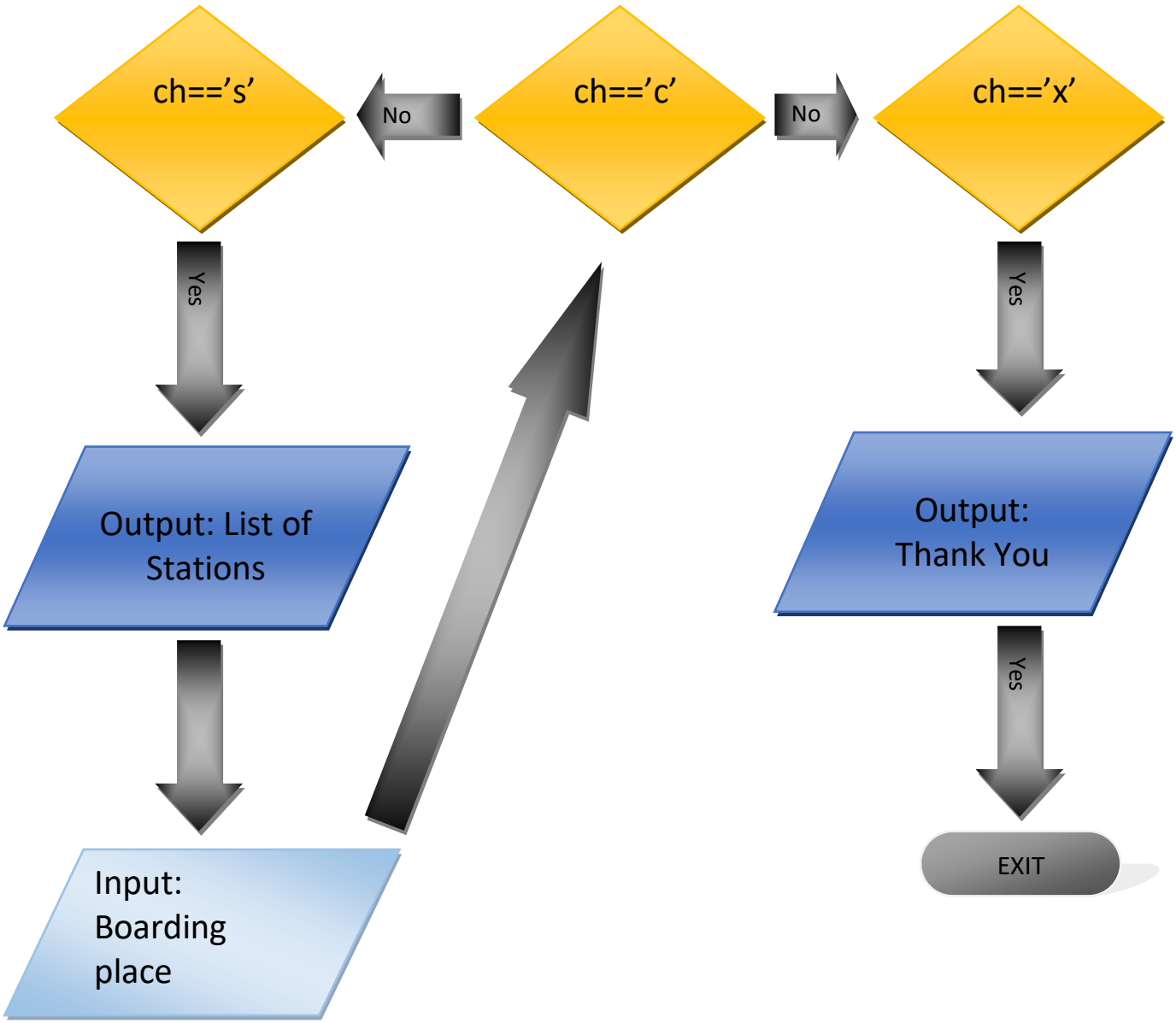


A sample image of how the ticket will look like after it is generated:

a) **Algorithm in the form of a Flow Chart**




ch=='c'
No
ch=='s'
No
ch=='x'
YES
NO
EXIT
Yes
Output: List of Stations
Input: Exit
Input: Destination
Output: Ticket
Input: No. of adult, children
Amount calculated
(based on distance, no of people, class, return)
Input: First /second class
Input: Return /normal


ch=='s'
No
ch=='c'
No
ch=='x'
Yes
Yes
Output: List of Stations
Output: Thank You
Yes
Input: Boarding place
EXIT

SOURCE CODE

```
import java.util.* ; // Importing Utility class
import java.io.*;
public class Welcome
{
   static Scanner sc = new Scanner (System.in);
   char ch;
   Input in = new Input();
   Structure struc = new Structure();
   Settings st = new Settings();
   Exit exit = new Exit();
   Error error = new Error();
   int n;
   // Declaring objects of all other classes and declaring required variables
   //Welcome ob = new Welcome();
   public Welcome () // putting forth the concept of Constructor
   {
      ch = '\u0000' ;
      n = 0;

      // intializing previously declared variables

   }
   public static void display ()
   {  Welcome ob = new Welcome();

System.out.println("\f,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n"+
            "----------------------------------------------------------------------------TICKET VENDING
MACHINE V1.0--------------------------------------------------------------------------------\n"+

"^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n\n\n"+
            "                                                              #***#\n"+
            "                                                            #*******#\n"+
            "                                                          #***********#\n"+
            "                                                        #***************#\n"+
            "                                                      #*******************#\n"+
            "
#***********************#\n"+
            "
#***************************#\n"+
            "
#*******************************#\n"+
```

```
          "                                                    #------------------------------#\n"+
          "                                                       <MUMBAI SUBURBAN
RAILWAYS>     \n"+
          "                                                    #------------------------------#\n"+
          "
#*******************************#\n"+
          "
#***************************#\n"+
          "
#************************#\n"+
          "                                                        #********************#\n"+
          "                                                         #****************#\n"+
          "                                                          #************#\n"+
          "                                                           #*******#\n"+
          "                                                            #***#\n"+
          "                                                             \n\n\n"              );

System.out.println("************************************************************
************************************************************************
*****************************************************\n"+
          "-------------------------------------------------------------------------------WELCOME---
-------------------------------------------------------------------------\n"+

"************************************************************************
************************************************************************
*************************************\n");
      ob.main();
   }
   public void main()
   {
         do {
         String str = "Enter 'c' to start issuing the ticket\nEnter 's' to change boarding
place (default is Virar)\nEnter 'x' to EXIT the application\t\t\t";
         // Displaying Choice for the user to decide what function is to be undertaken
         for(int ii = 0 ; ii < str.length() ; ii++) // Displaying 'str'
         {

            System.out.print(str.charAt(ii));
            for (int iii = 0 ; iii < 999999 ; iii++) // delay loop
            {
               for (int i = 0 ; i < 99 ; i++) // for running text effect
               {

               }
            }
         }
```

```
        ch = sc.next().charAt(0); // entering the users choice for the function
        switch(ch)
        {
           case 'c': struc.body(); // Go ahead with the program (Redirect to 'structure')

           break;
           case 's':st.set(); // Change departure station (Redirect to 'settings')

           break;
           case 'x':exit.exit(); // Terminate the program (Redirect to 'exit')
            break;
           default:System.out.println(error.choice());

           //If wrong value is entered, the error message is shown and input is asked
again
        }

      }
      while (n==0); // Continuing to run the loop until user does not mention 'x' (exit)

   }
}
public class Structure // main body of the program
{
   public void body ()
   {
      Scanner sc = new Scanner (System.in);
      int adult = 0;
      int child = 0;
      char class_type_s=' ';
      char returns;
      char exitval;
      double price = 0.0;
      String dest_s = "";
      String dest_l = "";
      String class_type_l = "";
      String date_to_pass = "";
      String returnl= "";
      String boarding_s = "";
      // declaring all required variables including the ones storing Return values
      Settings Settings = new Settings();
      Welcome start = new Welcome ();
      Input in = new Input();
      Stations stations = new Stations();
      Ticket tc = new Ticket();
```

```
        Amount amt = new Amount();
        Error error = new Error();
        Exit ex = new Exit();
        Secondary_info info = new Secondary_info();

        stations.display();
        boarding_s = Settings.boarding_s;
        // Calling objects of all classes as it contains the return values

        int n = 0;
        // acts like a flag variable to check if condition holds true
        do
        {
           System.out.println("Enter destination");
           dest_s = sc.nextLine(); // taking input of shortform from the user
           dest_l = Stations.check(dest_s); // corresponding the shortform the the long
string in the 'Stations' class
           if(dest_l.equals(""))
           {
              System.out.println(error.choice()); // invalid choice by invalid character
           }
           else
           if(dest_s.equals(boarding_s))
              System.out.println("The boarding and destination cannot be same");
              // invalid choice by departure and arrival beind the same
           else
           {
              System.out.println("Place selected: "+dest_l);
              n=1; // valid option
           }

        }
        while (n==0); // re-running program in case of error

        n=0; // re-initialization
        do
        {System.out.println ("\nEnter the number of adults:\t"); // storing number of adults
         try
         {
            adult = sc.nextInt();
            n=1;
         }
         catch (Exception e)
         {
           System.out.println(error.Format());
         }
```

```
        System.out.println ("\nEnter the number of children:\t"); // storing number of
children
        try
        {
           child = sc.nextInt();
           n=1;
        }
        catch (Exception e)
        {
          System.out.println(error.Format());
        }

       }
       while(n==0); // in case of error re-run the code

       n=0; // re-initialization
       do
       {
          System.out.println("\nPress f for first class\nPress s for second class"); //
Selection for first class of second class
          class_type_s= sc.next().charAt(0); // number of second class tickets
          class_type_l=info.trainclass(class_type_s); // Calculating with the second class
tickets

          if(class_type_l.equals(""))
             System.out.println(error.character()); // entered wrong character
          else
          {
             System.out.println("Class selected: "+class_type_l); // valid choice for __class
             n=1; // updating value to stop DoWhile from running
          }

       }
       while (n==0); // re-running the program

       n=0; // re-initialization
       do
       {
          System.out.println("\nPress r for Return \nPress n for One Way"); // one way or
return ticket
          returns=sc.next().charAt(0); // entering return
          returnl=info.returnv(returns); // Calculating with the Return Ticket
          if(returnl.equals("Wrong choice"))
             System.out.println(error.character()); //entered wrong character
          else
          if(returnl.equals("One Way"))
```

```
        {
          System.out.println("\nType Selected: Normal"); // entered one way ticket
          n=1;
        }
        else
        {
          System.out.println("\nType Selected: Return"); // entered return ticket
          n=1;
        }

      }
      while (n==0); // re-running the snippet in case of error

      date_to_pass=info.date();
      amt.amount(boarding_s,dest_s,returnl,class_type_s,adult,child);
      price=amt.amt;
      tc.tic(dest_s,class_type_l,boarding_s,adult,child,price,date_to_pass,returnl);
      n=0;// Re-initialization
      // Redirection to 'ticket' class after calculating date
      do
      {
        System.out.println("\nPress c to continue or press x to exit"); // Choosing the
welcome choice again
        exitval=sc.next().charAt(0); // Entering choice
        if(exitval=='c' || exitval=='C') // if the user wants to book more tickets once the
transaction is over
        {
          start.display();
          n=1;
        }
        else
        if(exitval=='x' || exitval=='X' ) // to terminate the program
        {
          ex.exit();
          n=1;
        }
        else
          System.out.println(error.character()); // Redirection to 'error'
      }
      while (n==0); // Re-running in case of error

    }

}
public class Input
{
```

```
   static Scanner sc = new Scanner (System.in); // Declaring Scanner
   static Error error=new Error();
   public char character() // Providing input for users (Welcome through Structure)
character choice
   {

      char c1= ' ';
      boolean n=true;
      while(n == true)
      {
         try
         {
            c1 = sc.next().charAt(0); // Taking users input for choice
            n = false; // Stopping While loop next run
         }
         catch(Exception e)
         {
            System.out.println(error.DataType()); // Redirecting to "Error"
         }
      }
      return c1;
   }
   public String string() // Inputing departure
   {
      String s2="";
      boolean n=true;
      while(n==true)
      {
         try
         {
            s2 = sc.nextLine(); // Taking users input
            n=false; // Stopping While loop next run
         }
         catch(Exception e)
         {
            System.out.println(error.DataType()); // Redirecting to "Error"
         }
      }
      return s2;
   }
   public int integer()
   {
      int i1=0;
      boolean condition=true;
      while(condition==true)
      {
```

```
        try
        {
           i1 = sc.nextInt();
           condition=false;
        }
        catch(Exception e)
        {
           System.out.println(error.DataType());
        }
      }
      return i1;
   }
}
public class Stations
{
   static String station[]=  {"Virar","NalaSopara", "Vasai-
Road","Nalgaon","Bhayander","Mira-Road","Dahisar","Borivali"
      , "Kandivali","Malad","Goregaon","Jogeshwari","Andheri","Vile-Parle","Santacruz",
        "Khar-Road","Bandra","Mahim-Junc","Matunga","Dadar","Elphinston","Lw-
Parel",
        "Mahalaxmi","Central","Grant-Road","Charni-Rd","Marine-Ln","Churchgate"};
   // full forms of all the stations

   static String sh_st[] =
{"vr","ns","va","na","ba","mr","da","bo","ka","ma","go","jo","an","vp",
        "sa","kr","bn","mj","mt","dr","er","lp","mh","mc","gr","ch","ml", "cg"};
   //short forms of all the stations

   static double dist[] =
{55.98,55.85,51.73,47.70,43.11,39.76,36.34,33.98,31.22,29.32,26.90,23.52,21.83,19.57,
17.51,16.29,
        14.66,12.93,11.75,10.17,8.98,7.57,5.95,4.48,3.59,2.21,1.30,0};
   // distance of stations from the main correspondance (VT) - Churchgate

   static Scanner sc = new Scanner (System.in);
   static double distance;
   static double distance1;
   static String from = "";
   static String to = "";
   int pos;
   Settings set = new Settings ();
   public void display()
   {
      int i;
      System.out.println ("Enter the 2 character code corresponding to the station");
      //display for taking short form input from user for the desired station
```

```
   for(i=0; i< station.length; i++) // displaying menu thrugh the formed arrays

   {
      System.out.print( (1+i) + "." + station[i] + ":" + " \t \t \t" + sh_st[i] );
      System.out.println();
   }
}
public static String check(String b)// Finally taking the station input from the user
{
   int pos = 0;
   int j;
   int flag = 0 ;

   for (j=0; j< station.length; j++)
   {
      if (sh_st[j].equalsIgnoreCase(b))
      {
         flag = 1; // Confirming that station is found
         pos = j; // Storing the source
         break;
      }
   }
   distance = dist[pos];
    from = station[j];



   return from; // Returning the station to 'Structure'
}
  public  static String departure (String a)
{   int pos1 = 0;
   int j1;
   int flag1 = 0 ;

   for (j1=0; j1< station.length; j1++)
   {
      if (sh_st[j1].equalsIgnoreCase(a))
      {
         flag1 = 1; // Confirming that station is found
         pos1 = j1; // Storing the destination
         break;
      }
   }
   distance1 = dist[pos1];
   to = station[j1];
   return to;
```

```
   }
}
// error in structure
// error in amount
// in Input, last block comments missing because mostly integer is in this class
public class Settings
{
   public static String boarding_s="vr";
   public void set()
   {
      String boarding_l=""; double d1=0;
      Input input = new Input();
      Stations stations=new Stations();
      Welcome welcome=new Welcome();
      Error error = new Error();
      boolean n =true;//variables initialized to true for while loop
      stations.display();//Displaying options
      while(n ==true)
      {
         System.out.println("Ticket to be issued from...");
         boarding_s = input.string();

         boarding_l = stations.departure(boarding_s);//boarding_s is converted to its long
form and stored in boarding_l
         d1=Stations.distance;
         if(boarding_l=="")
         {
            System.out.println(error.choice());
         }
         else
         {
            System.out.println("Place selected: "+boarding_l);
            n = false;// set to false so that the loop does not continue
         }
      }
      welcome.main();//main menu displayed
   }
}
public class Secondary_info
{
   public String trainclass(char a)
   {
      // Entering full forms for the ticket to be displayed
      if(a=='f')
         return "First class";
      else
```

```
      if(a=='s')
         return "Second class";
      else
         return ""; // Will produce the error statement
   }

   public String returnv (char returnc)
   // Entering full form for the ticket to be displayed
   {
      String returns=""; // redirects to 'error'
      if(returnc=='r')
         returns="Return";
      else
      if(returnc=='n')
         returns="One Way";
      else
         returns="Wrong choice";
      return returns; // returnig values to 'structure'
   }

   public String date ()
   {
      Calendar c = Calendar.getInstance();
      String
date=(c.get(Calendar.DATE))+"/"+(c.get(Calendar.MONTH)+1)+"/"+(c.get(Calendar.YEAR
));
      return date; // storing date for the ticket
   }
}




public class Amount
{
   double amt;
   public double amount(String boarding,String destination,String returns,char trclass,int
adult,int child)//All required data is recieved
   {
      Stations obj = new Stations();
      //initialization of variables:

      double place=0.0,place1=0.0,amountc=0.0,amounta=0.0,temp=0.0,temp2=0.0;
      double rate_c_s=1.125,rate_c_f=2.5,rate_a_s=1.25,rate_a_f=5;//Rates declared
```

```
String t=obj.departure(boarding);
place = Stations.distance1;//boarding place stored
String u = obj.check(destination);
place1=Stations.distance;//destination stored
//Calculating amount
double distance =Math.round(Math.abs(place1-place));//approximate positive
distance calculated
if(trclass=='s')//amount for second class
{
  if(child>0)//Amount for child
  {
    temp=distance*rate_c_s;
    if(temp<2)
    amountc=2*child;
    else
    amountc=child*distance*rate_c_s;
  }
  if(adult>0)//Amount for adult
  {
    temp2=distance*rate_a_s;
    if(temp2<4)
    amounta=4*adult;
    else
    amounta=adult*distance*rate_a_s;
  }
}
else//amount for first class:
{
  if(child>0)//Amount for child
  {
    temp=distance*rate_c_f;
    if(temp<20)
    amountc=20*child;
    else
    amountc=child*distance*rate_c_f;
  }
  if(adult>0)//Amount for adult
  {
    temp2=distance*rate_a_f;
    if(temp2<40)
    amounta=40*adult;
    else
    amounta=adult*distance*rate_a_f;
  }
}
amt=Math.round(amountc+amounta);//Total amount calculated
```

```
      if(returns.equals("Return"))
      amt=amt*2;
      return amt;//amount is passed to function
   }
}
public class Ticket
{
   public void tic(String s1, String s3, String s4, int adult, int child, double price , String
date, String returns  )
   {
      Stations obj = new Stations();

System.out.println("\n\n_____________________________________________\n******
**********************************\n      MUMBAI SUBURBAN RAILWAYS
\n*****************************************");
      System.out.print("Date:"+date+
         "\nRs."+price+"\n"+s3+"\t"+returns+"\n"+//price,class and return or not
         "From: "+ obj.to+"  \tDest: "+obj.from +"\n"+
         "Adult: "+adult+"\tChild: "+child+"\n\n"+
         "---------------HAPPY JOURNEY----------------\n"+
         "_____________THANK YOU FOR JOINING US_________\n\n\n"); // Printing the
final ticket

   }
}
public class Exit
{
   public void exit () // Prints the exit page once the user selects "x" to stop the program
from running
   {

System.out.println("**********************************************************
**********************************************************************
**********************************************************");
      System.out.println("-----------------------------------------------------------------------Thank
you for booking with us-----------------------------------------------------------------------");

System.out.println("**********************************************************
**********************************************************************
**********************************************************");
      System.exit(0);
   }
}

//This class provides with the error messages
public class Error
```

```
{
   public String choice()
   {
      return "Invalid choice!\n"; // Providing Error for welcome decision
   }
   public String character()
   {
      return "Invalid character!"; // Providing Error for invalid character
   }
   public String DataType()
   {
      return "Invalid data type!Please re=enter"; // Providing Error for wrong datatype
   }
   public String Format()
   {
      return "Invalid format!Please re-enter"; // Providing Error for choice in Adults and
Children
   }
}
```

TESTING

Testing is a set of activities that can be planned in advance and conducted systematically. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

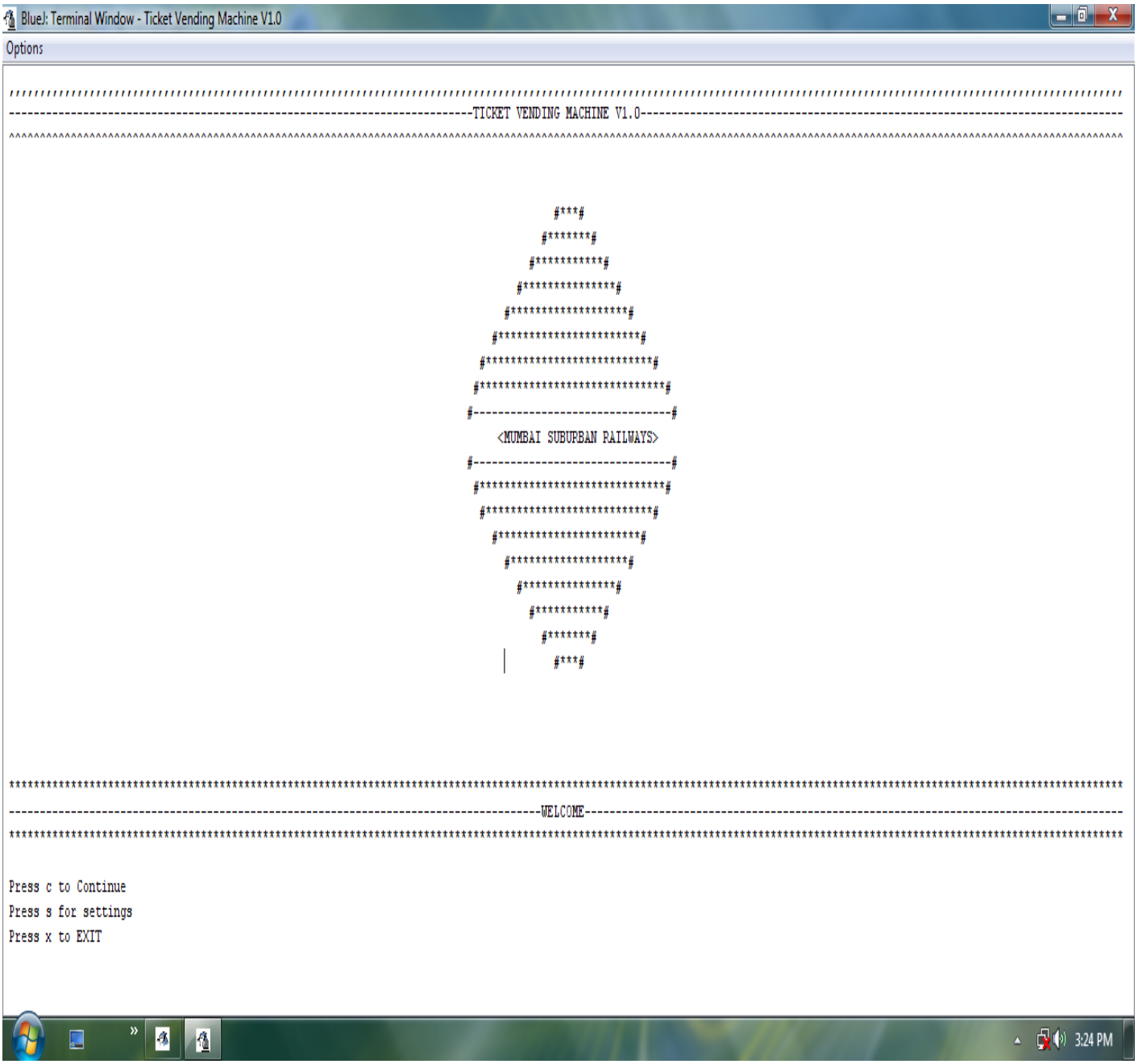
The main motive behind performing a series of tests is to evaluate the functionality of the software with intent to find whether the developed software meets the specified requirements of the customer, or not and to identify any unknown bugs or errors which might have been left unidentified during the execution; to ensure that the final software is defect free so that it does not harm the working environment/ businesses of the Indian Railways.

➢ Unit Testing

Testing begins at the module level and works outward towards the integration of the entire computer based system. Different testing techniques are appropriate at different level of time. Testing & debugging are different activities, but debugging must be accommodated in any testing strategy.

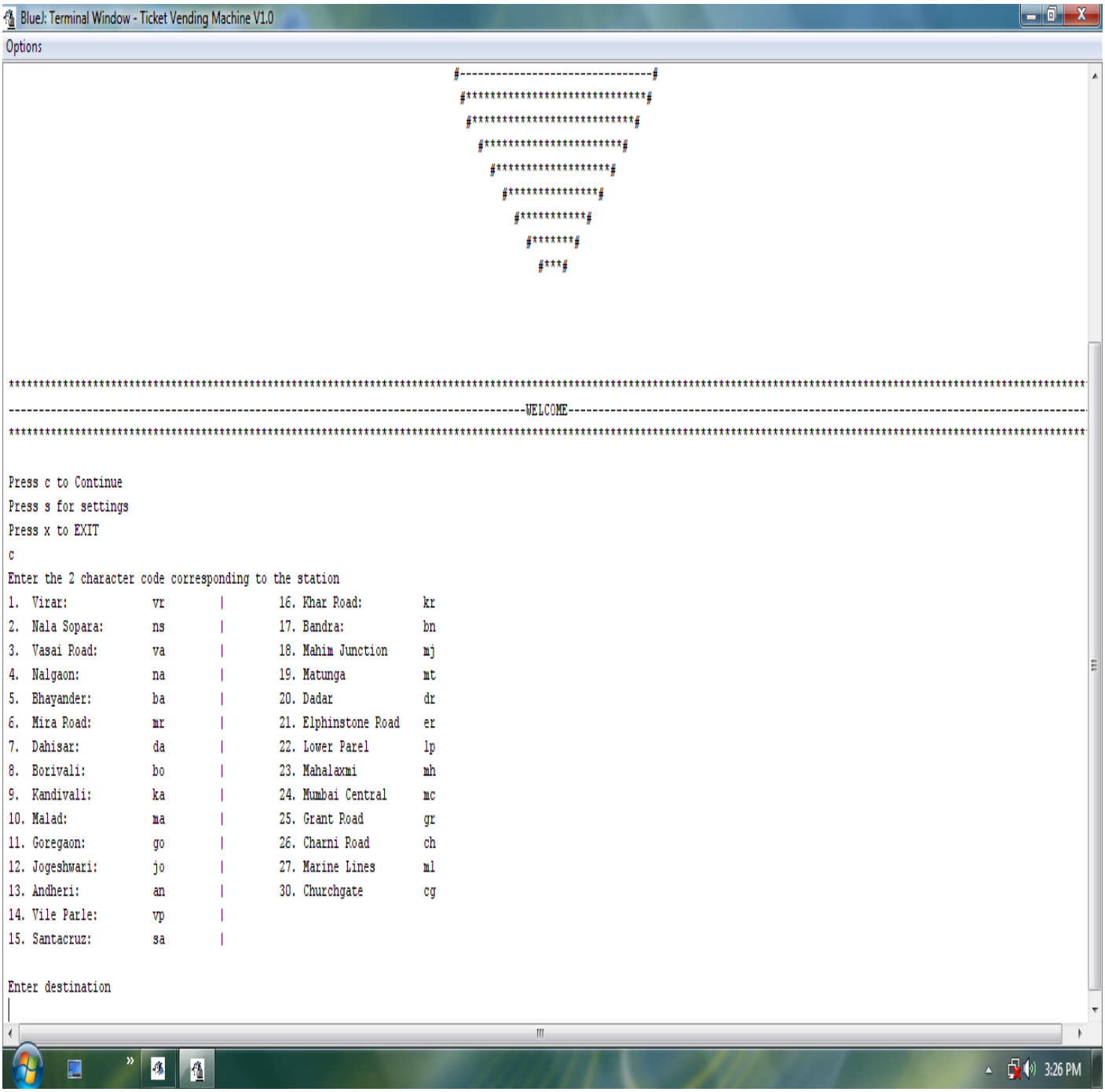
Unit testing is done on individual components of the software, where each section of the code is isolated in order to verify its correctness. It is undertaken when a module has been created and reviewed. It is an efficient way to fix bugs in the early development cycle, hence, saving costs. The purpose is to validate that each unit of the software performs as designed.

Initial window:



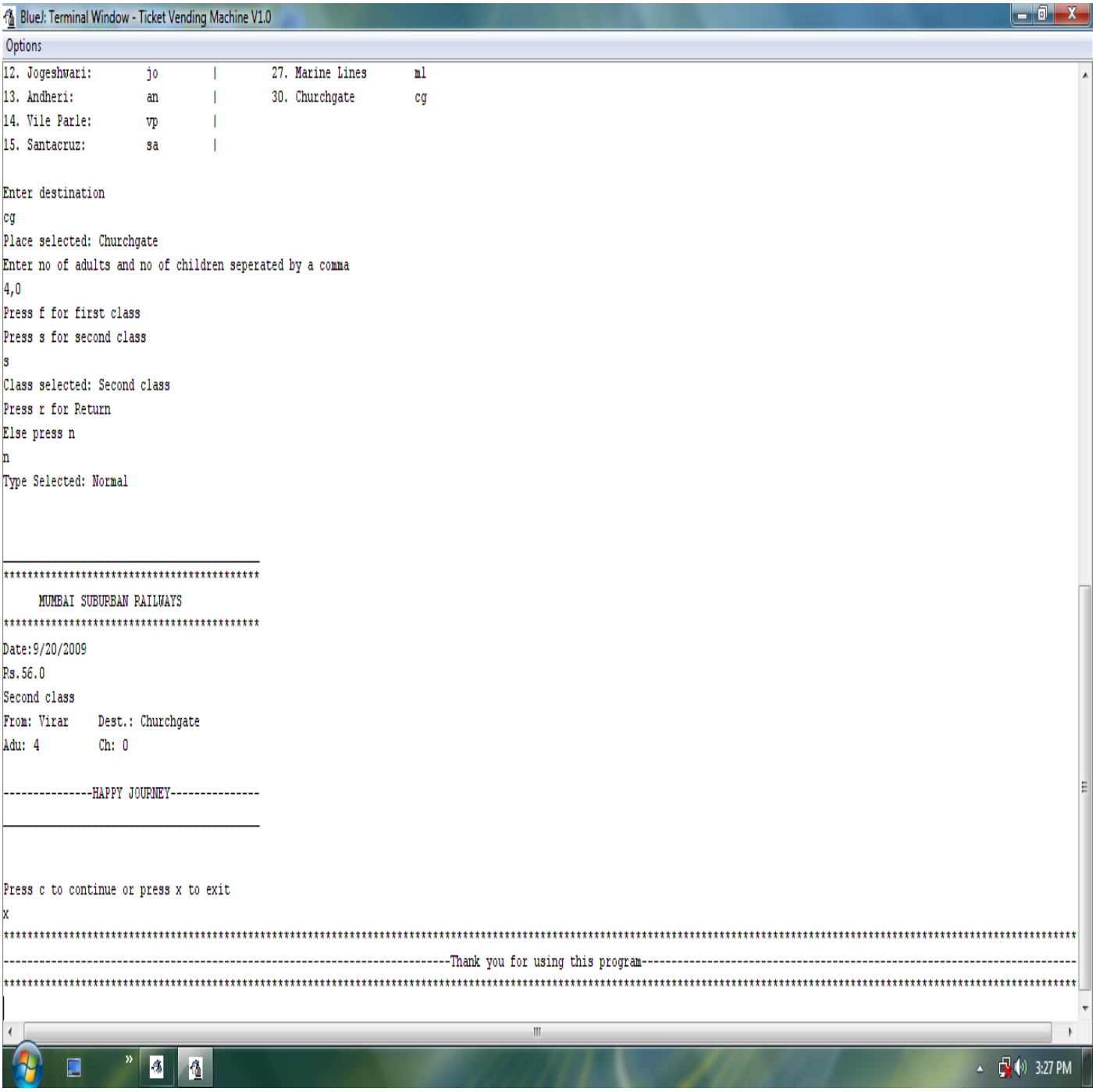
This is the main screen which one sees on accessing the main function of Welcome class. Three options are displayed:
c to continue, s for settings and x to exit.

When ‘c’ is entered:



When ‘c’ is entered, the list of stations is displayed as shown above. Destination is asked.

After entering required details:

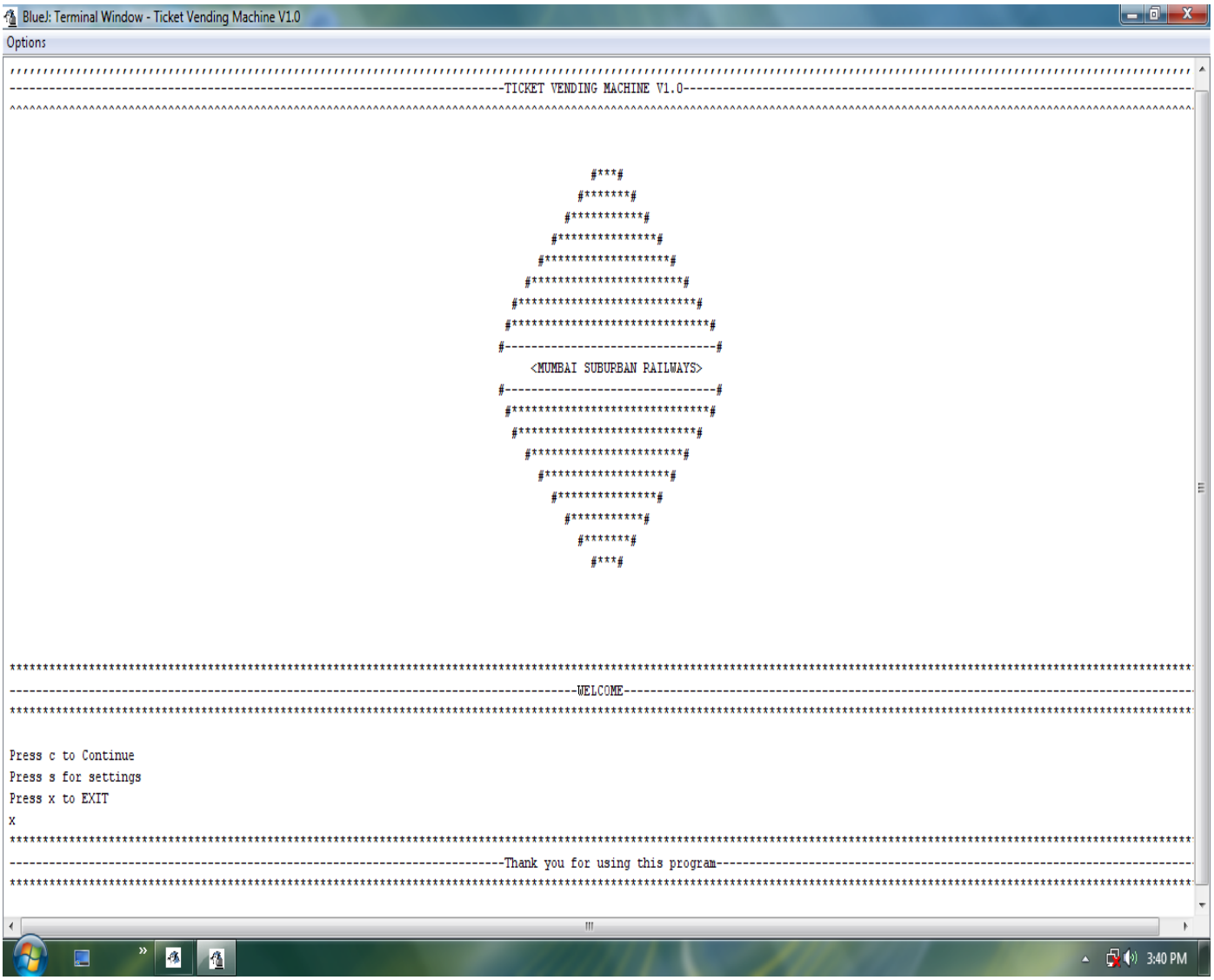


When the required details like Destination, number of adults, number of children, class and return/normal is entered as shown above. The ticket is displayed accordingly and amount is calculated depending on the distance.

If ‘s’ was entered instead of ‘c’:

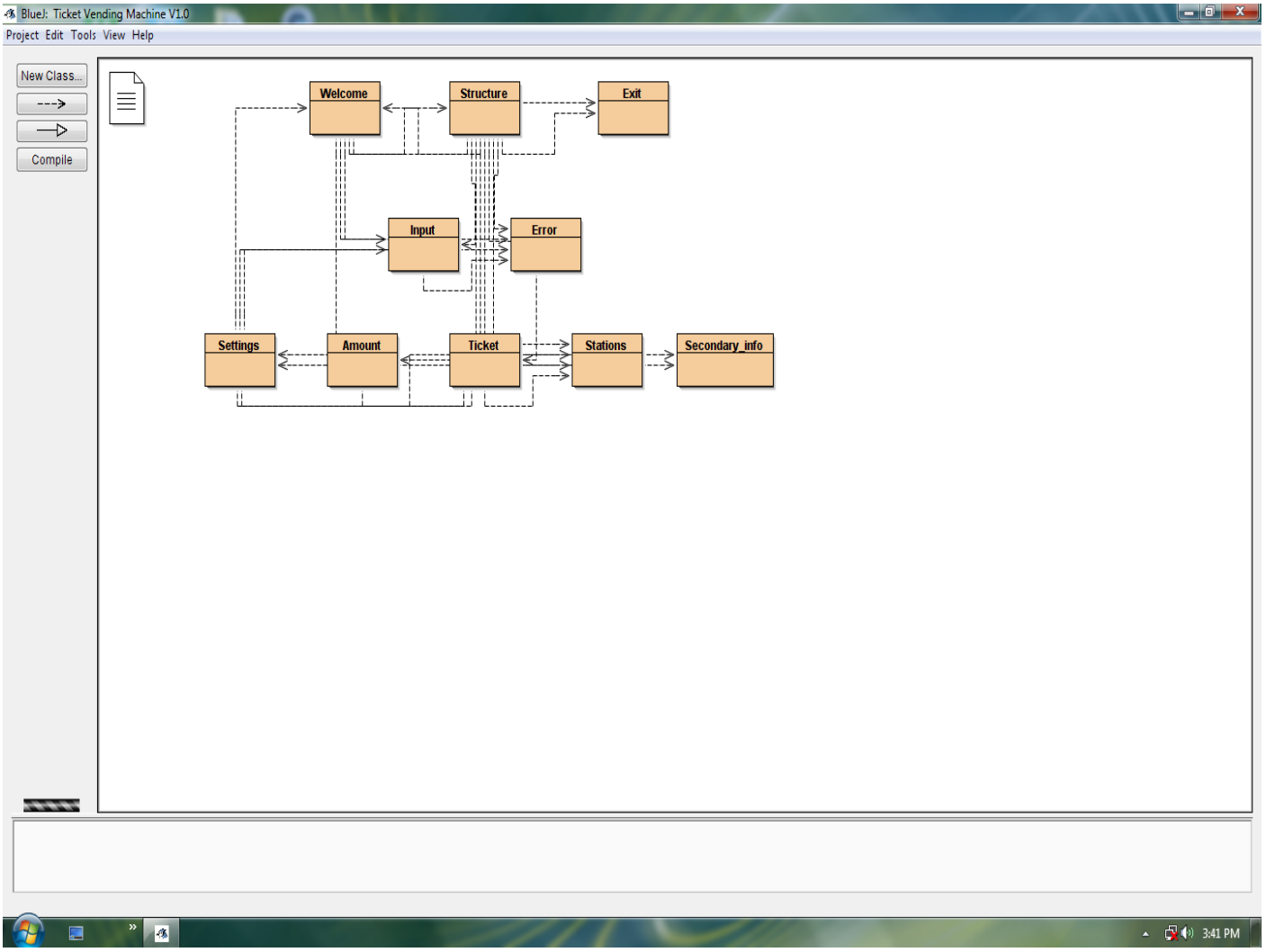
Boarding station is asked. On entering the code for the required station, the three options are asked once again.

If ‘x’ was entered instead of ‘s’ or ‘c’:



The program quits displaying “Thank you for using this program”

BlueJ screen:



All various classes are shown.

IMPLEMENTATION

Implementation refers to how well a system is put into practice and is fundamental to establishing the internal, external, construct and statistical conclusion of outcome evaluations.

Implementation is yet another intensive task which is as important as the whole program itself. It is very important that the new software is accurately implemented which will beneficial for timely roll out of the software.

- **Parallel Implementation**

It has been recommended that it is safer to implement a staged rollout of the new software in order to minimize the risk of any catastrophic event which might occur due to unidentified bugs in the software. Thereby, a failure in the system will in fact have a minimum impact. The existing practice of manually managing seats, movies and customer information, should run concurrently with the new software.

The basic objective of this activity is to ensure stability of the new system, enable the users to become comfortable with the new processes and to develop confidence leading to complete switch over. However, the parallel stage places tremendous pressures on resources, time constraints and co-operation by the Railways personnel. The Indian Railways can therefore compare the output of the old system with the output of the new system to ensure effective implementation of the new software.

Post Implementation Review

A post implementation review will be done after the completion of the coding and testing part. After the system is implemented and conversion is completed, a review of the system is generally conducted by the users and analysts in order to ensure the system has met its objectives and the requirements of the company.

MAINTENANCE

Our belief is that not a single project is ever considered as complete forever because our mind is always thinking something new and our necessities are growing day by day. Big ideas carry endless possibilities, and incorporating them is not a day's task. We always want something more than what we have.

Software enhancement is a vast activity which includes optimization, error correction and deletion of discarded features and augmentation of existing features. Since the software is fairly new and apart from various thorough beta testing, it might happen that at some point of time, the system might encounter some unexpected bugs. In any case, the program can be modified post-delivery to correct faults, improve performance or other attributes.

The entire project has been developed and deployed as per the requirements stated by the user, it is found to be bug free as per the testing standards that is implemented. Any specification-untraced errors will be concentrated in the coming versions, which are planned to be developed in near future.
Changes in the working environment sometimes require modifications in the software. However, additional functionality enhancements can further be done to incorporate new requirements from the client, post-delivery. The software is well optimized to incorporate additional features which will further amplify the functionality of the software.