MAXIMUM OPTIMIZED UTILISATION OF FLIGHT.

GUIDED BY:

Prof. Manjula R

SUBMITTED BY:

NAME	REG NO
Aditya Tiwari	16BCE0096
Tirumaredy Snehil	16BCE0193
Adway Ujjwal	16BCE0699

CONTENTS

Chapter – 1

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

1. Introduction

Document purpose

Product scope

Acknowledgement

Overview

2. Overall Description

Product perspective

Product functionality

Users and characteristics

Operating Environment

Assumptions and basic Dependencies

3. Specific Requirements

Use Case Diagram

Functional requirements

Usability

Benefits

1. INTRODUCTION

With the hard competitive environment of air transportation and fast growing world, airlines have to cope up with more and more complex and large optimization problems at planning and operations level, especially with the flight scheduling.

One of the major tasks consists in assigning portable aircraft type to each flight, given the set of flights.

The project has to meet a large variety of requirements with a special priority given to maximize the number of passengers boarding in every single flight and has deal with complementary objective of maximizing profit per flight and minimizing costs (such as fuel, maintenance operation, …) .

DOCUMENT PURPOSE

If we look into the flight or an air lines business, their main priority is to maximize the passenger capacity for a particular flight between any to airports so that no seat is left unoccupied and they don't suffer with any losses.

Taking into concern if any of the seat is left vacant or unoccupied the business suffer with the following consequences:

1. They will suffer with the loss made by the unoccupied seats.
2. The number of trips will be increased in order to drop the passenger with other time schedule.
3. Thus it will lead to more consumption of fuel which in turn cost a lot to the air line company.

PRODUCT SCOPE

Scope of this project is to replace the manual work of flight scheduling with a new advanced computerized system.

User does not need much training to use this software, as this software is very user friendly and easy to use. It replaces all the paper work also.

In this software we can store thousands of records. It replaces all the calculation works also as it automatically calculate and print the bill. Also, customer numbers are generated automatically by system itself.

ACKNOWLEDGEMENT

We would like to thank our teacher **Mrs. Manjula R** for her constant support and guidance that enabled us to make a wonderful project on “Flight Scheduling System**”**.

We learnt a lot during the project and it would not have been possible without the support system provided by VIT authorities .It was due to the mutual understanding, support, interest, cooperation and hard work of all the group members that enabled us to complete this project.

OVERVIEW

Thus to taking air lines business into concern company should take the passenger maximizing technique into their main priority. The company can make sure that the flight is travelling with their maximum capacity between the routes by using following methods:

1. Passengers should be rescheduled to another flights having seat vacancies and dropping the passengers to their destination or to another destination linked with another flight with maximum capacity which drop the passenger to their particular destination.

2. A list of the flights travelling between any two station must be made along with the combinations to reach that destination for example if there are 3 stations A,B and C the list of routes for travelling from A to B contain following number of routes A to B and, A to C then to B.

3. The list of seat vacancies must be made so to place the passengers and thus maximizing the capacity of a flight for a particular trip.

2. ***OVERALL DESCRIPTION***

PRODUCT PERSPECTIVE

We will be given with the number of airport we'll be handling accordingly the number of flights will be decided with the number of passengers traveling in all flights. We have to find a way to shift the passenger from the flight carrying minimum number of passenger so as to avoid the loss. We'll be focusing on filling more than 75% seats of the flight.

At the end, we output should be carrying more number of flights having passengers full to max or above 75%.

PRODUCT FUNCTIONALITY

Step 1) Input – Take the number of airport as 'n';

Step 2) Input – Take the details of n*n-n planes details

Step 3) Select the plane carrying minimum number of passenger from each airport. Minimum condition is minimum among the full airport + less than 135.

Note their number in a array “list”.

Step 4) Select the minimum from the array “list”.

Step 5) Fill the table.

Vacancy =180 – passenger of plane going to same airport from other airport.

Possible shift = 180 - passenger of plane going from same airport to other airport going to same destination.

Shift = minimum of vacancy and possible shift

Step 6) Update the shift column and priorities the shift column in ascending order.

Step 7) Start updating the number of passenger in plane selected according to shift.

Step 8) Fill the “depend” and “determine” list of each plane.

Step 9) Go back to step 3. Continue up till no more further shifting possible. Clue shift column will be like 0,0,0,0,0,…… .

Step 10) Fill the timing of the plane.

Step 11) automatically update the new timing to existing old timings if new>old.

USERS AND CHARACTERISTICS:

Customers: The Person who will use online flight optimization system for scheduling his/her flight by the use of this software. Will register and login through the login page

- Will optimize their aircrafts through webpage.
- Can choose an option of forgot password and reset it
- Can wish to save their flight details
- Can contact the customer care if some problem arises

Admin: Person who will add new user in the system. Add new categories in the online flight optimization system and change the plans of the all airline companies.

Adds a new user to the list

- Adds all the airline companies and alter them
- Adds all the plans for various service providers
- Take care of any problem that comes in any transaction

OPERATING ENVIRONMENT

Software Requirements:

Front end: Made by using html and css on Notepad++ (Dreamweaver is also an option)

Back end: Made on Mysql Front end and back end linked by using php either on Notepad ++ or
Netbeans

Code: turbo c

Hardware Requirements:

Processor: Pentium3 and above.

Ram: 1GB or higher.

Hard disk: 20GB

Main requirement is of a web browser that can load the platform

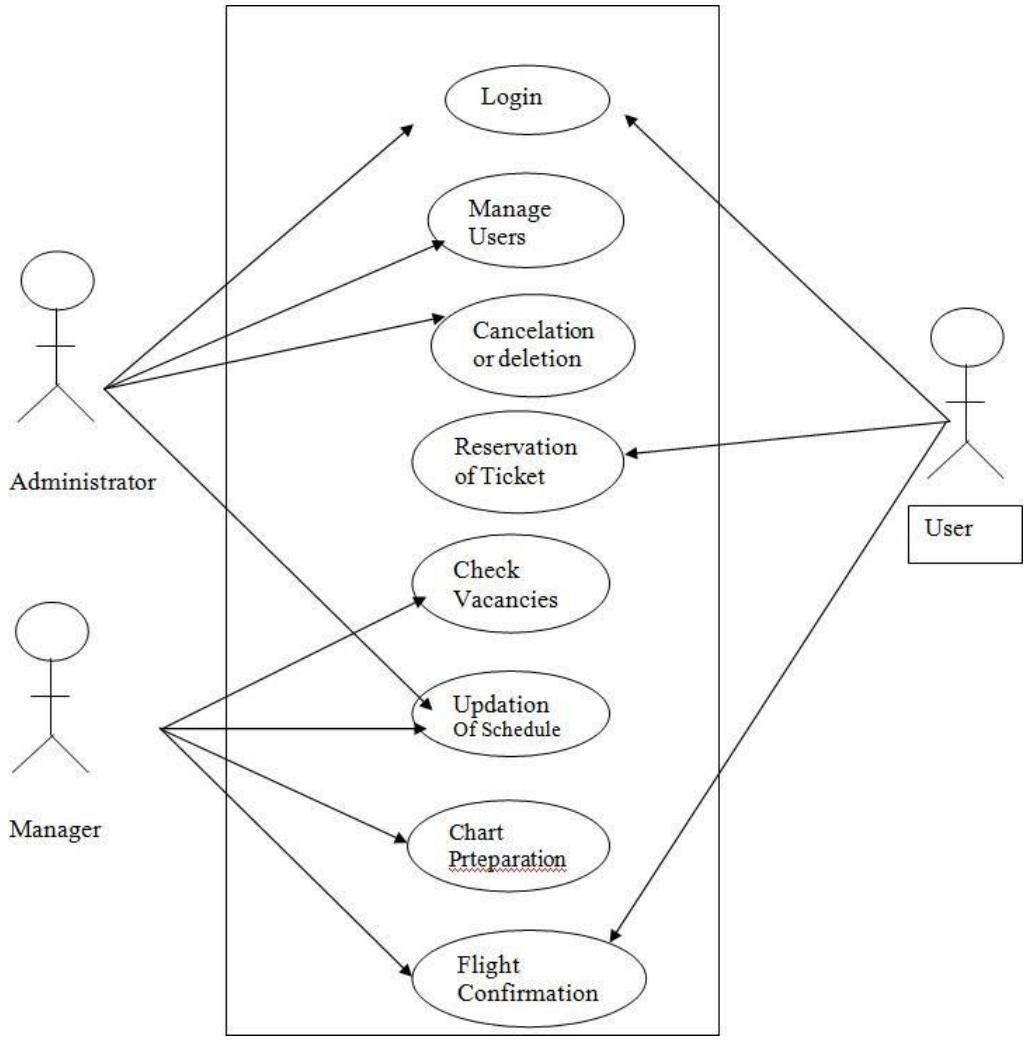
ASSUMPTIONS AND BASIC DEPENDENCIES

1. User has basic knowledge of computers.
2. User should have sufficient knowledge of English since the system interface will be in English.
3. The system is fast.
4. Internet connection is available to all the users who use this subsystem.
5. The user is assumed to give system correct information about his details and also about the symptoms.

6. The system will have simple and easy to use interfaces.
7. Provides accurate data.

3. SPECIFIC REQUIREMENTS

USE CASE DIAGRAM



FUNCTIONAL REQUIREMENT

For the proper management of the project, we have used open source software management tool ProjectLibre. Different tasks are being assigned to each person and are managed efficiently using this tool.

Delivery - D:\Projects\SE\Delivery.pod *

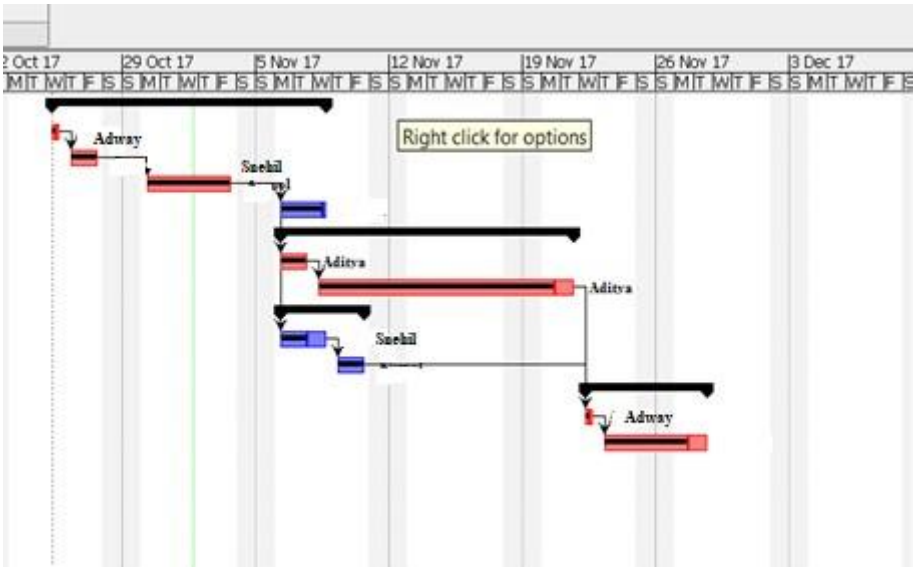
ProjectLibre OPENPROJ

File | Task | Resource | View

Network | Zoom In | WBS | Zoom Out | Gantt | Task Usage — Views
Copy | Cut | Paste — Clipboard
Insert | Indent | Link | Delete | Outdent | Unlink | Information | Assign Resources | Find | Calendar | Save Baseline | Scroll To Task | Notes | Clear Baseline | Update — Task

		Name	Duration	Actual D...	Start	Actual Start	Finish	Actual Finish	Predecessors	Resource Names
1		**Research & Developr**	11 days	**10 days**	**10/25/17 8:00 AM**	**10/25/17 8:00 AM**	**11/8/17 5:00 PM**			
2		Brainstorming Ideas	1 day	0.5 days	10/25/17 8:00 AM	10/25/17 8:00 AM	10/25/17 5:00 PM			Anmol
3	✔	Determine Product	2 days	2 days	10/26/17 8:00 AM	10/26/17 8:00 AM	10/27/17 5:00 PM	10/27/17 5:00 PM	2	Anmol
4	✔	Design Product	5 days	5 days	10/30/17 8:00 AM	10/30/17 8:00 AM	11/3/17 5:00 PM	11/3/17 5:00 PM	3	Anmol
5		Test and Fix	3 days	2.5 days	11/6/17 8:00 AM	11/6/17 8:00 AM	11/8/17 5:00 PM		4	Anmol
6		**Coding**	12 days	**11 days**	**11/6/17 8:00 AM**	**11/8/17 8:00 AM**	**11/21/17 5:00 PM**			
7	✔	Code	2 days	2 days	11/6/17 8:00 AM	11/6/17 8:00 AM	11/7/17 5:00 PM	11/7/17 5:00 PM	4	Aditya
8		Manufacture	10 days	9 days	11/8/17 8:00 AM	11/8/17 8:00 AM	11/21/17 5:00 PM		7	Aditya
9		**Testing**	5 days	**4 days**	**11/6/17 8:00 AM**	**11/6/17 8:00 AM**	**11/10/17 5:00 PM**			
10		Test cases	3 days	2 days	11/6/17 8:00 AM	11/6/17 8:00 AM	11/8/17 5:00 PM		4	Komal
11	✔	Manufacture initial pro	2 days	2 days	11/9/17 8:00 AM	11/9/17 8:00 AM	11/10/17 5:00 PM	11/10/17 5:00 PM	10	Komal
12		**Implementation**	5 days	**3.5 days**	**11/22/17 8:00 AM**	**11/22/17 8:00 AM**	**11/28/17 5:00 PM**			
13		Packing	1 day	0.5 days	11/22/17 8:00 AM	11/22/17 8:00 AM	11/22/17 5:00 PM		11;8	Anmol
14		Shipping	4 days	3 days	11/23/17 8:00 AM	11/23/17 8:00 AM	11/28/17 5:00 PM		13	Anmol

GANNT CHART:



USABILITY

1. The interface is simple and easy to use.
2. System is user friendly, self-explanatory and also it is provided with a help guide.
3. It can be used easily by anyone from a kid of 10 years to an elderly person of 80 years
4. keeping in mind that they have a basic knowledge of computers.

BENEFITS

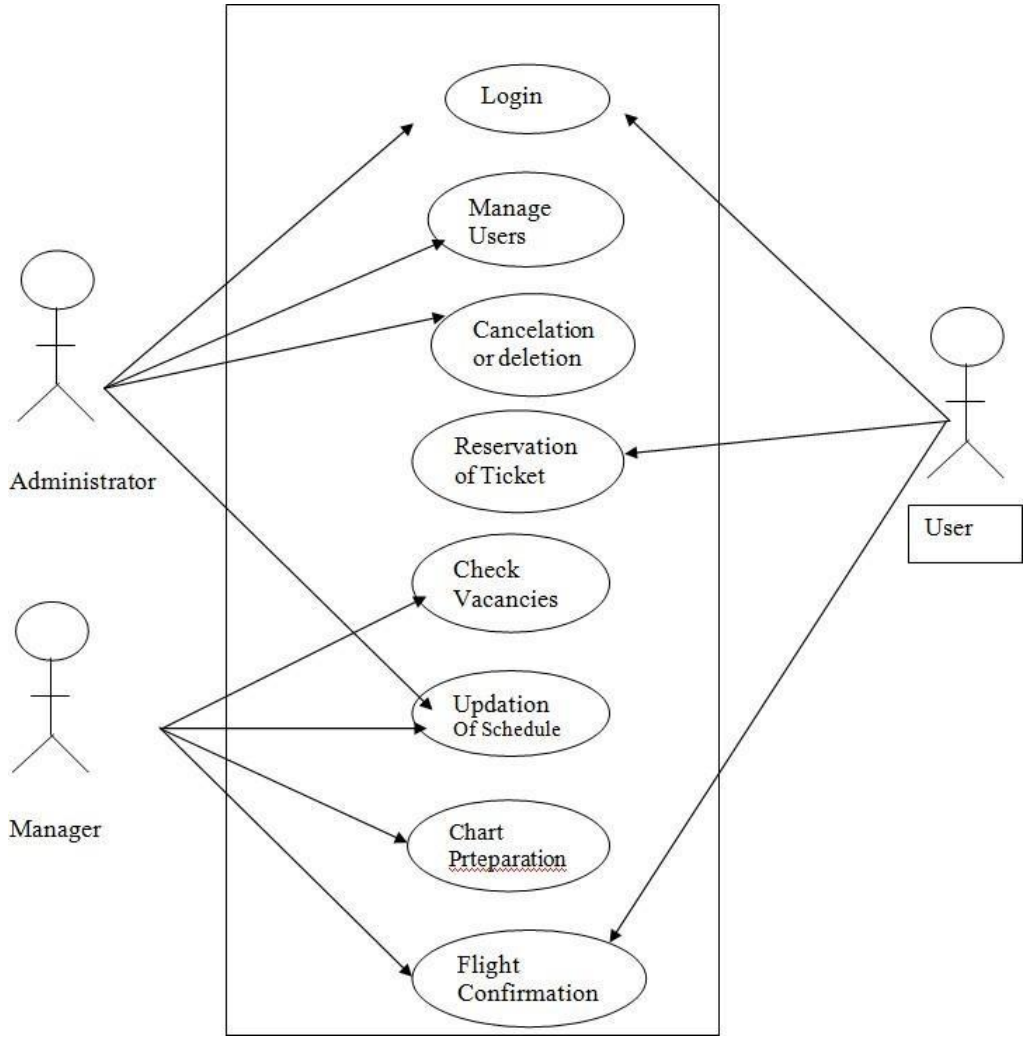
1. No flight with go vacant with this software as the passengers will be shifted to the flight with less no of passengers
2. No loss to the airline company as all flights will hav minimum of 75% allocated seats as a result the prices of flight ticket will lower down.
3. The passengers won't have to wait for long.
4. Customer Satisfaction will be full.

CHAPTER-2

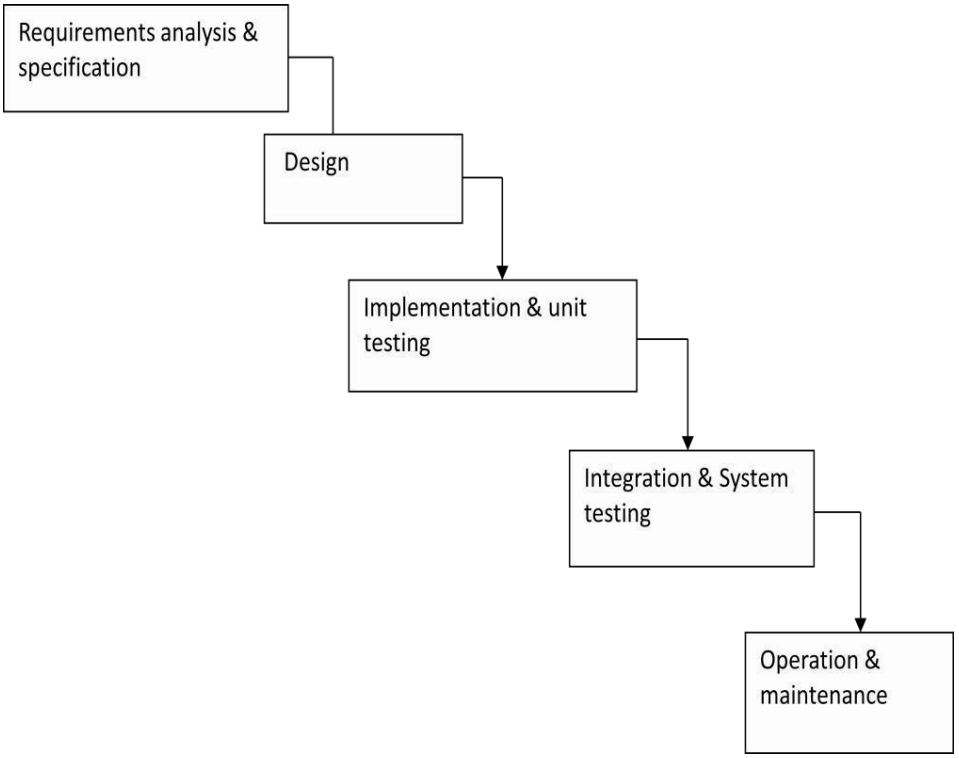
SOFTWARE DESIGN SPECIFICATION

UML DIAGRAM

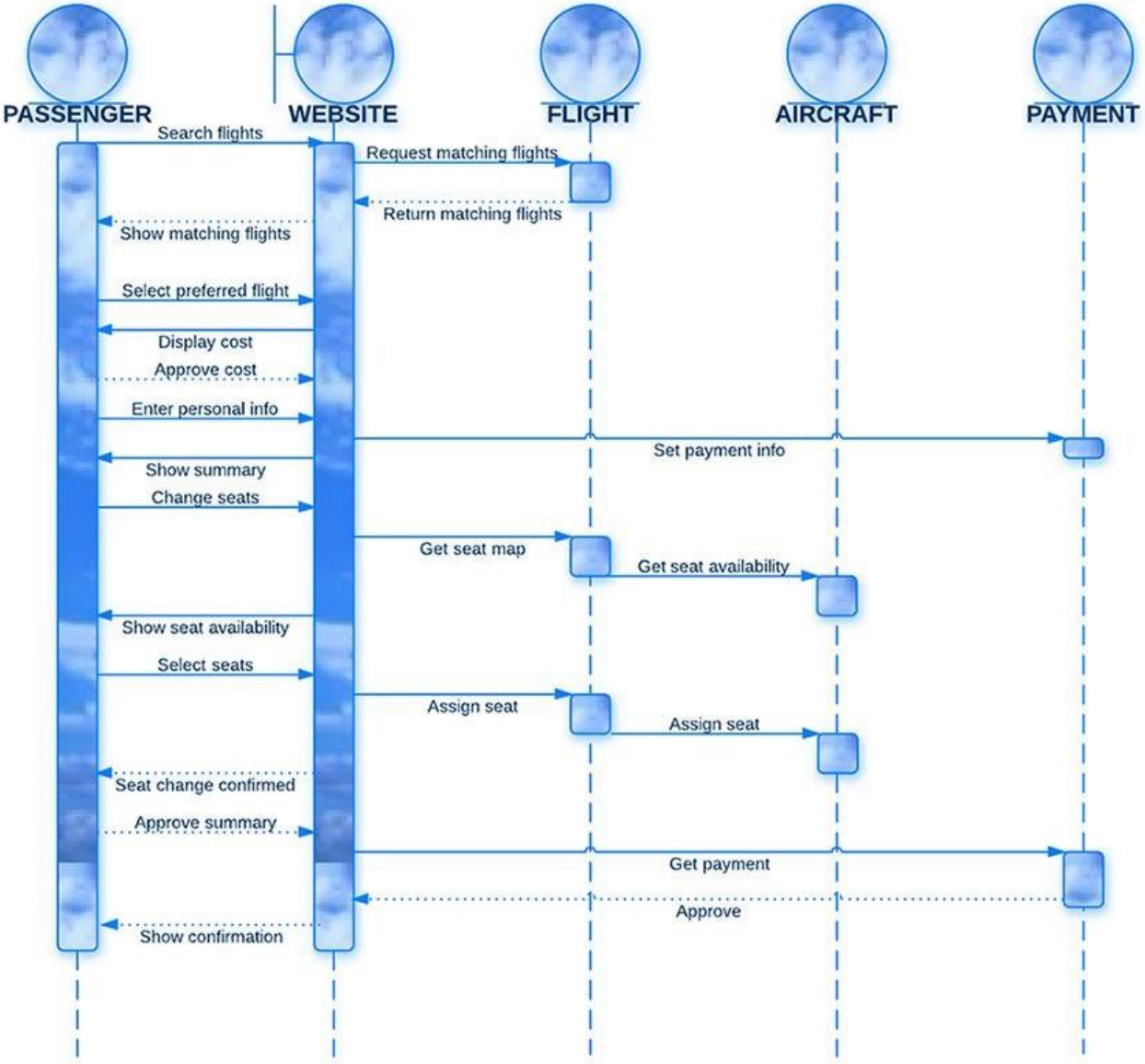
1. USE CASE DIAGRAM



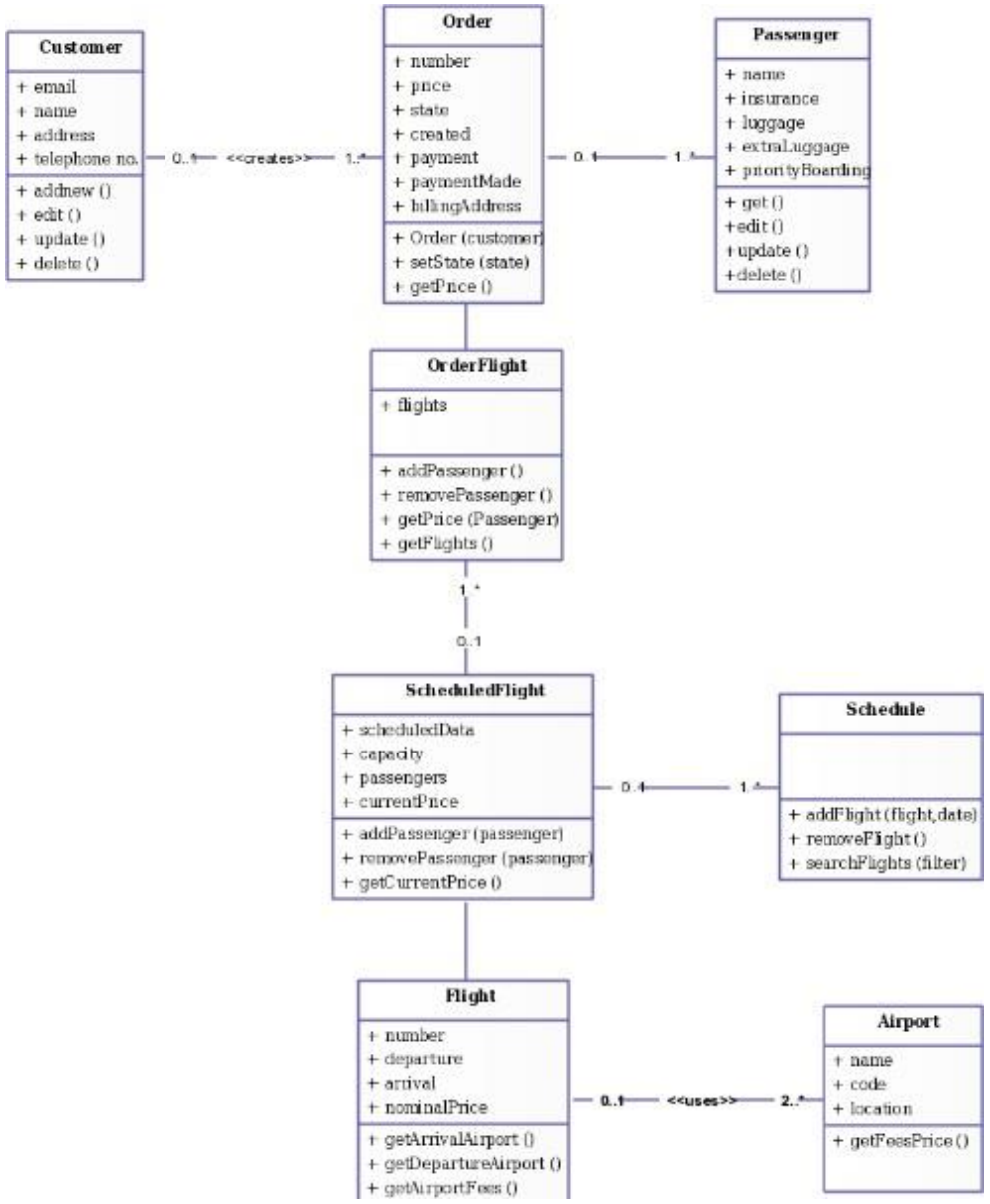
2. ACTIVITY DIAGRAM



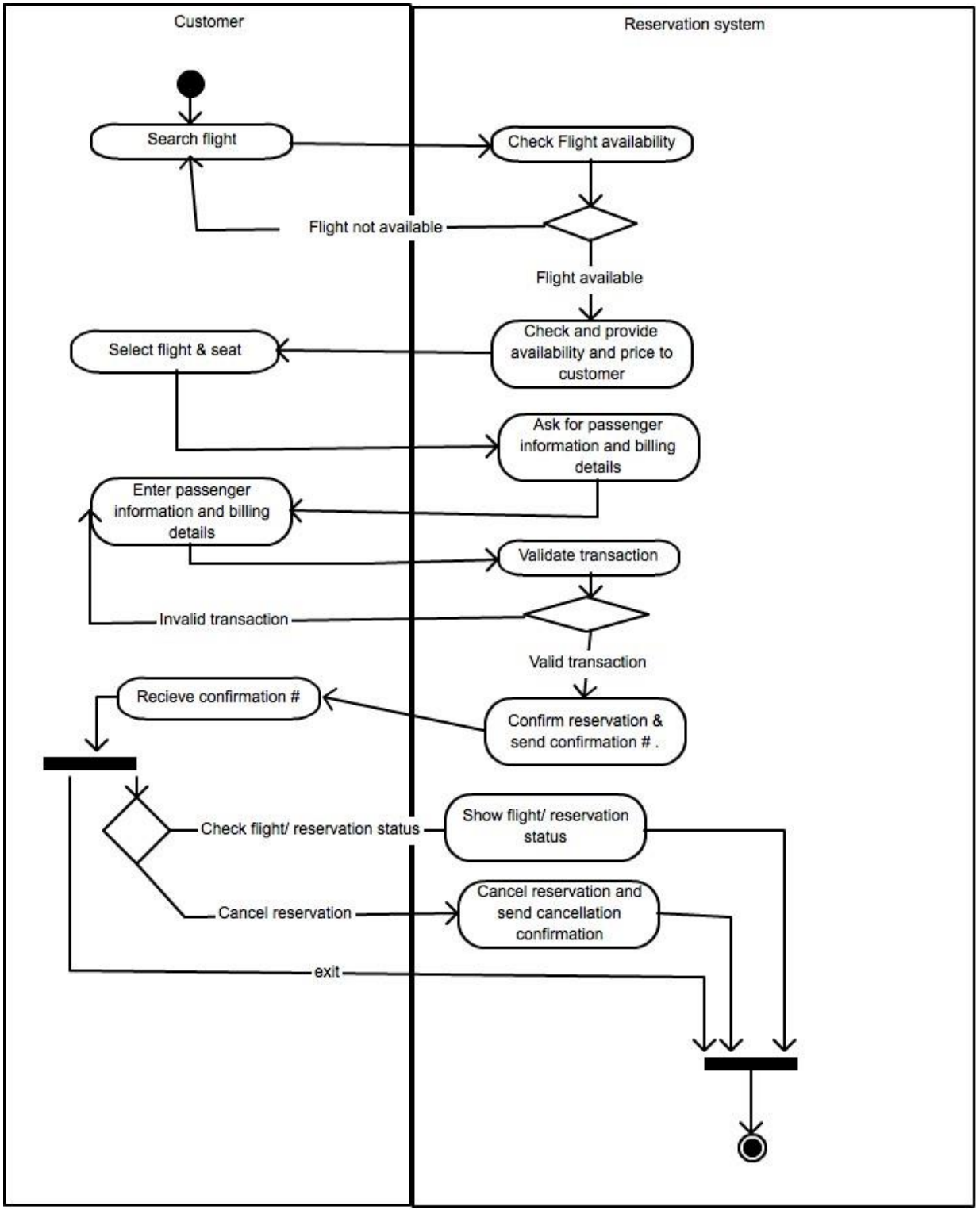
3. SEQUENCE DIAGRAM



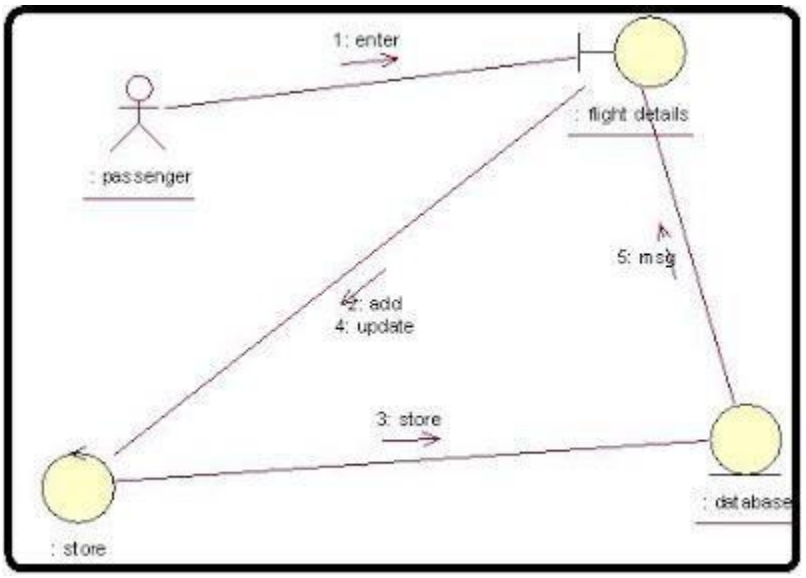
4. CLASS DIAGRAM



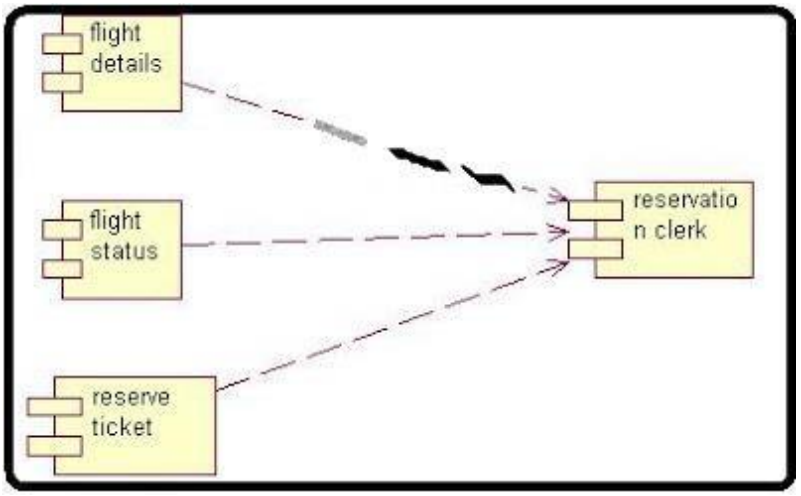
5. ACTIVITY DIAGRAM



6. COLLABORATION DIAGRAM:



7. COMPONENT DIAGRAM:



CHAPTER-3

IMPLEMENTATION (PROTOTYPE)

Total number of airports to be planned = n

Name of airports stored in A[i]

Time taken to travel between airports = t[n(n-1)].

Total number of aircrafts to be scheduled will be

Input the passengers from each airport to other airport conditions 180 max.

TOTAL NUMBER OF AIRPORTS	N

struct Airport
{
name[20];
mini;
}airport[5];
 	
NAME OF ith AIRPORTS
	void airportinput(int n)
	{
	for(int i=1;i<=n;i++)
	{
	cout<<"\n Enter Airport name: ";
	cin>>Airport[i].name;
	}
	}

struct plane
{
code;
current airport;
destination[20];
passenger;
depend[5];
determine[5];
}plane[25];
	void flightinput(int n)
	{
	int x=n*n,i,j,k;
	for(i=1,k=1,j=1;j<=x;j++,k++)
	{
	cout<<"\n Enter flight code: ";
	flight[i].code=i;
	
	flight[j].current_airport=Airport[i].name;
	
	flight[j].destination=Airport[k].name;

	if(k==n)
	{
	k=0;
	i++;
	}
	
	if(flight[j].destination==flight[j].current_airport)
	{
	flight[j].passenger=0;
	}
	else
	{
	cout<<"\n Enter total number of passengers: ";
	cin>>flight[j].passenger;
	}
	}
	}

PROCESSING-

SELECTING THE PLANE CARRYING LEAST NUMBER OF PASSENGERS FROM EACH AIRPORT	void minimum(int n)
	{
	inti,j,k;
	for(i=1,k=1;i<=n;i++,k++)
	{
	Airport[i].mini=135;
	for(j=1+(i-1)*n;j<1+(i)*n;j++)
	{
	if(flight[j].passenger!=0)
	{
	if(Airport[i].mini>flight[j].passenger && flight[j].passenger<135)
	{
	
	Airport[i].mini=flight[j].passenger;
	list[k]=j;
	}
	}
	}
	}
	}

SELECT THE MINIMUM OF THE LIST	void select(int n)
	{
	int i;
	for(i=1;i<=n;i++)
	{
	if(flight[list[i]].passenger !=0)
	m=list[i];
	}
	for(i=1;i<=n;i++)
	{

```
        if(flight[m].passenger>flight[list[i]].passenger
&& flight[list[i]].passenger !=0)
                {
                        m=list[i];
                }
        }
        cout<<m;
}
x=plane[m].passenger;
```

FILL THE SAMPLE TABLE

	VACANCY	POSSIBLE SHIFT	SHIFT	
B->E	100	18	15	
C->E	20	37	20	
D->E	80	60	60	

```
int p[4][6-2];
void calctable(int n)
{
        inti,k,j,row,col,f=0;
        int x=n*n;
        for(i=1;i<=n && f==0;i++)
        {
                col=0;
                for(j=1+(i-1)*n;j<1+(i)*n && f==0;j++)
                {
                        col++;
                        if(j==m)
                        {
                                row=i;
                                f=1;
                        }
                }
        }

        j=col;
        for(i=1,k=1;j<=x && i<=n && k<=n-
2;i+ ,j+=n)
        {
           if(j!=i+(i-1)*n && j!=m)
                        {
                                p[1][k]=180-
flight[j].passenger;
                                k++;
                        }
   }
        cout<<endl<<"Row"<<row;
        j=1+(row-1)*n;
        for(i=1,k=1;j<1+(row)*n && i<=n && k<=n-
2;i++,j++)
        {
                        if(j!=i+(i-1)*n && j!=m)
                        {
                                p[2][k]=180-
flight[j].passenger;
                                k++;
                        }
        }

        int sum=0;
```

	for(i=1;i<=n-2;i++)
	{
	if(p[1][i]>p[2][i])
	{
	p[3][i]=p[2][i];
	}
	else
	{
	p[3][i]=p[1][i];
	}
	sum+=p[3][i];
	}
	for(i=1;i<=3;i++)
	{
	for(j=1;j<=n-2;j++)
	{
	cout<<endl<<p[i][j];
	}
	cout<<endl;
	}
	updatep(n,sum);
	}

UPDATE THE P TABLE IF SUM>NUMBER OF PASSENGERS TO BE SHIFTED	void updatep(intn,int sum)
	{
	inti,j;
	int z[4];
	x=flight[m].passenger;
	for(i=1;i<=n-2;i++)
	{
	z[i]=p[3][i];
	}
	quickSort(z, 1, 3);
	printf("Sorted array: \n");
	printArray(z, 3);
	for(i=1;i<n-2;i++)
	{
	for(j=1;j<=n-2;j++)
	{
	if(z[i]==p[3][i])
	{
	if(x<=p[3][i])
	{
	p[3][i]=x;
	x=0;
	}
	else
	{
	x=x-p[3][i];
	p[3][i]=0;
	if(x<=0)
	{
	x=0;
	}
	}

	}
	}
	}
	for(i=1;i<=3;i++)
	{
	for(j=1;j<=n-2;j++)
	{
	cout<<endl<<p[i][j];
	}
	cout<<endl;
	}
	
	}

UPDATE THE PLANE PASSENGER STATUS AFTER SHIFT	void update(int n)
	{
	inti,k,j,row,col,f=0;
	int x=n*n;
	for(i=1;i<=n && f==0;i++)
	{
	col=0;
	for(j=1+(i-1)*n;j<1+(i)*n && f==0;j++)
	{
	col++;
	if(j==m)
	{
	row=i;
	f=1;
	}
	}
	}
	
	j=col;
	for(i=1,k=1;j<=x && i<=n && k<=n-2;i++,j+=n)
	{
	if(j!=i+(i-1)*n && j!=m)
	{
	
	flight[j].passenger+p[3][k];
	k++;
	}
	}
	cout<<endl<<"Row"<<row;
	j=1+(row-1)*n;
	for(i=1,k=1;j<1+(row)*n && i<=n && k<=n-2;i++,j++)
	{
	if(j!=i+(i-1)*n && j!=m)
	{
	
	flight[j].passenger+=p[3][k];
	k++;
	}
	}
	}

	Back to step SELECTING THE PLANE CARRYING LEAST NUMBER OF PASSENGERS FROM EACH AIRPORT

CHAPTER-4

DESIGN OF TEST CASES

Test Cases are nothing but a set of various inputs for which we check the Output of
the given Software. We usually provide sets of test cases and note the Output generated. If the Software show any ERROR in Output than the note those input conditions and ask the Software developer to look at Code once again and make the necessary changes.

1. Verify that we can add new flights in the sytem.
2. Verify that on filling flight details like flight name, code, from and to destinations, capacity, timings and frequency etc, new filghts get successfully added in the system.
3. Verify that user can search for flights by name, from-to airports or flight code for checking their status and timings.
4. Verify that search results have seat details and availability.
5. Verify that clicking the search results open complete details for flight.
6. Verify that user should see realtime flight status of availability of seats.
7. Verify that user is presented with graphical view of the airline's sitting arrangement along with seat number and availability status.
8. Verify that pricing of different types of seats is displayed to the users.

9. Verify that user can successfuly select single or more than one seat.
10. Verfiy that user cannnot select or is not permitted to select seats that are already booked or not allowed for booking.
11. Verify that after selecting seats, entering passenger details and making payment the selected seats get booked.
12. Verify that on successful scheduling the ticket should be visible and downloadable.
13. Verify that after successful booking the seat's stuatus is updated to booked.
14. Verify the maximum limit of seats that a user can book, selecting more seats than permiited results in error message.
15. Verify that user can also cancel the tickets booked by entering the mandatory details and the amount after deducting the cancellation fee gets refunded back to user.
16. Verify that after cancellation the seat's status is updated to available.

In this we used various strategies of testing and some are as follows:

- **UNIT TESTING:** To test each module. That is Testing of GUI(seats , passengers , flights) and various functions.
- **INTEGRATION TESTING :** To test that various modules are working together.
-
- For example we give the input as:
- Number of airport=n=5 for this case.
- There will be N*N-N total number of flights in general. This is how the model will interpret the data.

	A	B	C	D	E
A	X	154	141	130	75

B	108	X	150	138	165
C	126	110	X	139	155
D	170	160	130	X	145
E	168	156	100	130	X

-
- After marking all minimum passengers carrying flight from each airport, a list of the flights will be created. Only the flights having passenger less than 135 and not equal to 0 are selected for the list.

	A	**B**	**C**	**D**	**E**
A	X	154	141	130	75
B	108	X	150	138	165
C	126	110	X	139	155
D	170	160	130	X	145
E	168	156	100	130	X

-

List
75
108
110
130
100

-
- From the list of minimum passenger carrying flight, the least is selected. The selected minimum should have the scope of shifting passengers to alternate routes.
-

	A	**B**	**C**	**D**	**E**
A	X	154	141	130	75

B	108	X	150	138	165
C	126	110	X	139	155
D	170	160	130	X	145
E	168	156	100	130	X

-

Vacancy	Possible shift	Shift
15	26	15
25	39	25
35	50	35

-
- After shifting the passengers the flight table is updated.

	A	B	C	D	E
A	X	169	166	165	0
B	108	X	150	138	180
C	126	110	X	139	180
D	170	160	130	X	180
E	168	156	100	130	X

-
- This process of the selection of minimum and shifting goes until possible.
- At last step, the list table will be empty and flight table will look like below.
-

	A	B	C	D	E

A	X	180	178	165	0
B	0	X	180	148	180
C	143	0	X	159	180
D	180	180	180	X	180
E	180	180	0	180	X

-

CHAPTER-5

CONCLUSION

We can hereby conclude that:

- The system effectively automated the functions involved in the processes being handled manually before.
- The cost & benefit analysis shows that the system was quite successful in saving costs for the bank & generate equivalently huge benefits
- The system is secure & scalable. The system design has been done keeping user- friendliness and efficiency in mind.

****