Chapter 1

INTRODUCTION

1.1 Background

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, the database itself to capture and analyze the data and provides facilities to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

1.2 Introduction to Airline Reservation System

An airline reservation system (ARS) is part of the so-called passenger service systems (PSS), which are applications supporting the direct contact with the passenger.

ARS eventually evolved into the computer reservations system (CRS). A computer reservation system is used for the reservations of a particular airline and interfaces with a global distribution system (GDS) which supports travel agencies and other distribution channels in making reservations for most major airlines in a single system.

Today all persons are busy with their schedule and no one have time to make a trip for holidays with their family. And this Airline Reservation Process is very difficult to understand in General meaning. But we are providing a Solution for that Problem.

This system provides a facility to easy access towards a customers and a real time users. They can easily connected through it and just 3 steps. There is no requirement for any type of Agent.

The Airline Reservation System project is an implementation of a general Airline Ticketing website like Goibibo, which helps the customers to search the availability and prices of various airline tickets, along with the different packages available with the reservations. This project also covers various features like online registration of the users, modifying the details of the website by the management staff or administrator of the website, by adding, deleting or modifying the customer details, flights or packages information.

In general, this website would be designed to perform like any other airline ticketing website available online. It demonstrates the use of payment gateways to accept payment and shows the implementation of transaction management if multiple transactions are made at the same time. It also demonstrates the use of AJAX calls for dynamic control over webpages. This project also implements the use of PHPMailer which is used to automatically send emails of their ticket confirmation and boarding passes. All the data for this project is collected using the **Flight Data API** which results in a very realistic experience when using the website

Chapter 2

E R DIAGRAM AND RELATIONAL SCHEMA DIAGRAM

2.1 Description of ER Diagram

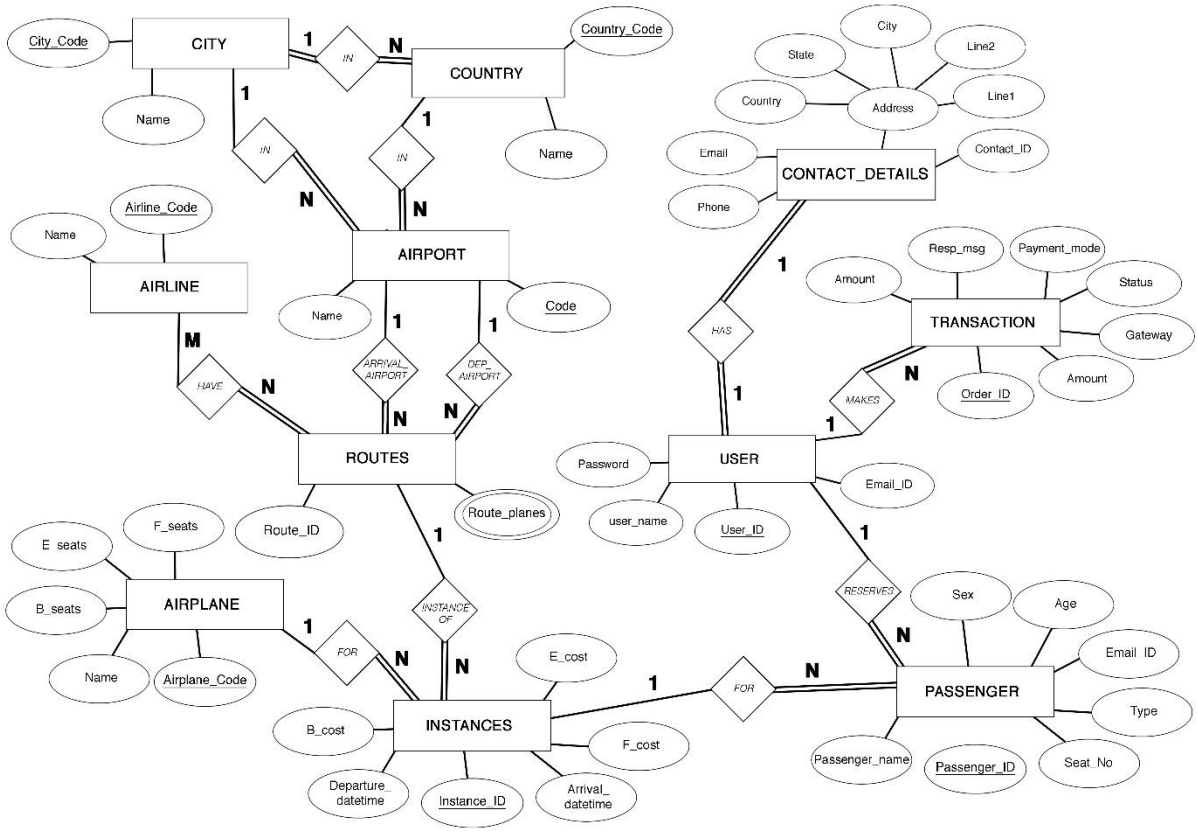

Figure 2.1: E-R Diagram for Airline Reservation System

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

2.1.1 Components of Airline Reservation System, E-R Diagram - Figure 2.1

1. Entity types like **ROUTES** and **PASSENGER** are in rectangular boxes.
2. Relationships like **ARRIVAL_AIRPORT** and **HAS** are in diamond boxes, attached to entity types with straight lines.
3. Attributes are shown in ovals like **Name** and **Seat_No**, each attached by a straight line to entity or relationship type.
4. Key attributes like **Passenger_ID** and **Airport_code** are underlined.
5. Component attributes of a composite attribute like **address** are attached to oval representing it.
6. Multivalued attributes like in **Route_planes,** are represented using double layered oval.

2.1.2 E-R Diagram Relationships Description

1. **CITY:AIRPORT** is of cardinality 1:N as one city can possibly have more than one airport and therefore connected by an **IN** relationship. There is total participation from AIRPORT as all airports are located in some city, but partial participation from CITY as a city may exist without an airport.
2. **COUNTRY:AIRPORT** is of cardinality 1:N as one country can have more than one airport and therefore connected by an **IN** relationship. There is total participation from AIRPORT as all airports are location is some country, but partial participation from COUNTRY as a country may exist without an airport.
3. **AIRPORT:ROUTES** is of cardinality 1:N in both the relationships **ARRIVAL_AIRPORT** and **DEPARTURE_AIRPORT** as one airport can be involved in multiple routes, but one route can have only one departure and arrival airport. There is total participation from routes as all routes are with respect to some airport, and partial participation from AIRPORT as an airport can exist without being assigned to a route.
4. **AIRLINE:ROUTES** is of cardinality M:N as one route can have multiple airlines assigned to it and one airline can service multiple routes and therefore connected by **HAVE** relationship. There is total participation from ROUTES as all routes must be serviced by some airline, but partial participation from AIRLINE as an airline may or may not service a route.
5. **ROUTES:INSTANCES** is of cardinality of 1:N as one route can have multiple instances depending on the departure date and time and therefore connected by an

INSTANCE_OF relationship. There is total participation from INSTANCES as all instances are derived from a route, but partial participation from ROUTES as a route may not have an instance.

6. **AIRPLANE:INSTANCES** is of cardinality 1:N as one airplane can be assigned to multiple instances, but one instance is assigned only one airplane and therefore connected by **FOR** relationship. There is total participation from INSTANCES as all instances are assigned some airplane, but partial participation from AIRPLANE as one airplane may not service any flight instance.
7. **CONTACT_DETAILS:USER** is of cardinality 1:1 as one user has only one contact detail and one contact details in for one user and is therefore connected by **HAS** relationship. There is total participation from both CONTACT_DETAILS and USER as all contact details are assigned to some user and all users have contact details.
8. **USER:TRANSACTION** is of cardinality 1:N as one user can make many transactions and one transaction is made by only one user and therefore connected by **MAKES** relationship. There is total participation from TRANSACTION as all transactions are made by some user, but partial participation from USER as may not make a transaction.
9. **USER:PASSENGER** is of cardinality 1:N as one user can book many passenger tickets and one passenger ticket is booked by only one user, and therefore connected by **RESERVES** relationship. There is a total participation from PASSENGER as all passenger tickets are booked by some user, but partial participation from USER as one user may or may not reserve a seat.
10. **INSTANCES:PASSENGER** is of cardinality 1:N as one instance can have multiple passengers and one passenger is for only one instance and therefore connected by **FOR** relationship. There is a total participation from PASSENGER as all passengers belong to some instance, but partial participation from INSTANCE as few instances may or may not have passengers assigned to them.

2.2 Description of Relational Schema Diagram

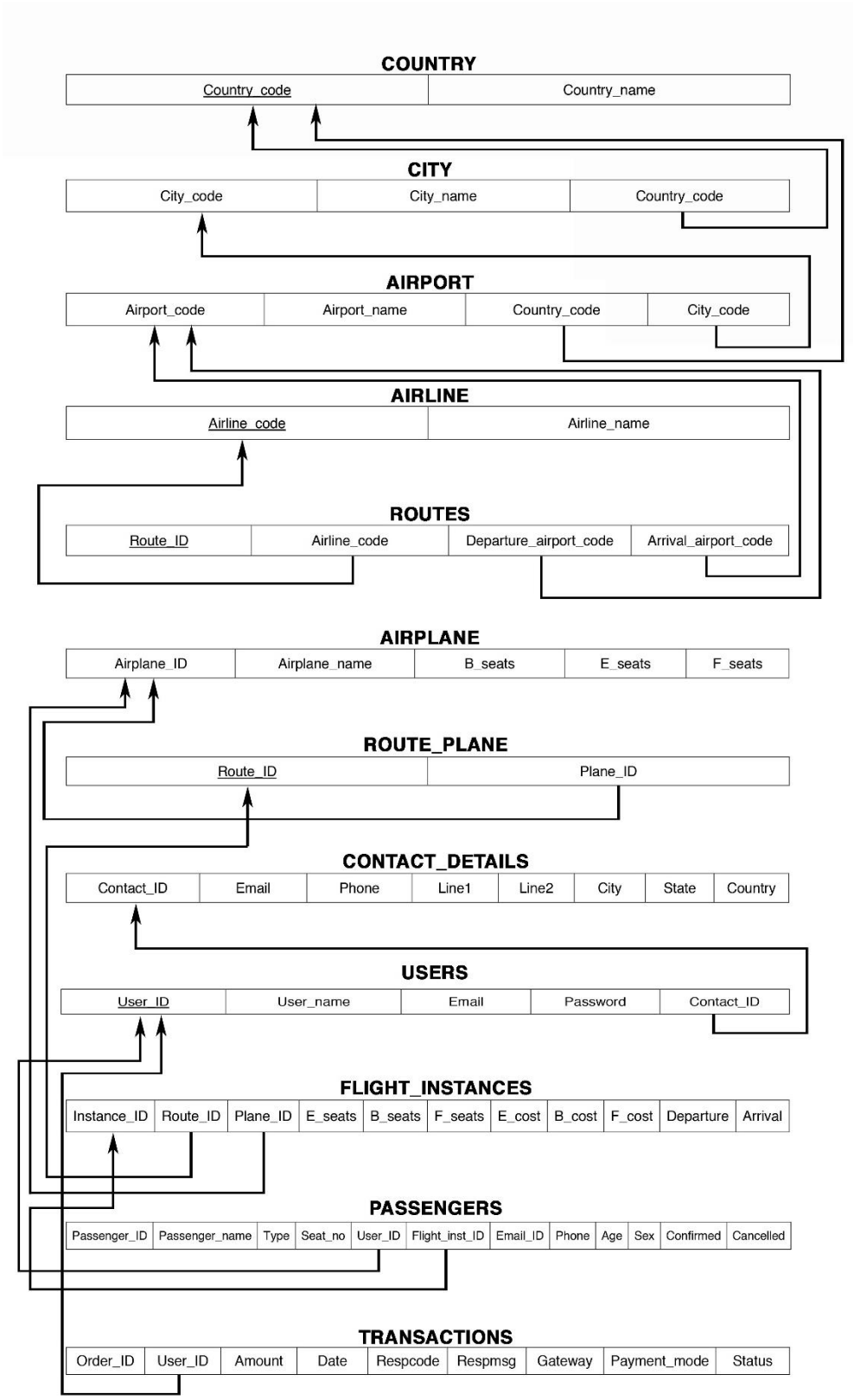

Figure 2.2 Relational Schema Diagram for Airline Reservation System

2.2.1 General Constraints

1. **NULL Constraint**: Attributes that are under NOT NULL constraints have to be filled compulsorily. Almost all the attributes in the project are under NOT NULL constraint.
2. **Entity Integrity Constraint**: This constraint makes sure that no primary key can have a NULL value assigned to it. The primary keys involved in the project include:
    - Code
    - Flight_inst_ID
    - Passenger_ID
3. **Referential Integrity Constraints**: A table in the back end of the project may have references pointing to an attribute in another table. For example: FLIGHT_INST_ID in the PASSENGER table refers to INSTANCE_ID in FLIGHT_INSTANCES table. The various tables are also linked with multiple foreign keys which are all set to cascade any update or delete operation on the attribute in the main table. The various Foreign Key attributes are:
    - Flight_inst_ID
    - Plane_ID
    - City_code
    - Country_code

2.2.2 Schema Description

The above Figure.2.2 shows the relational schema of Airline Reservation System. It has the following entities.

1. **COUNTRY:** This table contains the details like IATA code, name. IATA code is the primary key.
2. **CITY:** This table contains the names of all the cities in the world. It has 3 columns, IATA city code, name, Country code, referring to entry in previous table.
3. **AIRPORT:** This table consists the list of airports passengers can board and arrive at. It has 4 columns IATA airport code, name, country code and city code.
4. **AIRLINE:** This table consists of all the airlines and has IATA airline code and name as attributes
5. **ROUTES:** This table consists of list of all the routes serviced by each airline registered with the system. It has the following columns, Route ID, airline code,

departure airport code and arrival airport code. An assumption is made that, all flights taken into consideration are direct flights from source to destination.

6. **AIRPLANE:** This table contains a list of airplanes that can be used for any routes. It has airplane ID, name, economy seats, business seats and first class seats as its columns.
7. **ROUTE_PLANES:** This table is used to represent the MVD in the ROUTES entity, it has 2 columns route ID and airplane ID.
8. **CONTACT DETAILS:** This table contains the contact details of all the users registered with the reservation platform. It has contact ID, email, phone, line1, line2, city, state and country as attributes.
9. **USERS:** This contains the credentials of all the users registered with the booking system. It has user ID, username, email, password and contact ID as its columns.
10. **FLIGHT INSTANCES:** This is the master table, it contains the list of all routes and its schedules along with the number of seats available in each travel class and the airplane assigned to this instance. It has instance ID, route ID, plane ID, e seats, b seats, f seats, e cost, b cost, f cost, departure and arrival as its attributes.
11. **PASSENGER:** This table contains the list of passengers who have booked for multiple flight instances. It contains, passenger ID, name, type, seat no. , user ID, email ID, phone, age, sex as its attributes.
12. **TRANSACTIONS:** This table contains all the transactions made by the users for any particular bookings. It has the following columns order ID, user ID, amount, date, response code, response message, gateway payment mode and status.

Chapter 3

SYSTEM DESIGN

3.1 Table Description

COUNTRY

FIELD	TYPE	NULL?	KEY	DEFAULT
Code	VARCHAR(10)	NO	PRIMARY	None
Name	VARCHAR(100)	NO		None

Table 3.1 Country

CITY

FIELD	TYPE	NULL?	KEY	DEFAULT
Code	VARCHAR(10)	NO	PRIMARY	None
Name	VARCHAR(100)	NO		None
Country_code	VARCHAR(10)	NO	FOREIGN	None

Table 3.2 City

AIRPORT

FIELD	TYPE	NULL?	KEY	DEFAULT
Code	VARCHAR(10)	NO	PRIMARY	None
Name	VARCHAR(100)	NO		None
Country_code	VARCHAR(10)	NO	FOREIGN	None
City_code	VARCHAR(10)	NO	FOREIGN	None

Table 3.3 Airport

AIRLINES

FIELD	TYPE	NULL?	KEY	DEFAULT
Code	VARCHAR(10)	NO	PRIMARY	None
Name	VARCHAR(100)	NO		None

Table 3.4 Airlines

ROUTES

FIELD	TYPE	NULL?	KEY	DEFAULT
Route_ID	INTEGER	NO	PRIMARY	None
Airline_code	VARCHAR(10)	NO	FOREIGN	None
Departure_airport_code	VARCHAR(10)	NO	FOREIGN	None
Arrival_airport_code	VARCHAR(10)	NO	FOREIGN	None

Table 3.5 Routes

AIRPLANE

FIELD	TYPE	NULL?	KEY	DEFAULT
Code	VARCHAR(10)	NO	PRIMARY	None
Name	VARCHAR(100)	NO		None
Eseats	INTEGER	YES		0
Bseats	INTEGER	YES		0
Fseats	INTEGR	YES		0

Table 3.6 Airplane

ROUTE_PLANES

FIELD	TYPE	NULL?	KEY	DEFAULT
Route_ID	INTEGER	NO	PRIMARY	None
Plane_ID	VARCHAR(10)	NO	PRIMARY FOREIGN	None

Table 3.7 Route Planes

INSTANCES

FIELD	TYPE	NULL?	KEY	DEFAULT
Instance_ID	BIG INTEGER	NO	PRIMARY	None
Route_ID	INTEGER	NO	FOREIGN	None
Plane_ID	VARCHAR(10)	NO	FOREIGN	None
Eseats	INTEGER	NO		None
Bseats	INTEGER	NO		None
Fseats	INTEGER	NO		None

ecost	INTEGER	NO		None
bcost	INTEGER	NO		None
fcost	INTEGER	NO		None
departure	DATETIME	NO		None
arrival	DATETIME	NO		None

Table 3.8 Instances

CONTACT_DETAILS

FIELD	TYPE	NULL?	KEY	DEFAULT
Contact_ID	INTEGER	NO	PRIMARY	None
Email	VARCHAR(40)	NO		None
Phone	VARCHAR(10)	NO		None
Line1	VARCHAR(40)	NO		None
Line2	VARCHAR(40)	NO		None
City	VARCHAR(40)	NO		None
State	VARCHAR(30)	NO		None
Country	VARCHAR(30)	NO		None

Table 3.9 Contact Details

USERS

FIELD	TYPE	NULL?	KEY	DEFAULT
User_ID	INTEGER	NO	PRIMARY	None
U_name	VARCHAR(50)	NO		None
Email	VARCHAR(40)	NO		None
Pass	VARCHAR(200)	NO		None
Contact_det	INTEGER	NO	FOREIGN	None

Table 3.10 Users

PASSENGER

FIELD	TYPE	NULL?	KEY	DEFAULT
Passenger_ID	INTEGER	NO	PRIMARY	None
Passenger_name	VARCHAR(40)	NO		None

Type	VARCHAR(40)	NO		None
Seat_no	VARCHAR(5)	YES		None
User_ID	INTEGER	NO	FOREIGN	None
Flight_inst_ID	INTEGER	NO	FOREIGN	None
Email_ID	VARCHAR(40)	NO		None
Phone	VARCHAR(10)	NO		None
Age	INTEGER	NO		None
Sex	VARCHAR(10)	NO		None
confirmed	INTEGER	NO		0
cancelled	INTEGER	NO		0

Table 3.11 Passenger

TRANSACTION

FIELD	TYPE	NULL?	KEY	DEFAULT
Order_ID	VARCHAR(100)	NO	PRIMARY	None
User_ID	INTEGER	NO	FOREIGN	None
Amount	FLOAT	NO		None
Date	DATETIME	NO		None
Respcode	VARCHAR(100)	NO		None
Respmsg	VARCHAR(100)	NO		None
Gateway	VARCHAR(100)	NO		None
Payment_mode	VARCHAR(100)	NO		None
Status	VARCHAR(100)	NO		None

Table 3.12 Transaction

3.2 Stored Procedures

One stored is used to get the flight details when the user submits the initial form to view the tickets from source to destination on a specified date for a specified number of travelers and travel class type.

```
CREATE PROCEDURE `flight_data` (IN `origin_code` VARCHAR(10), IN `destination_code`
VARCHAR(10), IN `travellers` INT, IN `dep` VARCHAR(20), IN `d` DATETIME, IN `type`
VARCHAR(20))  NO SQL
BEGIN
    IF (type = 'eseats') THEN
        SELECT *
        FROM instances i, routes r, airplane ap
        WHERE i.Route_ID = r.Route_ID
        AND i.plane_ID = ap.Code
        AND r.departure_airport_code = origin_code
        AND r.arrival_airport_code = destination_code
        AND i.eseats > 0
        AND i.eseats >= travellers
        AND departure LIKE dep
        AND i.departure > d
        ORDER BY i.ecost;
    ELSEIF (type = 'bseats') THEN
        SELECT *
        FROM instances i, routes r, airplane ap
        WHERE i.Route_ID = r.Route_ID
        AND i.plane_ID = ap.Code
        AND r.departure_airport_code = origin_code
        AND r.arrival_airport_code = destination_code
        AND i.bseats > 0
        AND i.bseats >= travellers
        AND departure LIKE dep
        AND i.departure > d
        ORDER BY i.bcost;
    ELSE
        SELECT *
        FROM instances i, routes r, airplane ap
        WHERE i.Route_ID = r.Route_ID
        AND i.plane_ID = ap.Code
        AND r.departure_airport_code = origin_code
        AND r.arrival_airport_code = destination_code
        AND i.fseats > 0
        AND i.fseats >= travellers
        AND departure LIKE dep
        AND i.departure > d
        ORDER BY i.fcost;
    END IF;
END
```

It takes six parameters origin airport code, destination airport code, number of travelers, date of departure, current date and type of seats being booked.

3.3 Triggers

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

In this project a trigger is added to reduce the number of seats left when a passenger is confirmed.

```
CREATE TRIGGER update_inst
AFTER UPDATE
ON Passenger FOR EACH ROW
    BEGIN
        iF (OLD.confirmed = 0) AND (NEW.confirmed = 1) THEN
            IF (OLD.Type = 'Economy') THEN
                UPDATE instances SET eseats = eseats - 1 WHERE Instance_ID =
                NEW.Flight_inst_ID;
            ELSE IF (OLD.Type = 'Business') THEN
                UPDATE instances SET bseats = bseats - 1 WHERE Instance_ID =
                NEW.Flight_inst_ID;
            ELSE
                UPDATE instances SET fseats = fseats - 1 WHERE Instance_ID =
                NEW.Flight_inst_ID;
            END IF;
        END IF;
    END
```

The above trigger is called whenever an update is made on the passenger table, and if the confirmed value is changed from 0 to 1, i.e. the passenger details are confirmed, then decrement the number of seats left in the instances table.

Chapter 4

IMPLEMENTATION

4.1 Front-end Development

The front-end is built using a combination of technologies such as Hypertext Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS). Front-end developers design and construct the user experience elements on the web page or app including buttons, menus, pages, links, graphics and more.

4.1.1 Hypertext Markup Language

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. HTML is the standard markup language for creating Web pages. It stands for Hyper Text Markup Language. It describes the structure of a Web page. It consists of a series of elements. It elements tell the browser how to display the content. It elements are represented by tags. HTML tags label pieces of content such as "heading", "paragraph", "table", and so on. Browsers do not display the HTML tags, but use them to render the content of the page.

4.1.2 Cascading style sheets

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications. Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. CSS solved that issue. SS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file. CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

4.1.3 JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly. Advantages are: **Less server interaction, immediate feedback to the visitors, increased interactivity** and **richer interfaces.**

4.2 Back-end Development

Backend is server side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs, creating libraries, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.

4.2.1 Backend scripting language - PHP Hypertext Preprocessor

PHP is used as the server side scripting language. PHP is an acronym for "PHP: Hypertext Preprocessor". PHP is a widely-used, open source scripting language. PHP scripts are executed on the server. It is compatible with all servers used today. It is easy to use and runs efficiently on the server side. It can run on various platforms like windows, Linux, UNIX, Mac OS-X etc. and since it is a scripting language, it comes with predefined functions which makes it easy to implement any logic necessary.

4.2.2 Web Server – Apache

Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation. It allows website owners to serve content on the web — hence the name "web server". Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server. Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). Apache is a cross-platform software, therefore it works on both UNIX and Windows servers.

4.2.3 Database – MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. It is developed, marketed and supported by MySQL AB, which is a Swedish company. It is released under an open-source license. So you have nothing to pay to use it. It is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages. It uses a standard form of the well-known SQL data language. It works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. It works very quickly and works well even with large data sets. It is very friendly to PHP, the most appreciated language for web development. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB). It is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

4.3 User Flow Diagram

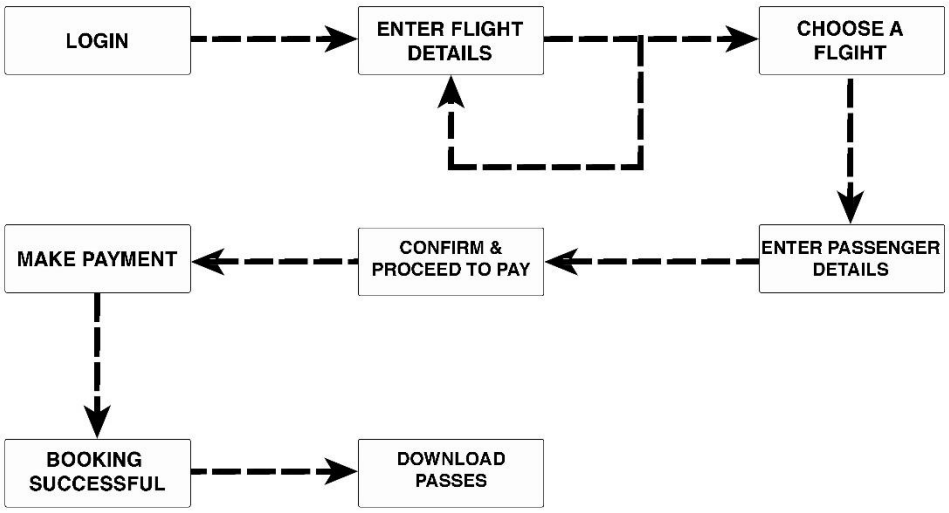

Figure 4.1 Successful Booking user flow diagram

The above Figure 4.1 shows the user flow diagram for a successful booking. The user logs in, enters the flight details he is looking for, chooses one of the flights from the available list, enters the passenger details and confirm and makes the payment, on payment successful he is allowed to download his boarding passes.

4.4 Discussion of Code Segment

This section talks about the important code sections and modules that are implemented in the Airline Reservation System project. These modules add logic to the complete system, and make it function the way it is supposed to. It also talks about the integration between the front end HTML code and the back end MySQL database.

4.4.1 Login Module

Pseudocode Login:

```
username := input()
password := input()
if(submit.click()) then
    if (validate(username, password) then
        make ajax call to login.php
```

```
            select userID, pass from users where U_name = username
            if(password_verify(password, pass) then
               session['uid'] := userID
               session['uname'] := username
               print(true)
            else
               print(false)
            end if;
         if true then
            refresh page
         else
            print("wrong username or password")
         end if;
      else
         print("Enter valid details")
      end if;
   end if;
```

4.4.2 Sign-up module

Pseudocode Signup:

```
   details := array()
   details[username] := input()
   details[email_id] := input()
   details[password] := input()
   details[re-enter_password] := input()
   details[phone] := input()
   details[address] := input()
      if(submit.click()) then
         if(validate(details)) then
            if(details[password] = details[re-enter_password]) then
```

```
        make ajax call to signup.php
            insert details into users
            if(success(insert)) then
                print(true)
            else
                print(false)
            end if;
        if true then
            print('Sign Up succesful')
            close(signup-box)
            open(login-box)
        end if;
    else
        print("passwords don't match")
    end if;
else
    print("Enter valid details")
end if;
```

4.4.3 Get Flight Details module

Pseudocode Get_flight_details:

```
details := array()
details[origin] := input()
details[destination] := input()
details[date_of_departure] := input()
details[travel_class] := input()
details[travelers] := input()
if(validate(details)) then
    make ajax call to getplanedetails.php
    res:= call procedure(flight_data)
```

4.4.4 Book Tickets module

Pseudocode Book_tickets:

```
passenger_details := array()
passenger_details := input()
if(confim.click()) then
    redirect(payment_gateway, callback = callback.php)
on(callback) insert response into transactions
    if(response_code = '01') then
        if(seats_available(flught_id) >= travelers) then
            assign_seats(id = flight_id, details = passenger_details)
            reduce_seats(id = flight_id, seats = travelers)
            redirect(bookingsuccessful.php)
        else
            print("No seats available, already alloted")
            initiate_refund(id = transaction_id)
            redirect(bookingfail.php)
        end if;
    else
        print("Transaction unsuccessful")
        redirect(bookingfail.php)
    end if;
```

4.5 Discussion of Results

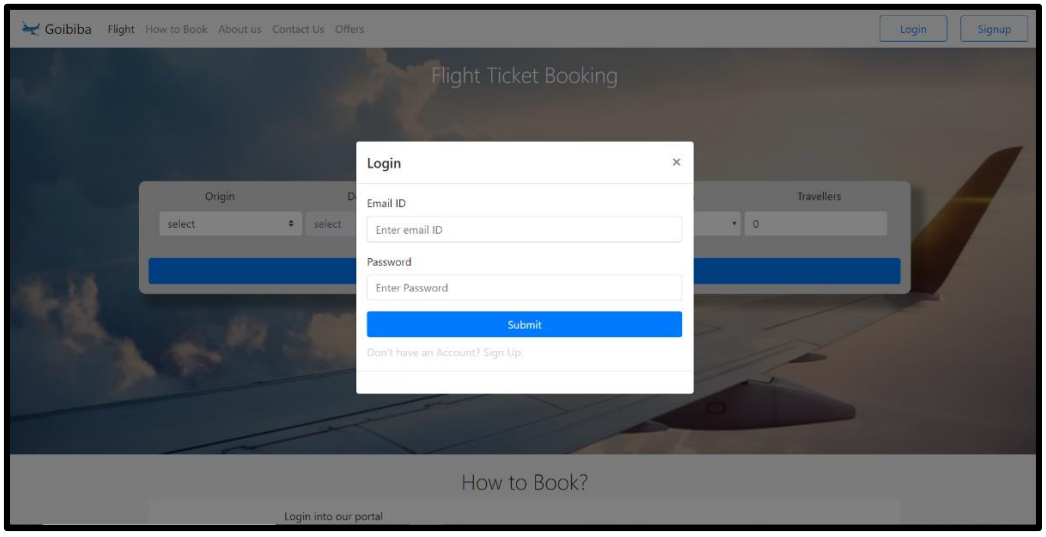

Figure 4.2 Homepage with login prompt

The above Figure 4.1 is the snapshot of the homepage with the login prompt. If the user has already registered with the website, he can login using his email ID or username which he used to register along with the valid password also which he set during sign up.

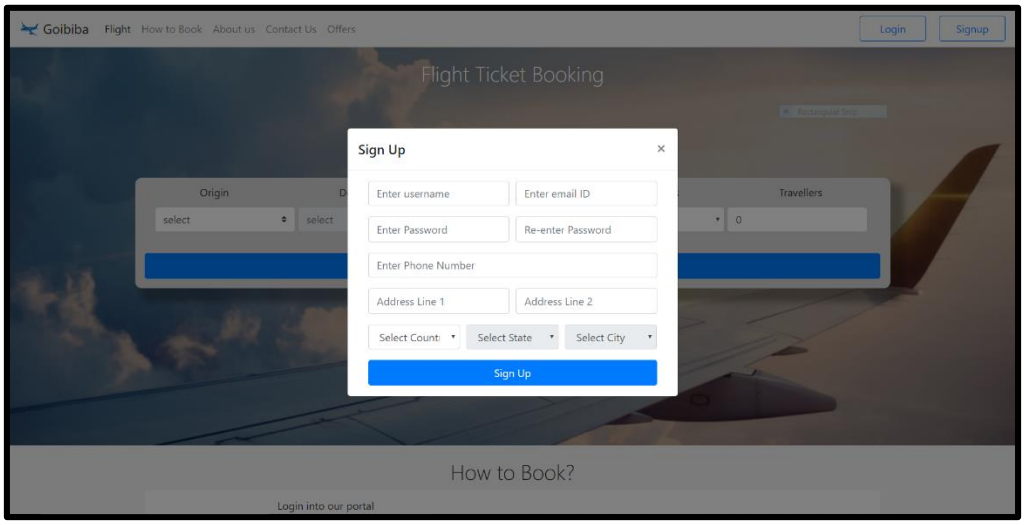

Figure 4.3 Signup Prompt

This is the signup prompt, if a new user wants to register with the website, he can use this form to register. It accepts the username, email ID, password, and his contact details like phone number and address. The state and city fields automatically get enabled once a country is selected.

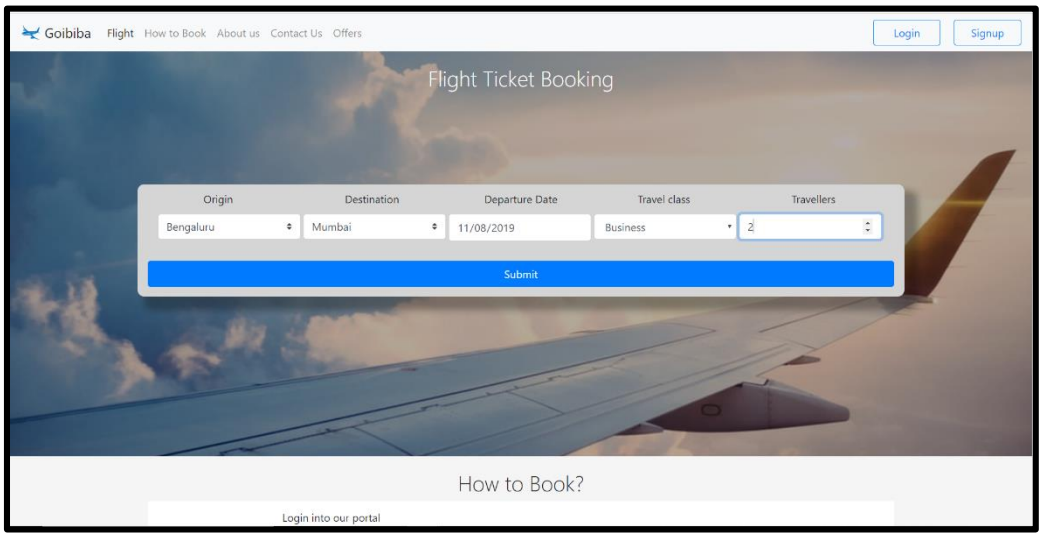

Figure 4.4 Flight details entered in the flight ticket booking form

The above figure 4.4 shows the form filled with the origin and destination cities, the departure date, travel class and number of travelers. When the origin destination is selected, the destination cities are automatically loaded in the second drop down using an ajax call. Errors are also handled for choosing wrong date, not selected a travel class or enter travelers less than 1 or more than 9.

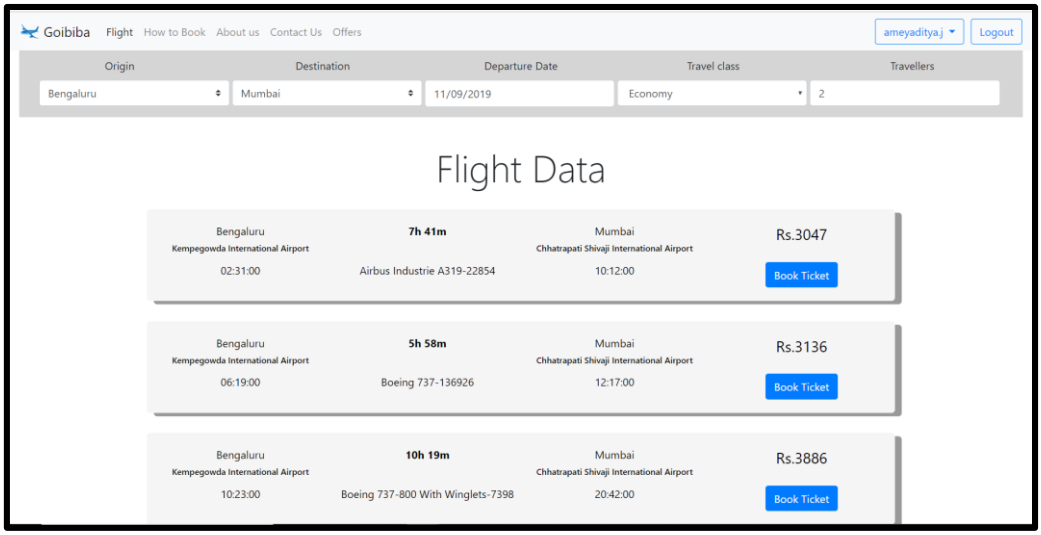

Figure 4.5 List of flights available

This page shown in the above Figure 4.5 displays the list of flights available from the origin to destination city, on the date selected for the entered number of passengers. The list of order based on the price per ticket. Users can dynamically change the search to their interest.

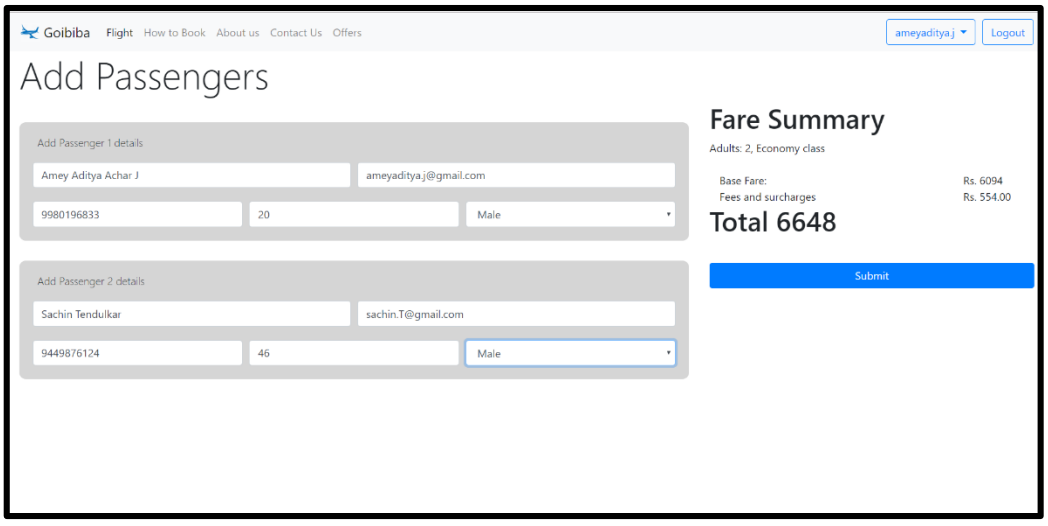

Figure 4.6 prompt to enter passenger details

The above Figure 4.6 show a form that prompts the user to enter the passenger details, their name, email ID, Phone number, Age and Sex. The total fare along with fees and surcharges is also displayed.

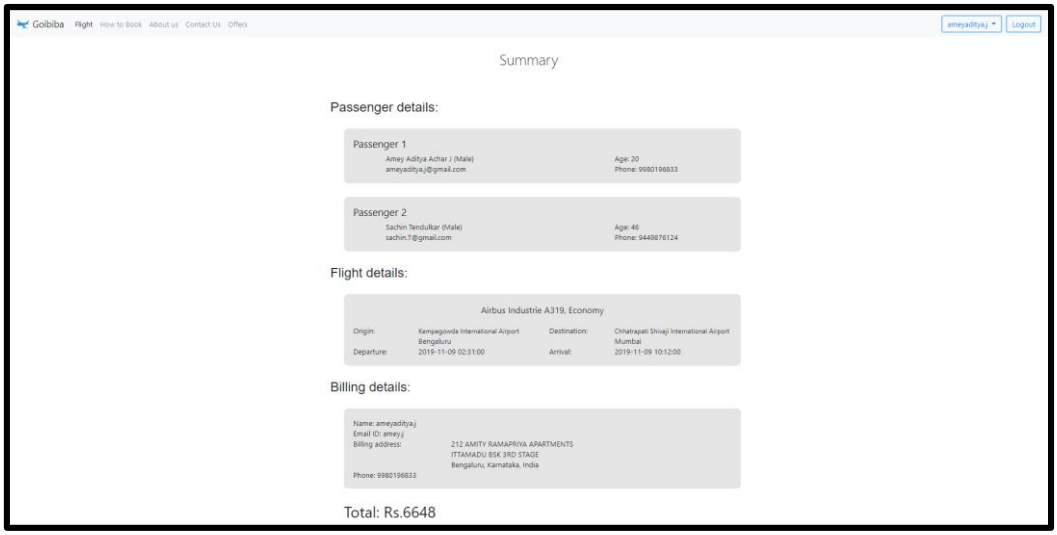

Figure 4.7 Summary of passenger booking

The above Figure 4.7 shows the summary of the passenger details booked by the user. The user can verify the details and click confirm and pay.

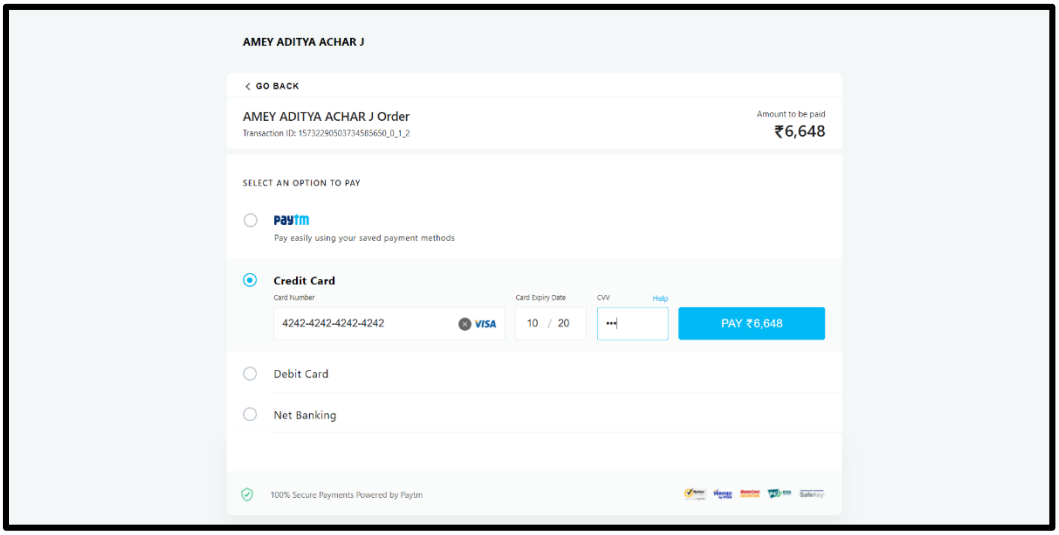

Figure 4.8 Payment gateway redirection to complete payment

The above Figure 4.8 shows the webpage being redirected to the payment gateway, to complete the payment for the specified amount. User can opt to use any of the payment modes specified.

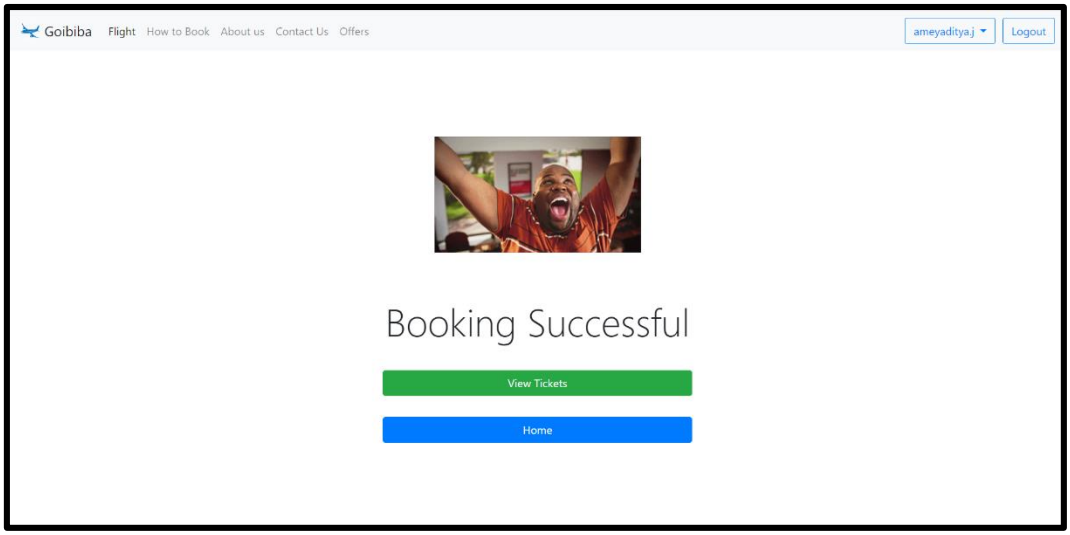

Figure 4.9 booking successful

Once the payment is made, if successful, user is redirected to this page displayed in the above Figure 4.9 where the user may choose to view his tickets or exit to home page.

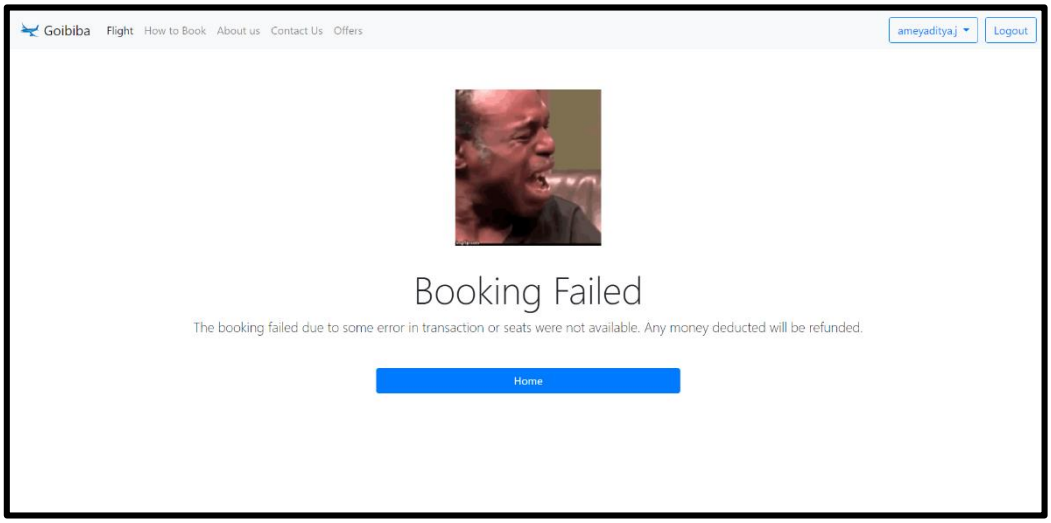

Figure 4.10 Booking Failed

Once the payment is made, if unsuccessful, user is redirected to this page displayed in the above Figure 4.10 where the user is prompted with a message about failed transaction and a button to redirect to homepage.

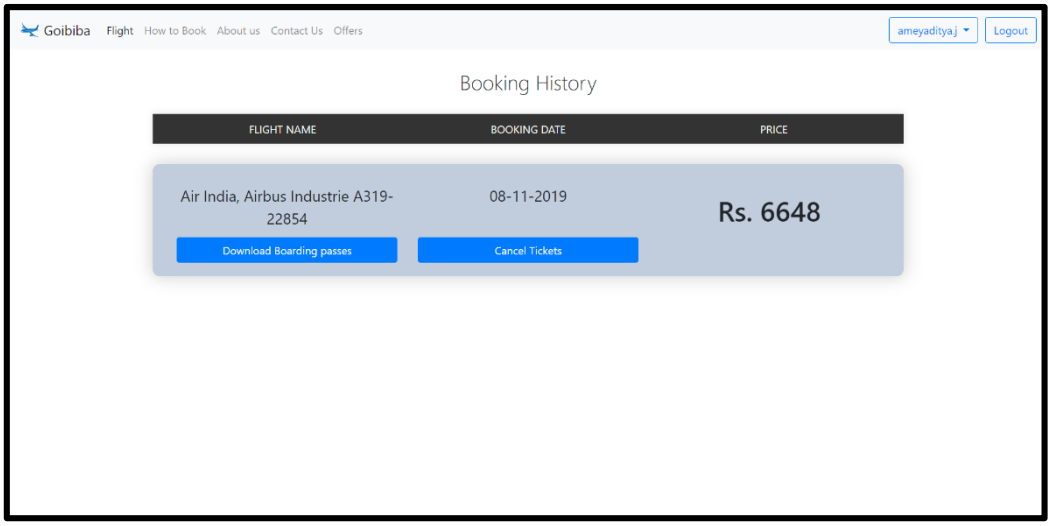

Figure 4.11 booking history

This page displayed in the Figure 4.11 above shows the list of tickets the user has booked, the user can choose to download the boarding passes or cancel the tickets.

Figure 4.12 boarding passes

The above displayed Figure 4.12 shows the boarding passes of the passenger tickets booked by the customer. They have the necessary details written on them.

4.6 Application of project

These types of Airline Reservation Systems are examples of flourish technology has had on the world. These help users book their flight tickets from their laptops, mobiles phones or tablets. They are easy to use and get the job done in very few steps. The old methods of standing is long lines and waiting for hours is gone. As they are deployed on the web, they can be multi lingual which would support diversity and cause these systems to spread worldwide. These system also help integrate use of multiple currency options to make the payments. As the boarding passes can be carried on phones, the problem of losing them is also removed and is also eco-friendly.

Major applications of Airline Reservation Systems are:

1. All flight and route data are available at one place.
2. Easy access to boarding pass and available as a digital copy.
3. A database of all the year's activity is created for use if needed later.
4. Support multilingualism and use of multiple currencies.

Chapter 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

This project was an attempt to make the structure and working of an airline reservation system simpler and user-friendly. This was an attempt to make it as similar to the real world implementations. In this scenario, all the undertakings of the Airline Reservation System was achieved in a constructive manner. Given the right guidance and support its applications and availability can be enhanced.

5.2 Future Enhancements

1. The project support airline reservation of direct flights without any transfers, in the future this can be an addition to make the system more robust.
2. Email and message verifications could be sent to the users on successful booking of tickets.
3. Hosting it on an online web server.
4. Integrating both onward and return journey ticket booking in the one single transaction.
5. Collecting and analyzing flight price data to predict and book ticket when the price is lowest.

Chapter 6

REFERNCES

[1] Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, McGRAW HILL, 3rd Edition.

[2] Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, Pearson, 7th Edition.

[3] Herbert Schildt, Java: The Complete Reference, McGRAW HILL, 7th Edition.

[4] www.stackoverflow.com

[5] www.mysql.com/doc

[6] https://docs.microsoft.com

[7] https://www.geeksforgeeks.org

[8] https://www.rapidapi.org