VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

Mini Project Report
on

AIRLINE RESERVARTION MANAGEMENT SYSTEM

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by
GANESH SINDAGI
(1BG18CS038)

Vidyayāmruthamashnuthe

B.N.M. Institute of Technology

Approved by AICTE, Affiliated to VTU, Accredited as grade A Institution by NAAC.
All UG branches – CSE, ECE, EEE, ISE & Mech.E accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

Department of Computer Science and Engineering
2020-21

B.N.M. Institute of Technology

Approved by AICTE, Affiliated to VTU, Accredited as grade A Institution by NAAC.

All UG branches – CSE, ECE, EEE, ISE & Mech.E accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021

Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA

Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

Department of Computer Science and Engineering

Vidyayāmruthamashnuthe

CERTIFICATE

Certified that the Mini Project entitled **AIRLINE RESREVATION Management System** carried out by **Mr. GANESH SINDAGI** USN **1BG18CS038** a bonafide student of V Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of the **Visvesvaraya Technological University**, Belagavi during the year 2020-21. It is certified that all corrections/ suggestions indicated for internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Database Management Systems Laboratory with Mini Project as prescribed for the said degree.

Prof. Asha K
Assistant Professor & Lab-Incharge
Department of CSE
BNMIT, Bengaluru

Dr. Sahana D. Gowda
Professor & HOD
Department of CSE
BNMIT, Bengaluru

Name & Signature

Examiner1:

Examiner2:

ABSTRACT

The Project ‘Airline reservation System’ is a computerized system used to store and retrieve information and conduct transactions related to air travel. The aim of the project is to expose the relevance and importance of Airline Reservation Systems.

The system allows the airline passenger to search for flights that are available between the two travel cities, namely the “Departure city” and “Arrival city” for a particular departure date. The system is designed such that flights are available on all days. The system displays all the flight’s details such as flight no, name, price etc.

Then the system checks for the availability of seats on the flight. If the seats are available, then the system allows the passenger to book a seat. Otherwise it asks the user to choose another flight.

The system asks the customer to enter his details such as name, age, email and contact number to book a flight. The system also allows the customer to cancel his/her reservation, if any problem occurs.

The main purpose of the software is to reduce the manual errors involved in the airline reservation process and make it convenient for the customers to book the flights as and when they require. The software allows customer to make reservations, modify reservations or cancel a particular reservation.

# ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I would like to thank **Shri. Narayan Rao R Maanay**, Secretary, BNMIT, Bengaluru for providing excellent academic environment in the college.

I would like to sincerely thank **Prof. T J Rama Murthy**, Director, BNMIT, Bengaluru for having extended his support and encouragement during the course of the work.

I would like to express my gratitude to **Prof. Eishwar N Maanay**, Dean, BNMIT, Bengaluru for his relentless support, guidance and assistance.

I would like to thank **Dr. Krishnamurthy G N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank, **Dr. Sahana D Gowda**, Professor and Head of the Department of Computer Science and Engineering who has shared her opinions and thoughts which helped me in giving my presentation successfully.

I extend my heartfelt gratitude to **Prof Asha K**, Assistant Professor, Department of Computer Science and Engineering, BNMIT, Bengaluru, for her assistance and guidance in doing this project.

Finally, I would like to thank all the faculty members of Department of Computer Science and Engineering, BNMIT, Bengaluru, for their support. I would like to thank my family and friends for their unfailing moral support and encouragement.

GANESH SINDAGI
1BG18CS038

Table of Contents

List of Figures

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Overview of Database Management Systems

A Database Management System(DBMS) is a general purpose software system that allows creation, definition and manipulation of a database, allowing users to store, process and analyze data easily. A Database Management System (DBMS) provides with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the data base and a lot more. Modern Database Management Systems (DBMS) also provide protection and added security features to the databases. In addition, it also maintains data consistency in case of multiple users. Some examples of the most commonly used Database Management Systems are MySQL, ORACLE DB, IBM DB2, and Amazon Simple DB.

1.1.1 Characteristics of a Database Management System

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

1.1.2 Advantages of a Database Management System

- **Controlling data redundancy:**

  In non-database systems each application program has its own private files. In this case, the duplicated copies of the same data are created in many places. In DBMS, all data of an organization is integrated into a single database file. The data is recorded in only one place in the database and it is not duplicated.

- **Sharing of data:**

  In DBMS, data can be shared by authorized users of the organization. The database administrator manages the data and gives rights to users to access the data. Many users can be authorized to access the same piece of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

- **Data consistency:**

  By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users.

- **Data security:**

  Form is very important object of DBMS. It can be created very easily and quickly in DBMS. Once a form is created, it can be used many times and it can be modified very easily. The created forms are also saved along with database and behave like a software component. A form provides very easy way (user- friendly) to enter data into database, edit data and display data from database. The non-technical users can also perform various operations on database through forms without going into technical details of a database.

- **Integration of data:**

  In Database management system, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). It makes easy to retrieve and update data.

- **Integration constraints:**

  Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or may be applied to relationships between records.

1.1.3 Advantages of Database Management System

- Provides data abstraction and segregation of application program from the data.
- Reduced redundancy of data ensures maximum cost efficiency for the storage of data
- Reduced development time while building applications that use database

1.2 Problem Statement

Flight booking system should not be limited only to airport and some flight booking agencies. Flight booking should be available to all the people who desire to travel by plane. There is a requirement of an application which is easy to use and takes care of all the passenger's needs like searching for flights, checking for available seats, booking a seat in the flight, etc.

1.3 Objectives

The project is aimed to reduce the manual work involved in data maintenance in the Flight Booking and automates the Airline Reservation System. The project is developed mainly to simplify the manual work and allows smooth administration of the operations of airlines. The purpose of the project is to computerize the administrative operations of a Flight Booking and to develop software which is user friendly, simple, fast, and cost – effective. It deals with the collection of Users, Flights, Seats and Booking information, Fare details, etc. Traditionally, it was done manually. The main function of the system is to enter and book Flights and retrieve the details as and when required, and also to manipulate these details meaningfully.

1.4 Data Set Description

Given below are the entities along with its attributes and relations present in the database of the application that are used to retrieve information from the database as per requirement of user.

- An entity type **'ADMIN'** with attributes admin_id, username and password where admin_id is the primary key attribute.
- An entity type **'USER** with attributes user_id, username, password and email where user_id is the primary key attribute.
- An entity type **'FLIGHTS** with attributes flight_id, name, source, destination, date, arr_time, dept_time and total_seats where flight_id is the primary key attribute.
- An entity type **'SEATS'** with attributes seat_no, flight_id, seat_type, class, fare, status where seat_no is the primary key attribute and flight_id is the foreign key attribute referencing from '**FLIGHTS'** table.
- An entity type **'BOOKING'** with attributes booking_id, passenger_id, flight_id, seat_no, name, age, email, phone, payment_type, act_fare, disc_fare where booking_id acts as the primary key attribute and passenger_id is the foreign key attribute referencing from '**USER'** table and seat_no is the foreign key attribute referencing from '**SEATS'** table.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

2.1.1 Front End

- HTML5/CSS/JavaScript/EJS Template engine.
- Google Chrome/Mozilla Firefox/Edge (Web Browsers).

2.1.2 Back End

- Node (v14.15.1) Environment.
- Express.js (v4.17.1) Backend Framework.
- PostgreSQL (v12.3) for Database Management System
- pgAdmin4 (v4) server for Database
- Visual Studio Code (v1.52.1) – Code Editor
- Windows 10

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

- Cores: Single-Core (Dual-Core is recommended)
- RAM: minimum 4GB (6GB recommended)
- Hard disk: 40GB hard disk

CHAPTER 3

SYSTEM DESIGN

3.1 Entity Relationship Diagram

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them. ER modeling helps to analyze data requirements systematically to produce a well-designed database. Entity relationship diagram displays the relationships of entity set stored in a database. In other words, ER diagrams help in explaining 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 the model unique.

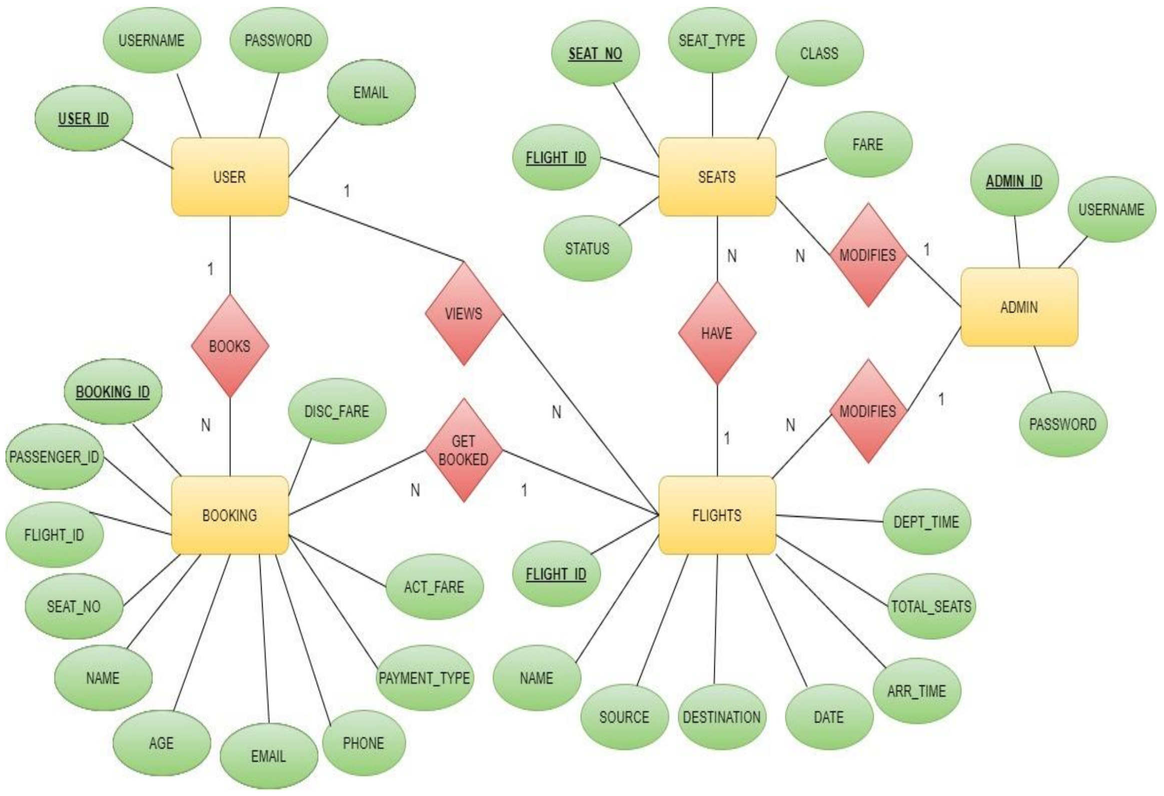


Figure 3.1 Entity Relationship Diagram

3.2 Schema Database Relationship Diagram

Database schema is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the data base, which can be depicted by means of schema diagrams.

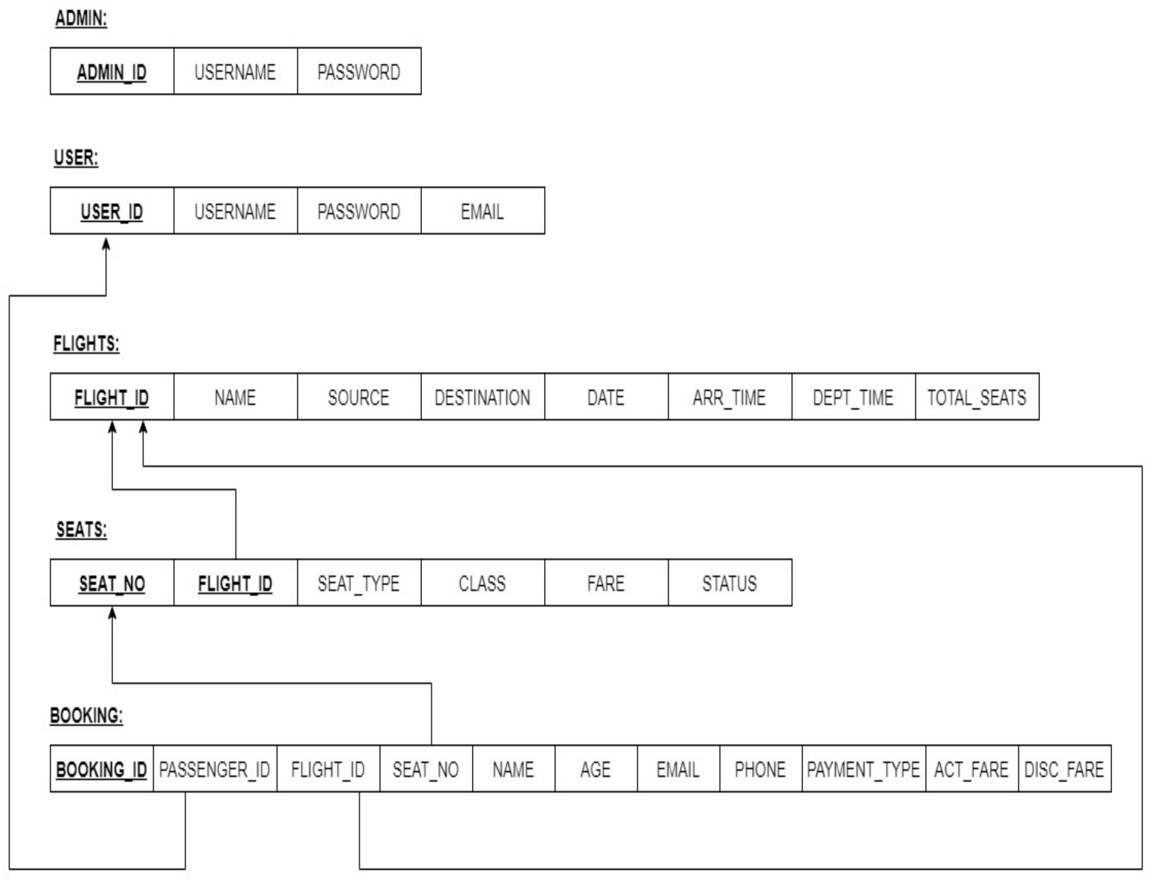


Figure 3.2 Schema Diagram

3.2 Overview of Graphical User Interface

GUI is program interface that takes advantages of the computer's graphics capabilities to make the program easier to user. Well-designed graphical user interfaces can free the user from learning complex command language. On the other hand, many users find that they work more efficiently with a command-driven interface, especially if they already know the command language.

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document and is applicable to rendering in speech, or on other media. CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

Node.js is an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

1) Admin Section Page:

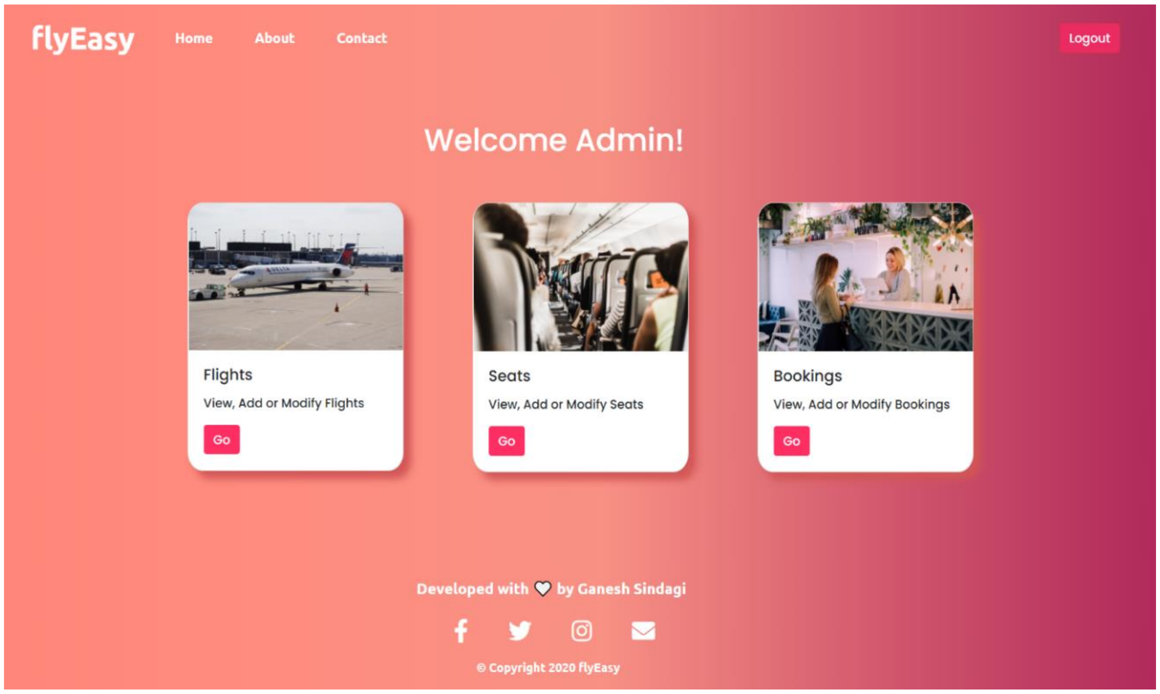
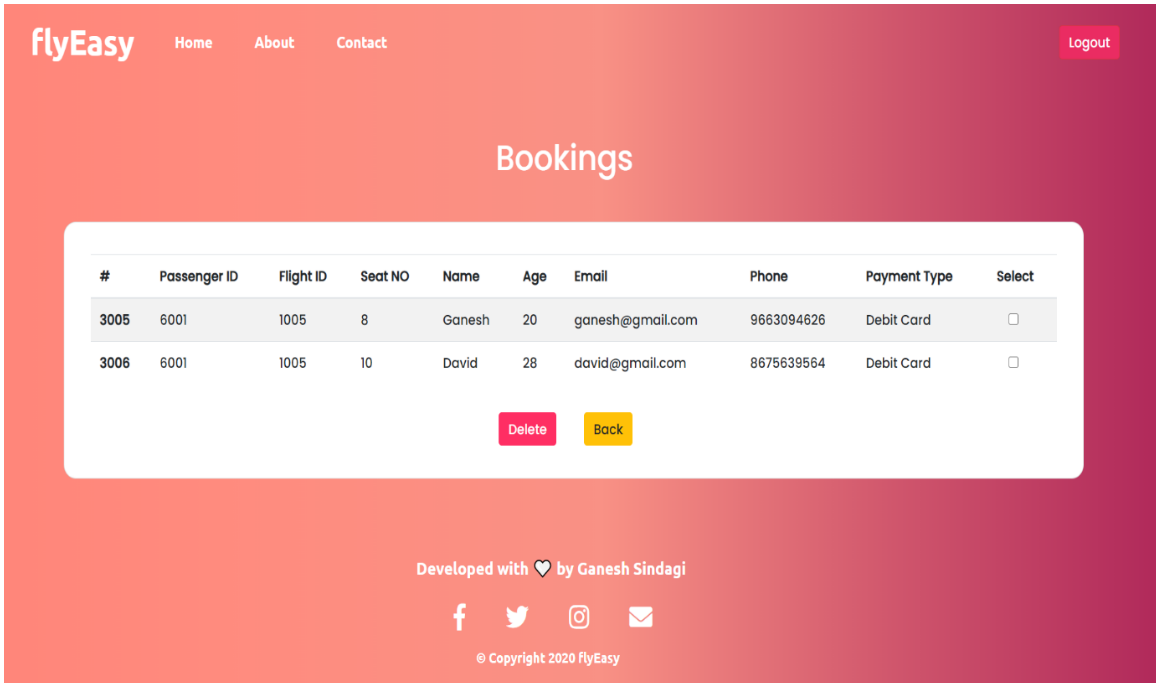

flyEasy
Home
About
Contact
Logout
Welcome Admin!
Flights
View, Add or Modify Flights
Go
Seats
View, Add or Modify Seats
Go
Bookings
View, Add or Modify Bookings
Go
Developed with ♡ by Ganesh Sindagi
© Copyright 2020 flyEasy

Figure 3.3: Admin Section Page.

2) Admin Flights Page:



#	Passenger ID	Flight ID	Seat NO	Name	Age	Email	Phone	Payment Type	Select
3005	6001	1005	8	Ganesh	20	ganesh@gmail.com	9663094626	Debit Card	☐
3006	6001	1005	10	David	28	david@gmail.com	8675639564	Debit Card	☐

Figure 3.4: Admin Flights Page.

3.4 Normalization

Normalization is a process of analyzing the given relation schema based on their functional dependencies and primary key to achieve desirable properties of minimizing redundancy and minimizing insert, delete, update anomaly. The normalization process takes a relation schema through a series of tests to certify whether it satisfies a certain normal form. The normal form of a relation refers to the highest normal form condition that it meets, and hence the degree to which it has been normalized.

There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as it reduces the amount of space a database consumes and ensure that data is logically stored.

3.4.1 1NF (First Normal Form):

1NF states that "the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute".

User_id	Username	Password	Email
6001	Ganesh	1234	ganesh@gmail.com
6002	John	3456	john@yahoo.com
6003	David	7896	david@gmail.com

Table 3.1 User table in 1NF

- The entity Admin also has atomic values; therefore, it is 1NF.
- The entity Flights also has atomic values; therefore, it is 1NF.
- The entity Seats also has atomic values; therefore, it is 1NF.
- The entity Booking also has atomic values; therefore, it is 1NF.

3.4.2 2NF (Second Normal Form):

Rules for 2NF:

- The table should be in 1NF
- Every Non-prime attribute should be fully functional dependent on the primary key.

Seat_no	**Flight_id**	Seat_type	Class	Fare	Status
1	3001	Recliner	Business	3457	Available
1	3002	Pushback	Economy	2453	Booked
2	3001	Euro Coach	Business	4564	Available

Table 3.2 Flights table in 2NF

- In Above Table Seat_no and Flight_id both determine each attribute class, fare and status.
- Hence table is fully functional dependent and it is 2nd Normal Form.

3.4.3 3NF (Third Normal Form):

Rules for 3NF:

- The table should be in 2NF
- Relation should have no transitive functional dependencies.

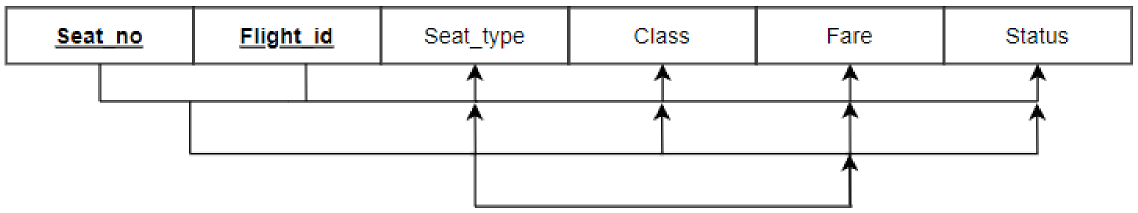
A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.

Consider the table Flights:

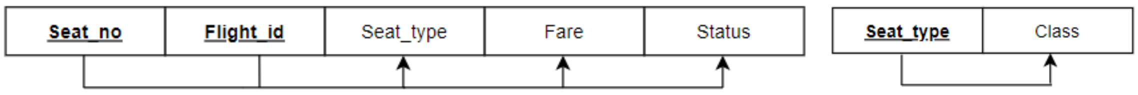
Seat_no	**Flight_id**	Seat_type	Class	Fare	Status
1	3001	Recliner	Business	3457	Available
1	3002	Pushback	Economy	2453	Booked
2	3001	Euro Coach	Business	4564	Available

Table 3.3 Flights table in 3NF

- Already it is shown above that the table Flights is in 2NF.
- It is observed that the table has transitive functional dependencies.
- Hence the Table is not in 3rd Normal Form.



- Seat_no and Flight_id determine Seat_type. Seat_type determines fare. Hence the table has transitive dependencies.
- Normalize the table by splitting into two relations.



<u>Seat_no</u>	**<u>Flight_id</u>**	Seat_type	Fare	Status
1	3001	Recliner	3457	Available
1	3002	Pushback	2453	Booked

Table 3.4 First Flights table in 3NF

<u>Seat_type</u>	Class
Recliner	Business
Pushback	Economy

Table 3.5 Second Flights table in 3NF

Chapter 4
IMPLEMENTATION

4.1 Table Creation

Admin:

```
CREATE TABLE airline.booking
(
   booking_id integer NOT NULL DEFAULT
   nextval('airline.booking_booking_id_seq'::regclass),
   passenger_id integer,
   flight_id integer,
   seat_no integer,
   name character varying(30) COLLATE pg_catalog."default",
   age integer,
   email character varying(40) COLLATE pg_catalog."default",
   phone character varying(15) COLLATE pg_catalog."default",
   payment_type character varying(20) COLLATE pg_catalog."default",
   act_fare integer,
   disc_fare integer,
   CONSTRAINT booking_pkey PRIMARY KEY (booking_id),
   CONSTRAINT "Bflight_id" FOREIGN KEY (flight_id)
      REFERENCES airline.flights (flight_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT "Bpassenger_id" FOREIGN KEY (passenger_id)
      REFERENCES airline."user" (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
```

User:

```
CREATE TABLE airline."user"
(
   user_id integer NOT NULL DEFAULT nextval('airline.user_user_id_seq'::regclass),
   username character varying(30) COLLATE pg_catalog."default",
   password character varying(200) COLLATE pg_catalog."default",
   email character varying(40) COLLATE pg_catalog."default",
   CONSTRAINT user_pkey PRIMARY KEY (user_id)
)
```

Flights:

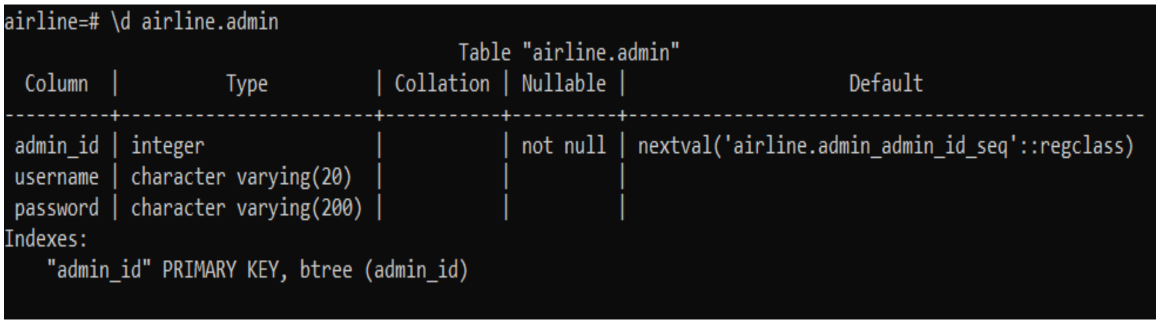
```
CREATE TABLE airline.flights
(
   flight_id integer NOT NULL DEFAULT
   nextval('airline.flights_flight_id_seq'::regclass),
   name character varying(20) COLLATE pg_catalog."default",
   source character varying(20) COLLATE pg_catalog."default",
   destination character varying(20) COLLATE pg_catalog."default",
   date date,
   duration time without time zone,
   dep_time time without time zone,
   arr_time time without time zone,
   fare integer,
   CONSTRAINT flights_pkey PRIMARY KEY (flight_id)
)
```

Booking:

```
CREATE TABLE airline.booking
(
   booking_id integer NOT NULL DEFAULT
   nextval('airline.booking_booking_id_seq'::regclass),
   passenger_id integer,
   flight_id integer,
   seat_no integer,
   name character varying(30) COLLATE pg_catalog."default",
   age integer,
   email character varying(40) COLLATE pg_catalog."default",
   phone character varying(15) COLLATE pg_catalog."default",
   payment_type character varying(20) COLLATE pg_catalog."default",
   act_fare integer,
   disc_fare integer,
   CONSTRAINT booking_pkey PRIMARY KEY (booking_id),
   CONSTRAINT "Bflight_id" FOREIGN KEY (flight_id)
      REFERENCES airline.flights (flight_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT "Bpassenger_id" FOREIGN KEY (passenger_id)
      REFERENCES airline."user" (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
```

4.2 Description of Tables

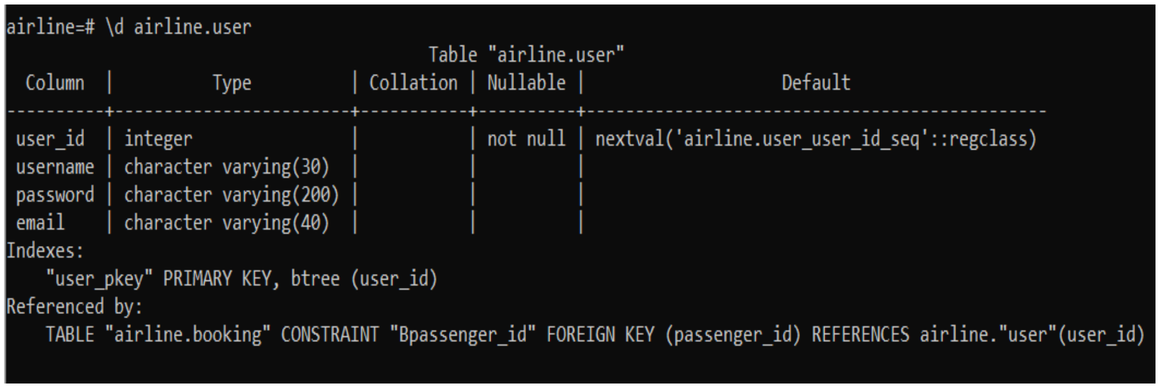
Admin:



```
airline=# \d airline.admin
                                    Table "airline.admin"
  Column  |          Type          | Collation | Nullable |                    Default
----------+------------------------+-----------+----------+-----------------------------------------------
 admin_id | integer                |           | not null | nextval('airline.admin_admin_id_seq'::regclass)
 username | character varying(20)  |           |          |
 password | character varying(200) |           |          |
Indexes:
    "admin_id" PRIMARY KEY, btree (admin_id)
```

Figure 4.1 Admin Description

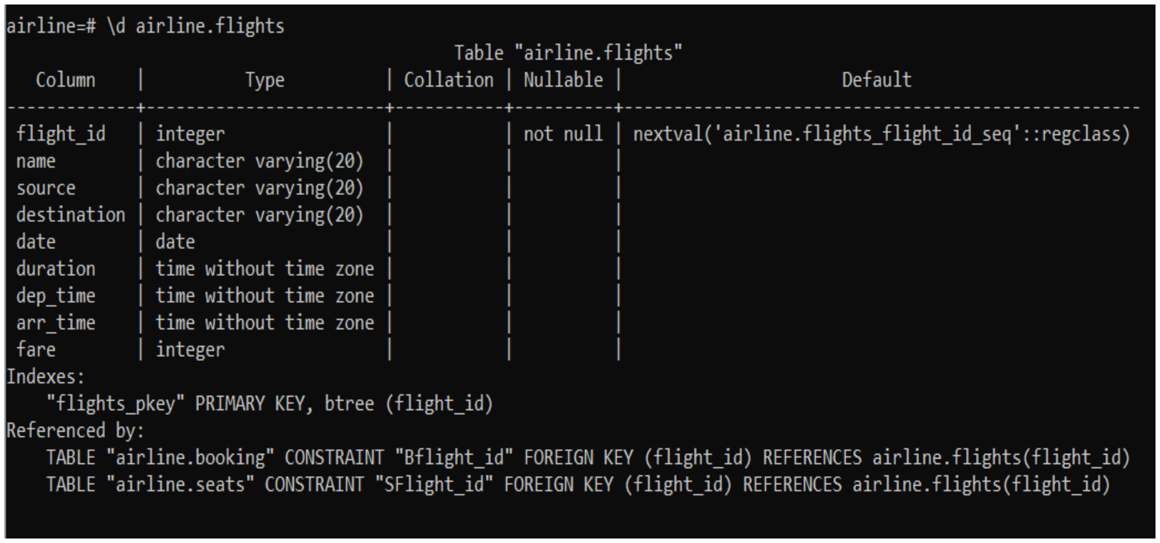
User:



```
airline=# \d airline.user
                                    Table "airline.user"
  Column  |          Type          | Collation | Nullable |                  Default
----------+------------------------+-----------+----------+---------------------------------------------
 user_id  | integer                |           | not null | nextval('airline.user_user_id_seq'::regclass)
 username | character varying(30)  |           |          |
 password | character varying(200) |           |          |
 email    | character varying(40)  |           |          |
Indexes:
    "user_pkey" PRIMARY KEY, btree (user_id)
Referenced by:
    TABLE "airline.booking" CONSTRAINT "Bpassenger_id" FOREIGN KEY (passenger_id) REFERENCES airline."user"(user_id)
```

Figure 4.2 User Description

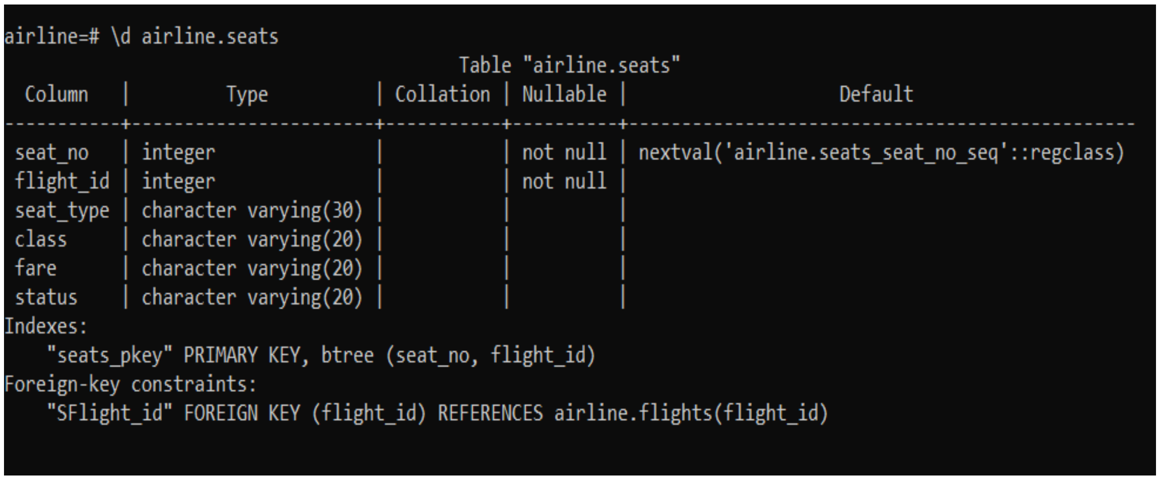
Flights:



```
airline=# \d airline.flights
                                    Table "airline.flights"
   Column    |          Type          | Collation | Nullable |                    Default
-------------+------------------------+-----------+----------+---------------------------------------------------
 flight_id   | integer                |           | not null | nextval('airline.flights_flight_id_seq'::regclass)
 name        | character varying(20)  |           |          |
 source      | character varying(20)  |           |          |
 destination | character varying(20)  |           |          |
 date        | date                   |           |          |
 duration    | time without time zone |           |          |
 dep_time    | time without time zone |           |          |
 arr_time    | time without time zone |           |          |
 fare        | integer                |           |          |
Indexes:
    "flights_pkey" PRIMARY KEY, btree (flight_id)
Referenced by:
    TABLE "airline.booking" CONSTRAINT "Bflight_id" FOREIGN KEY (flight_id) REFERENCES airline.flights(flight_id)
    TABLE "airline.seats" CONSTRAINT "SFlight_id" FOREIGN KEY (flight_id) REFERENCES airline.flights(flight_id)
```

Figure 4.3 Flights Description

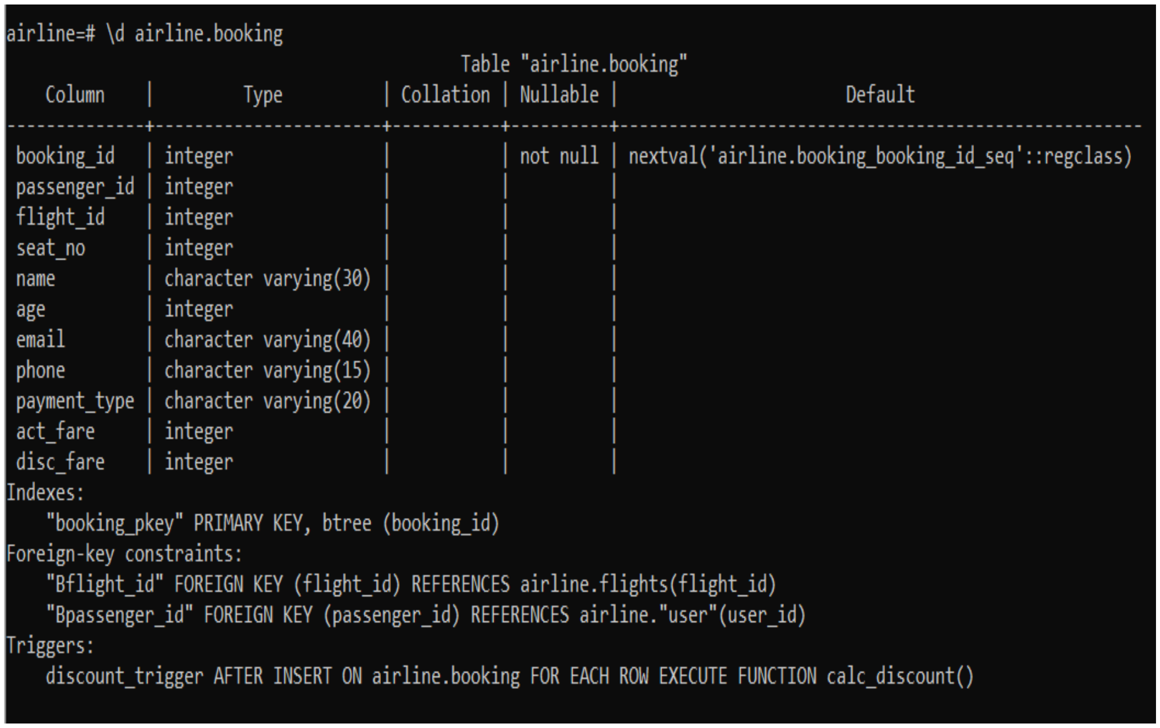
Seats:



```
airline=# \d airline.seats
                                          Table "airline.seats"
  Column   |         Type          | Collation | Nullable |                    Default
-----------+-----------------------+-----------+----------+-----------------------------------------------
 seat_no   | integer               |           | not null | nextval('airline.seats_seat_no_seq'::regclass)
 flight_id | integer               |           | not null |
 seat_type | character varying(30) |           |          |
 class     | character varying(20) |           |          |
 fare      | character varying(20) |           |          |
 status    | character varying(20) |           |          |
Indexes:
    "seats_pkey" PRIMARY KEY, btree (seat_no, flight_id)
Foreign-key constraints:
    "SFlight_id" FOREIGN KEY (flight_id) REFERENCES airline.flights(flight_id)
```

Figure 4.4 Seats Description

Booking:

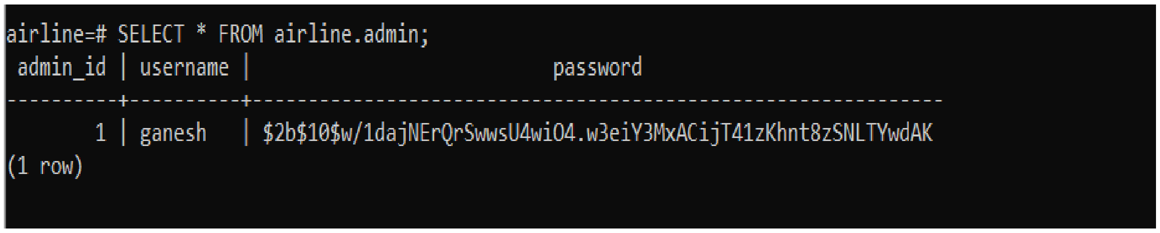


```
airline=# \d airline.booking
                                          Table "airline.booking"
    Column    |         Type          | Collation | Nullable |                     Default
--------------+-----------------------+-----------+----------+----------------------------------------------------
 booking_id   | integer               |           | not null | nextval('airline.booking_booking_id_seq'::regclass)
 passenger_id | integer               |           |          |
 flight_id    | integer               |           |          |
 seat_no      | integer               |           |          |
 name         | character varying(30) |           |          |
 age          | integer               |           |          |
 email        | character varying(40) |           |          |
 phone        | character varying(15) |           |          |
 payment_type | character varying(20) |           |          |
 act_fare     | integer               |           |          |
 disc_fare    | integer               |           |          |
Indexes:
    "booking_pkey" PRIMARY KEY, btree (booking_id)
Foreign-key constraints:
    "Bflight_id" FOREIGN KEY (flight_id) REFERENCES airline.flights(flight_id)
    "Bpassenger_id" FOREIGN KEY (passenger_id) REFERENCES airline."user"(user_id)
Triggers:
    discount_trigger AFTER INSERT ON airline.booking FOR EACH ROW EXECUTE FUNCTION calc_discount()
```

Figure 4.5 Booking Description

4.3 Populated Tables

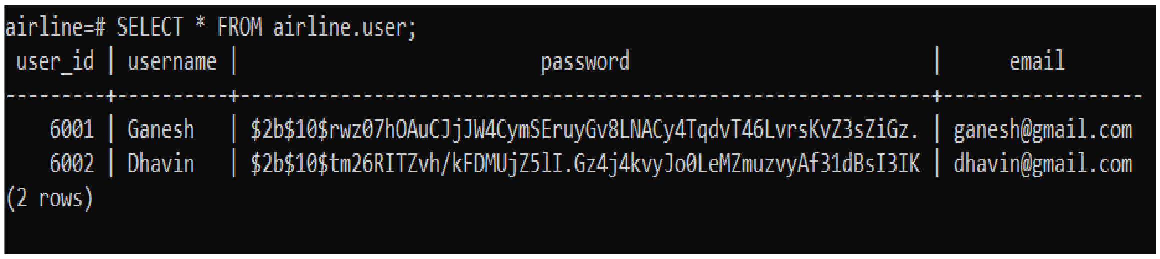
Admin:



```
airline=# SELECT * FROM airline.admin;
 admin_id | username |                          password
----------+----------+--------------------------------------------------------------
        1 | ganesh   | $2b$10$w/1dajNErQrSwwsU4wiO4.w3eiY3MxACijT41zKhnt8zSNLTYwdAK
(1 row)
```

Figure 4.6 Admin Table Values

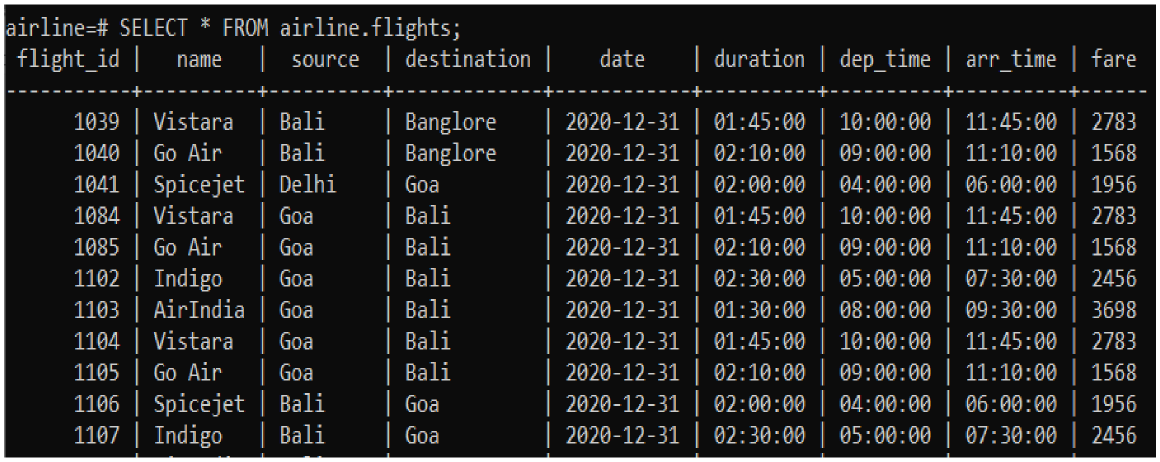
User:



```
airline=# SELECT * FROM airline.user;
 user_id | username |                           password                           |      email
---------+----------+--------------------------------------------------------------+------------------
    6001 | Ganesh   | $2b$10$rwz07hOAuCJjJW4CymSEruyGv8LNACy4TqdvT46LvrsKvZ3sZiGz. | ganesh@gmail.com
    6002 | Dhavin   | $2b$10$tm26RITZvh/kFDMUjZ5lI.Gz4j4kvyJo0LeMZmuzvyAf31dBsI3IK | dhavin@gmail.com
(2 rows)
```

Figure 4.7 User Table Values

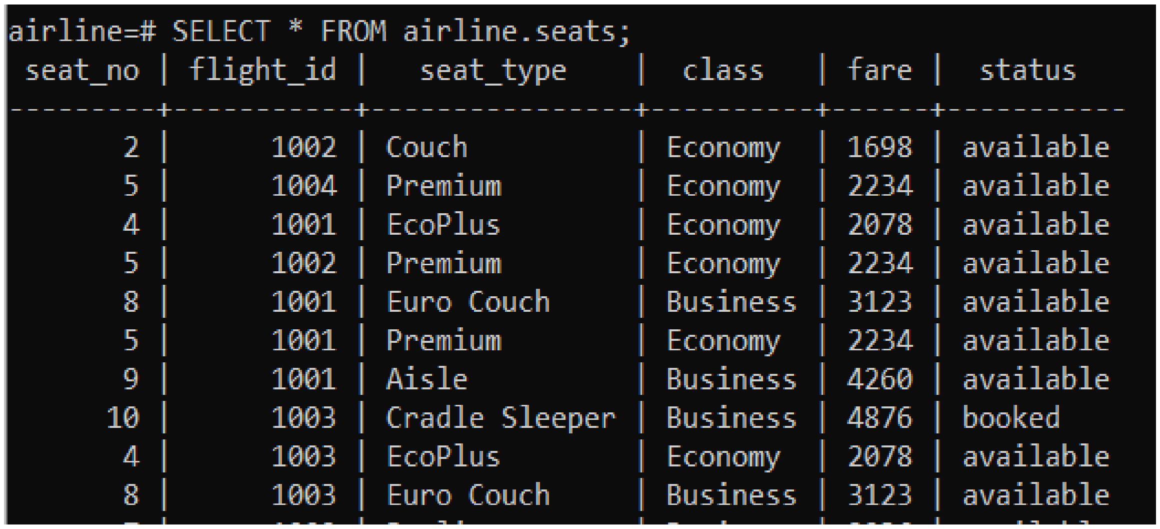
Flights:



```
airline=# SELECT * FROM airline.flights;
 flight_id |   name   |  source  | destination |    date    | duration | dep_time | arr_time | fare
-----------+----------+----------+-------------+------------+----------+----------+----------+------
      1039 | Vistara  | Bali     | Banglore    | 2020-12-31 | 01:45:00 | 10:00:00 | 11:45:00 | 2783
      1040 | Go Air   | Bali     | Banglore    | 2020-12-31 | 02:10:00 | 09:00:00 | 11:10:00 | 1568
      1041 | Spicejet | Delhi    | Goa         | 2020-12-31 | 02:00:00 | 04:00:00 | 06:00:00 | 1956
      1084 | Vistara  | Goa      | Bali        | 2020-12-31 | 01:45:00 | 10:00:00 | 11:45:00 | 2783
      1085 | Go Air   | Goa      | Bali        | 2020-12-31 | 02:10:00 | 09:00:00 | 11:10:00 | 1568
      1102 | Indigo   | Goa      | Bali        | 2020-12-31 | 02:30:00 | 05:00:00 | 07:30:00 | 2456
      1103 | AirIndia | Goa      | Bali        | 2020-12-31 | 01:30:00 | 08:00:00 | 09:30:00 | 3698
      1104 | Vistara  | Goa      | Bali        | 2020-12-31 | 01:45:00 | 10:00:00 | 11:45:00 | 2783
      1105 | Go Air   | Goa      | Bali        | 2020-12-31 | 02:10:00 | 09:00:00 | 11:10:00 | 1568
      1106 | Spicejet | Bali     | Goa         | 2020-12-31 | 02:00:00 | 04:00:00 | 06:00:00 | 1956
      1107 | Indigo   | Bali     | Goa         | 2020-12-31 | 02:30:00 | 05:00:00 | 07:30:00 | 2456
```

Figure 4.8 Flights Table Values

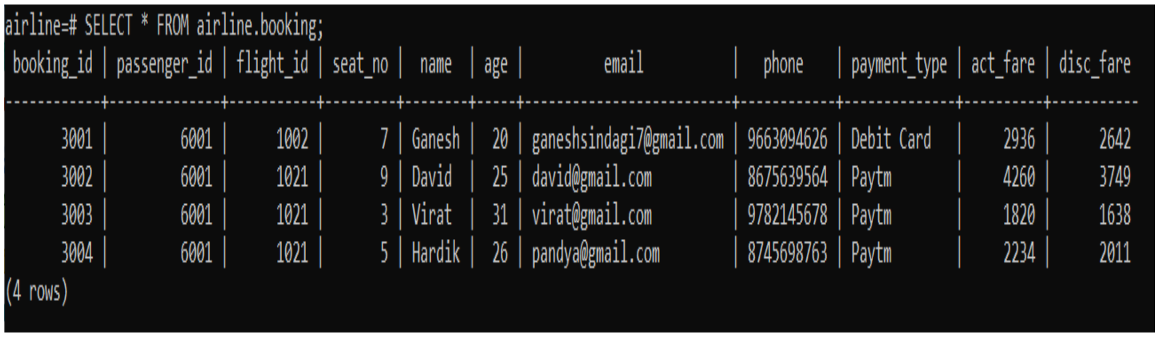
Seats:



```
airline=# SELECT * FROM airline.seats;
 seat_no | flight_id |   seat_type    |  class   | fare |  status
---------+-----------+----------------+----------+------+-----------
       2 |      1002 | Couch          | Economy  | 1698 | available
       5 |      1004 | Premium        | Economy  | 2234 | available
       4 |      1001 | EcoPlus        | Economy  | 2078 | available
       5 |      1002 | Premium        | Economy  | 2234 | available
       8 |      1001 | Euro Couch     | Business | 3123 | available
       5 |      1001 | Premium        | Economy  | 2234 | available
       9 |      1001 | Aisle          | Business | 4260 | available
      10 |      1003 | Cradle Sleeper | Business | 4876 | booked
       4 |      1003 | EcoPlus        | Economy  | 2078 | available
       8 |      1003 | Euro Couch     | Business | 3123 | available
```

Figure 4.9 Seats Table Values

Booking:



```
airline=# SELECT * FROM airline.booking;
 booking_id | passenger_id | flight_id | seat_no |  name  | age |         email          |   phone    | payment_type | act_fare | disc_fare
------------+--------------+-----------+---------+--------+-----+------------------------+------------+--------------+----------+-----------
       3001 |         6001 |      1002 |       7 | Ganesh |  20 | ganeshsindagi7@gmail.com | 9663094626 | Debit Card   |     2936 |      2642
       3002 |         6001 |      1021 |       9 | David  |  25 | david@gmail.com        | 8675639564 | Paytm        |     4260 |      3749
       3003 |         6001 |      1021 |       3 | Virat  |  31 | virat@gmail.com        | 9782145678 | Paytm        |     1820 |      1638
       3004 |         6001 |      1021 |       5 | Hardik |  26 | pandya@gmail.com       | 8745698763 | Paytm        |     2234 |      2011
(4 rows)
```

Figure 4.10 Booking Table Values

4.4 SQL Triggers and Stored Procedures

4.4.1 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. BEFORE triggers run the trigger action before the triggering statement is run. AFTER triggers run the trigger action after the triggering statement is run.

The trigger used in the application is – After passenger has book the seat, the data is inserted into the booking table. Insertion of each row to the booking table triggers the **discount_trigger** which then calls the stored procedure **calc_discount**.

```
CREATE TRIGGER discount_trigger
   AFTER INSERT
   ON airline.booking
   FOR EACH ROW
   EXECUTE PROCEDURE public.calc_discount();
```

4.4.2 Stored Procedure:

A stored procedure is a prepared SQL code that can be saved and can be reused over and over again. So if a query has to be written over and over again, instead of having to write that query each time, it can be saved as a stored procedure and can be executed just by calling the procedure. In addition, parameters can also be passed to the stored procedure So depending on the need, the stored procedure can act accordingly. Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. It reduces duplication of effortand improves software modularity.
- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.

- These procedures can enhance the modelling power provided by views by allowing, more complex types of derived data to be made available to the database users via the stored procedures. Additionally, it can be used to check for complex constraints that are beyond the specification power of assertions and triggers.

The stored procedure used in the application is **calc_discount**, it calculates the discount fare for each of the booking made by the passenger. If the fare amount is greater than or equal to ₹5000 then 15% discount is applied on the actual fare. If the fare amount is greater than or equal to ₹3000 then 12% discount is applied on the actual fare. Also for every ticket which has fare less than ₹3000 will get a 10% discount on the actual fare.

```
CREATE FUNCTION public.calc_discount()
   RETURNS trigger
   LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
     IF NEW.act_fare >= 5000 THEN
          UPDATE airline.booking SET disc_fare = (act_fare - (act_fare*0.15)) WHERE
          booking_id = new.booking_id;
          RETURN NEW;
     ELSEIF NEW.act_fare >= 3000 THEN
          UPDATE airline.booking SET disc_fare = (act_fare - (act_fare*0.12)) WHERE
          booking_id = new.booking_id;
          RETURN NEW;
     ELSE
          UPDATE airline.booking SET disc_fare = (act_fare - (act_fare*0.1)) WHERE
          booking_id = new.booking_id;
          RETURN NEW;
     END IF;
     END;
$BODY$;
```

4.5 Database Connectivity

In computer science, a database connection is the means by which a database server and its client software communicate with each other. The term is used whether the client and the server are on different machines. The client uses a database connection to send commands to and receive replies from the server. A database is stored as a file or a set of files on magnetic disk or tape, optical disk, or some other secondary storage device. The information in these files may be broken down into records, each of which consists of one or more fields.

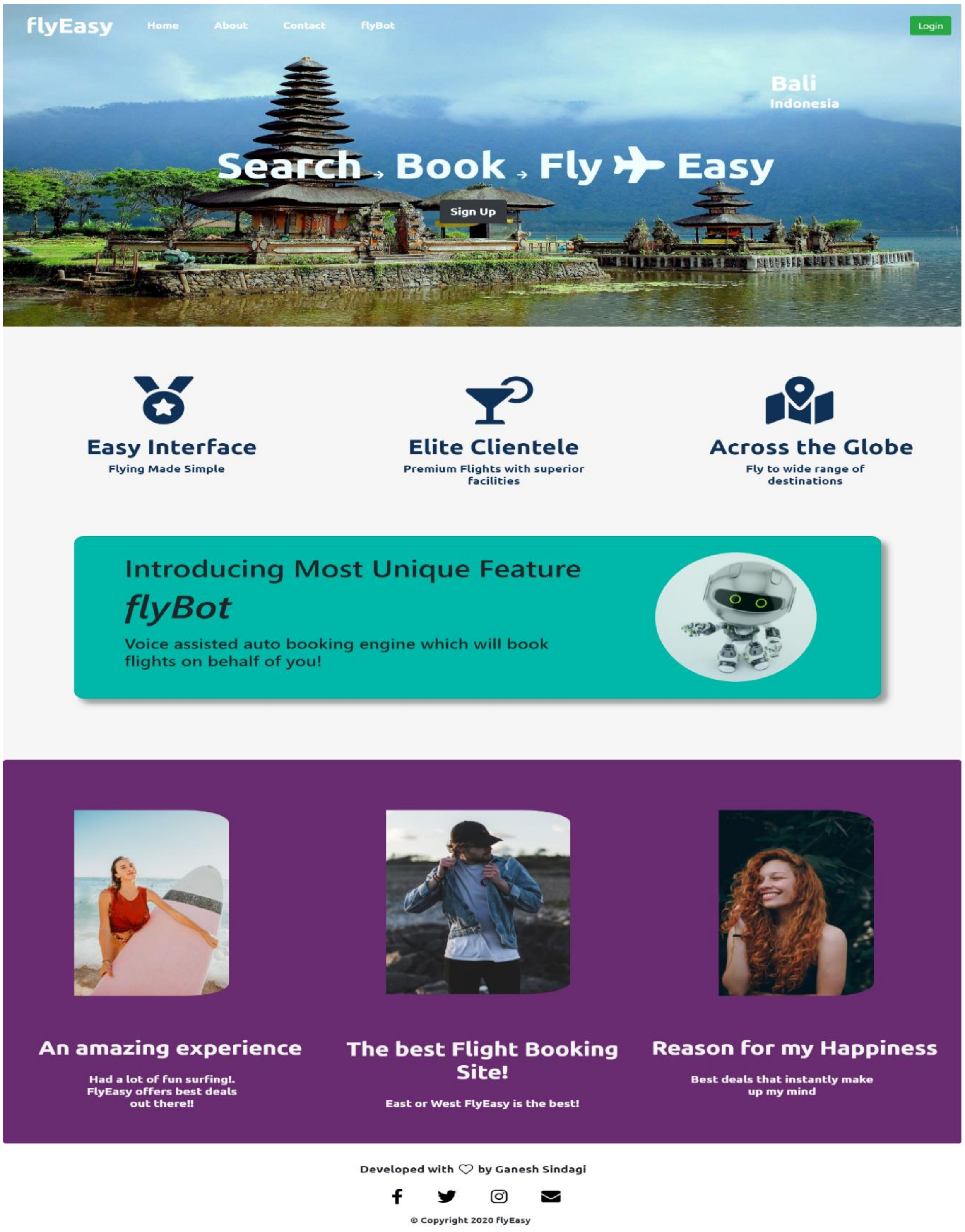
Using npm module called ‘pg’ establishes a connection to the database (i.e. PostgreSQL) for the backend express app. The connection is achieved by writing following lines of code.

```
var connectionString = “postgres://postgres:ganesh@localhost:5432/airline”;
const client = new Client({
    connectionString: connectionString
});
client.connect();
```

Chapter 5

RESULTS

Home page for airline reservation system: flyEasy



flyEasy Home About Contact flyBot Login

Bali
Indonesia

Search → Book → Fly ✈ Easy

Sign Up

Easy Interface
Flying Made Simple

Elite Clientele
Premium Flights with superior facilities

Across the Globe
Fly to wide range of destinations

Introducing Most Unique Feature
flyBot
Voice assisted auto booking engine which will book flights on behalf of you!

An amazing experience
Had a lot of fun surfing!. FlyEasy offers best deals out there!!

The best Flight Booking Site!
East or West FlyEasy is the best!

Reason for my Happiness
Best deals that instantly make up my mind

Developed with ♡ by Ganesh Sindagi

© Copyright 2020 flyEasy

Figure 5.1: Home page.

Signup page for new users

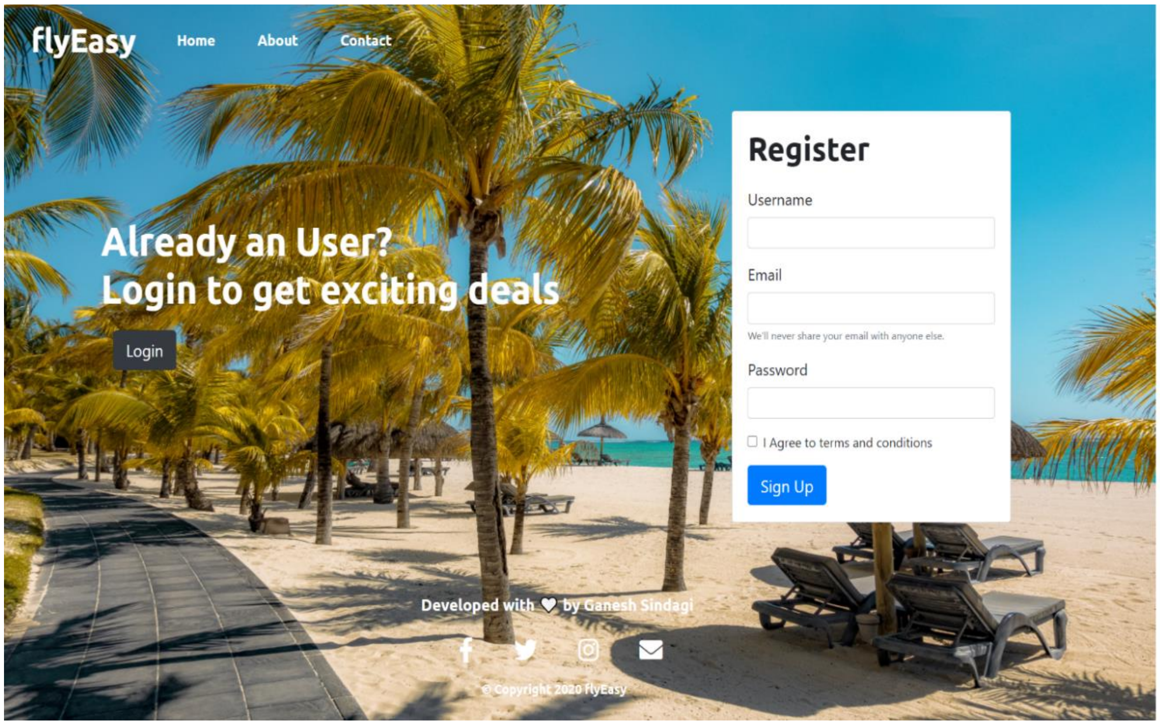

flyEasy
Home
About
Contact
Already an User?
Login to get exciting deals
Login
Register
Username
Email
We'll never share your email with anyone else.
Password
I Agree to terms and conditions
Sign Up
Developed with ♥ by Ganesh Sindagi
© Copyright 2020 flyEasy

Figure 5.2: User Registration Page.

User login page

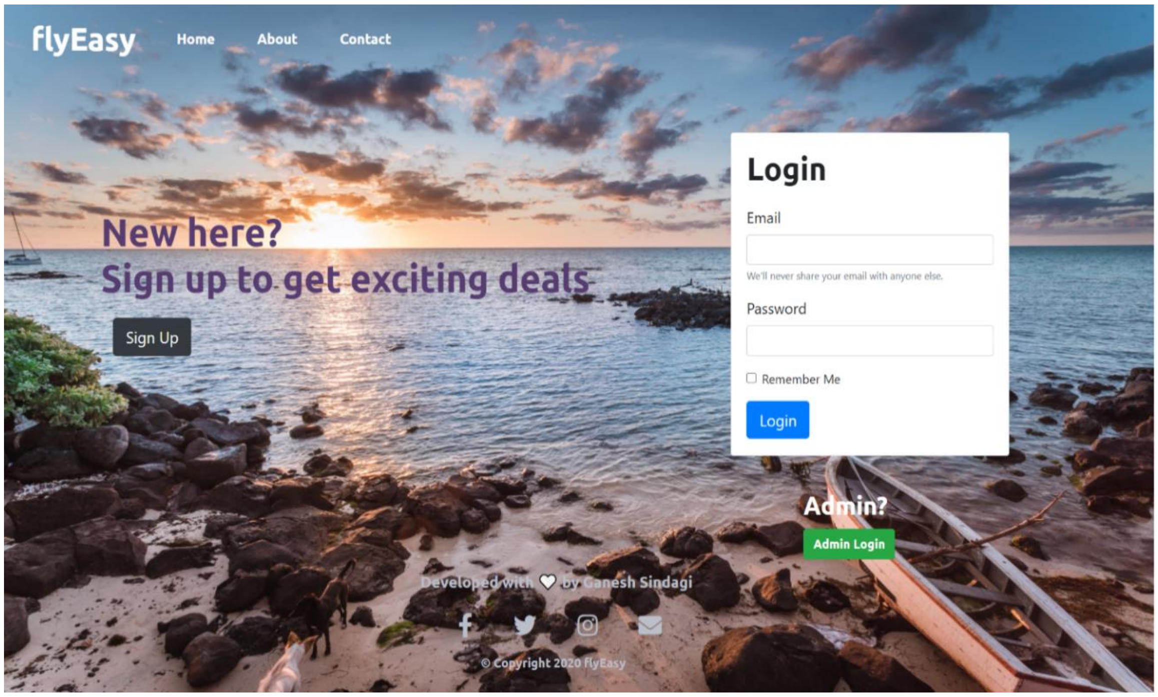


Figure 5.3: User Login Page.

Flight Search Page

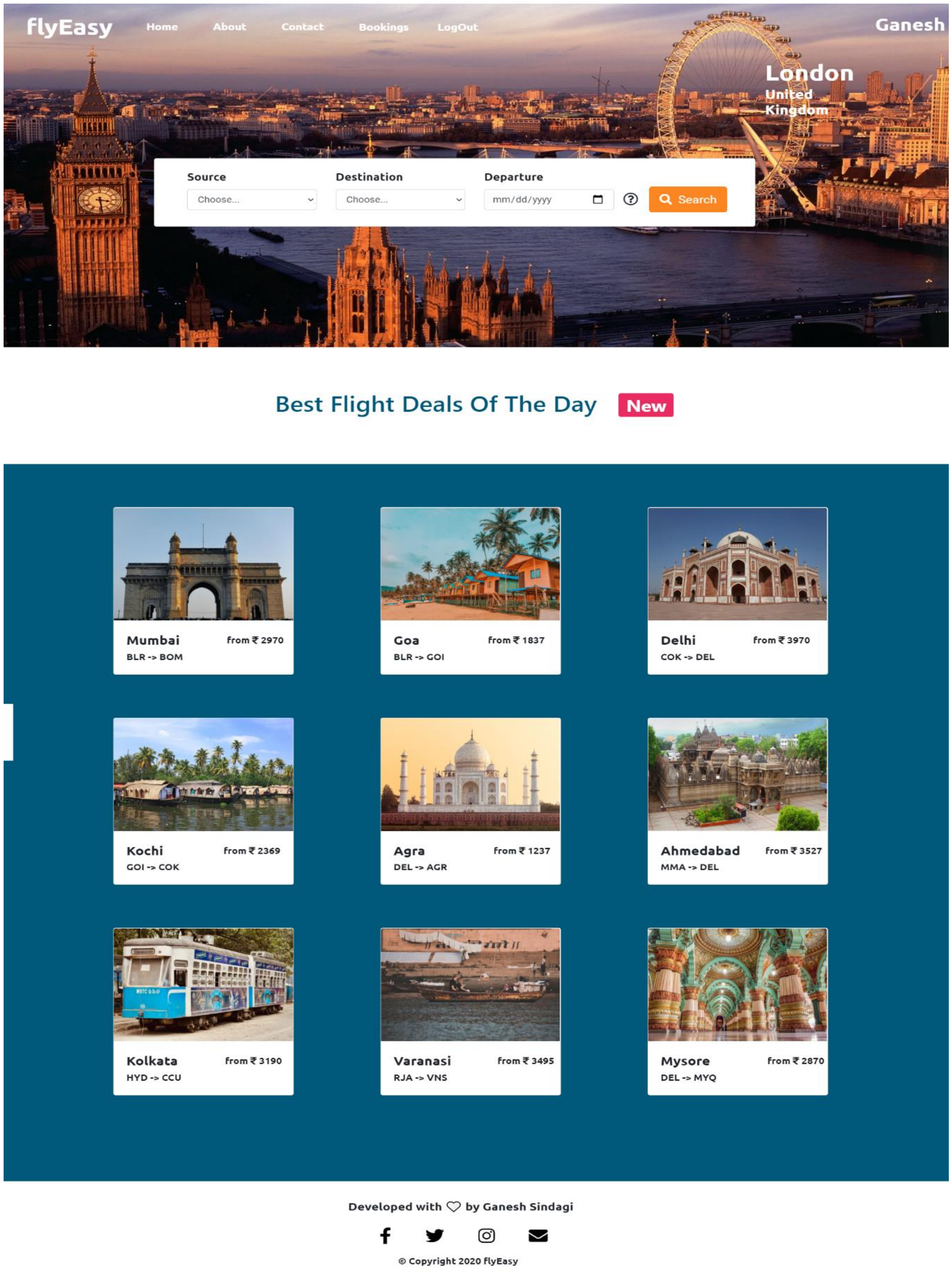
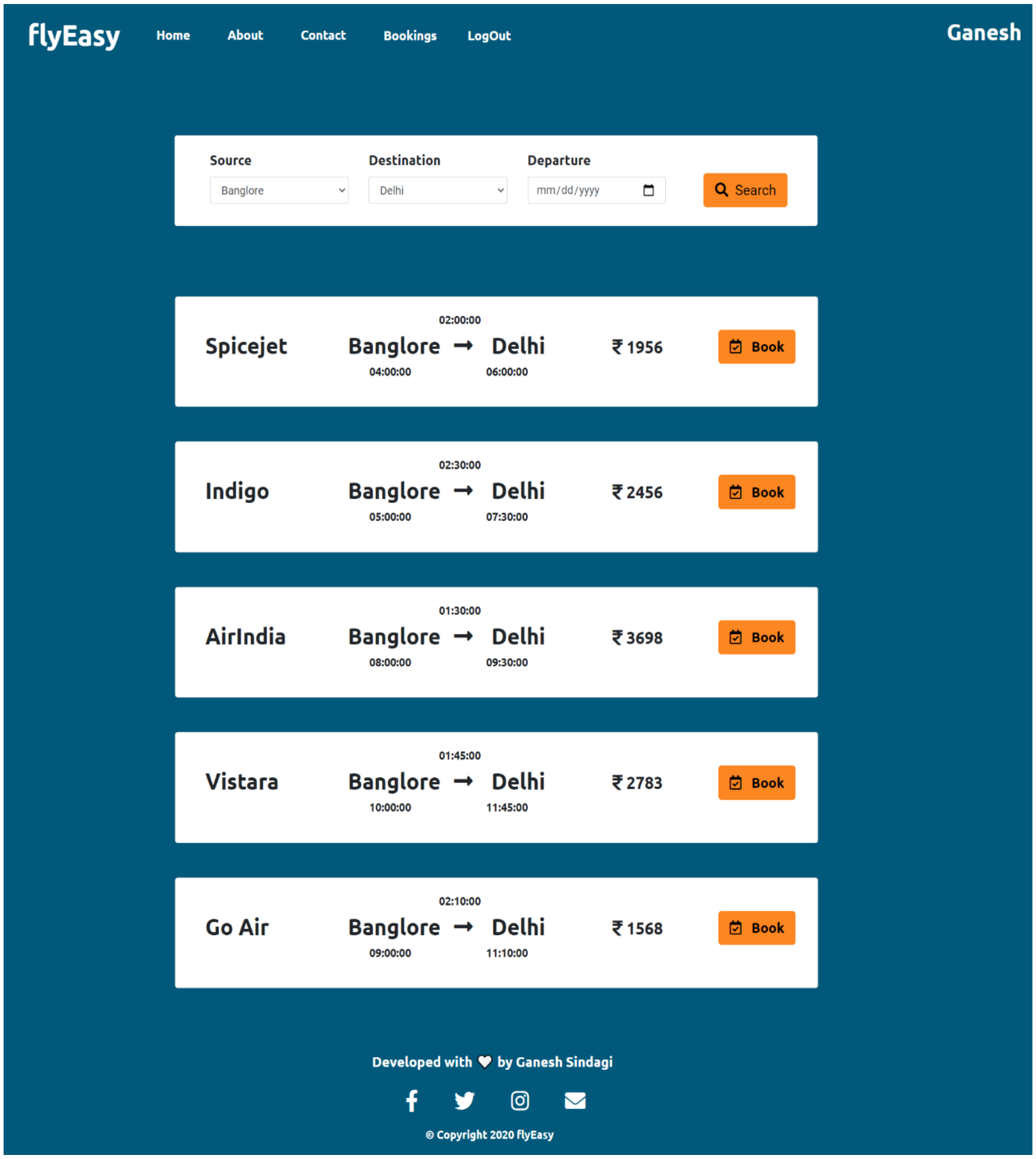

flyEasy
Home
About
Contact
Bookings
LogOut
Ganesh
London
United Kingdom
Source
Choose...
Destination
Choose...
Departure
mm/dd/yyyy
Search
Best Flight Deals Of The Day
New
Mumbai from ₹ 2970 BLR -> BOM
Goa from ₹ 1837 BLR -> GOI
Delhi from ₹ 3970 COK -> DEL
Kochi from ₹ 2369 GOI -> COK
Agra from ₹ 1237 DEL -> AGR
Ahmedabad from ₹ 3527 MMA -> DEL
Kolkata from ₹ 3190 HYD -> CCU
Varanasi from ₹ 3495 RJA -> VNS
Mysore from ₹ 2870 DEL -> MYQ
Developed with ♡ by Ganesh Sindagi
© Copyright 2020 FlyEasy

Figure 5.4: Flight Search Page.

Available Flights Page



Source: Banglore
Destination: Delhi
Departure: mm/dd/yyyy

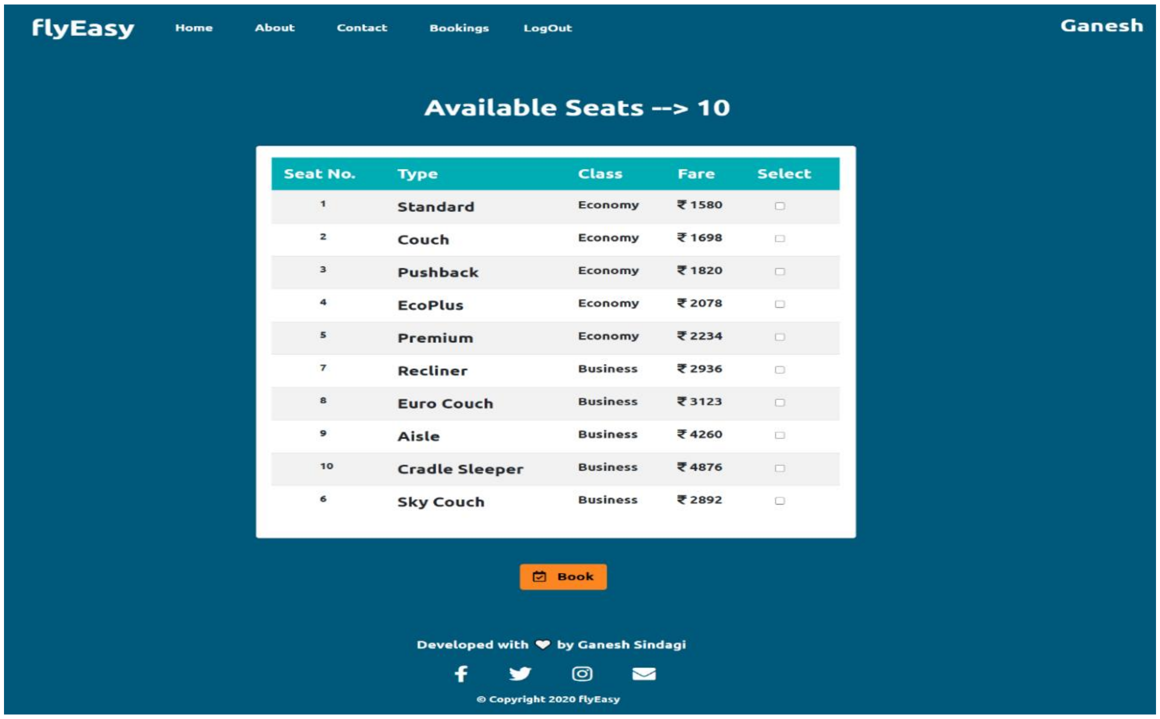
Spicejet	Banglore 04:00:00	→	Delhi 06:00:00	02:00:00	₹ 1956	Book
Indigo	Banglore 05:00:00	→	Delhi 07:30:00	02:30:00	₹ 2456	Book
AirIndia	Banglore 08:00:00	→	Delhi 09:30:00	01:30:00	₹ 3698	Book
Vistara	Banglore 10:00:00	→	Delhi 11:45:00	01:45:00	₹ 2783	Book
Go Air	Banglore 09:00:00	→	Delhi 11:10:00	02:10:00	₹ 1568	Book

Developed with ♥ by Ganesh Sindagi

© Copyright 2020 flyEasy

Figure 5.5: Available Flights Page.

Available Seats Page

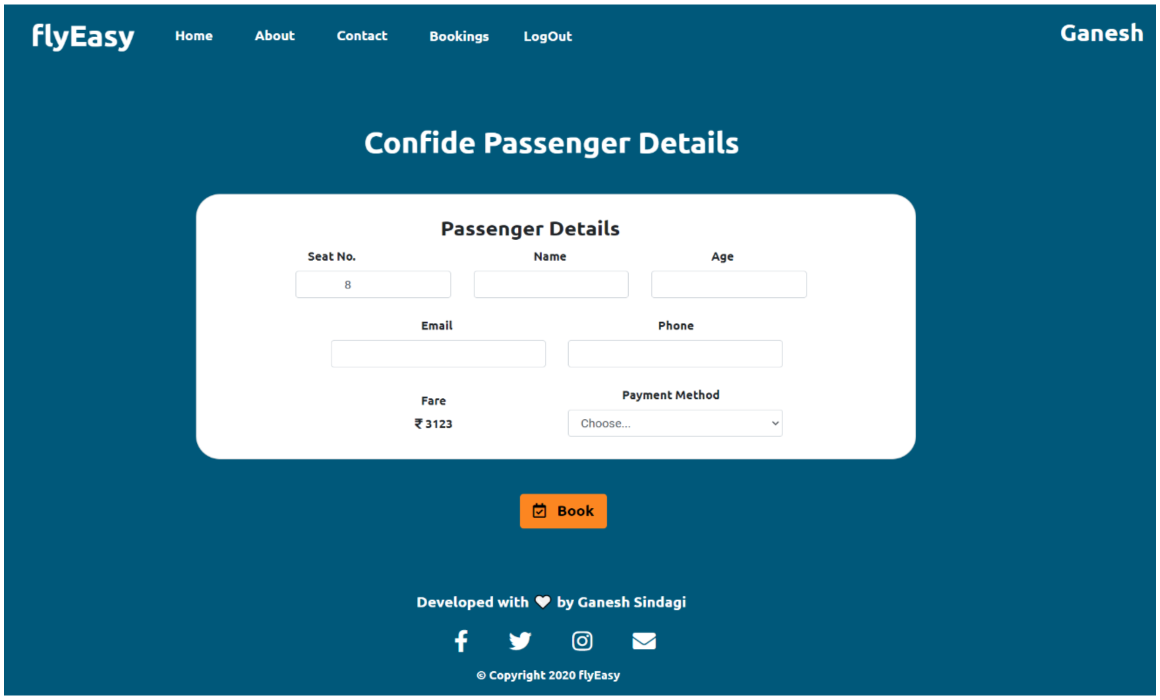


Available Seats --> 10

Seat No.	Type	Class	Fare	Select
1	Standard	Economy	₹ 1580	☐
2	Couch	Economy	₹ 1698	☐
3	Pushback	Economy	₹ 1820	☐
4	EcoPlus	Economy	₹ 2078	☐
5	Premium	Economy	₹ 2234	☐
7	Recliner	Business	₹ 2936	☐
8	Euro Couch	Business	₹ 3123	☐
9	Aisle	Business	₹ 4260	☐
10	Cradle Sleeper	Business	₹ 4876	☐
6	Sky Couch	Business	₹ 2892	☐

Figure 5.6: Available Seats Page.

Passenger Details Page



Confide Passenger Details

Passenger Details

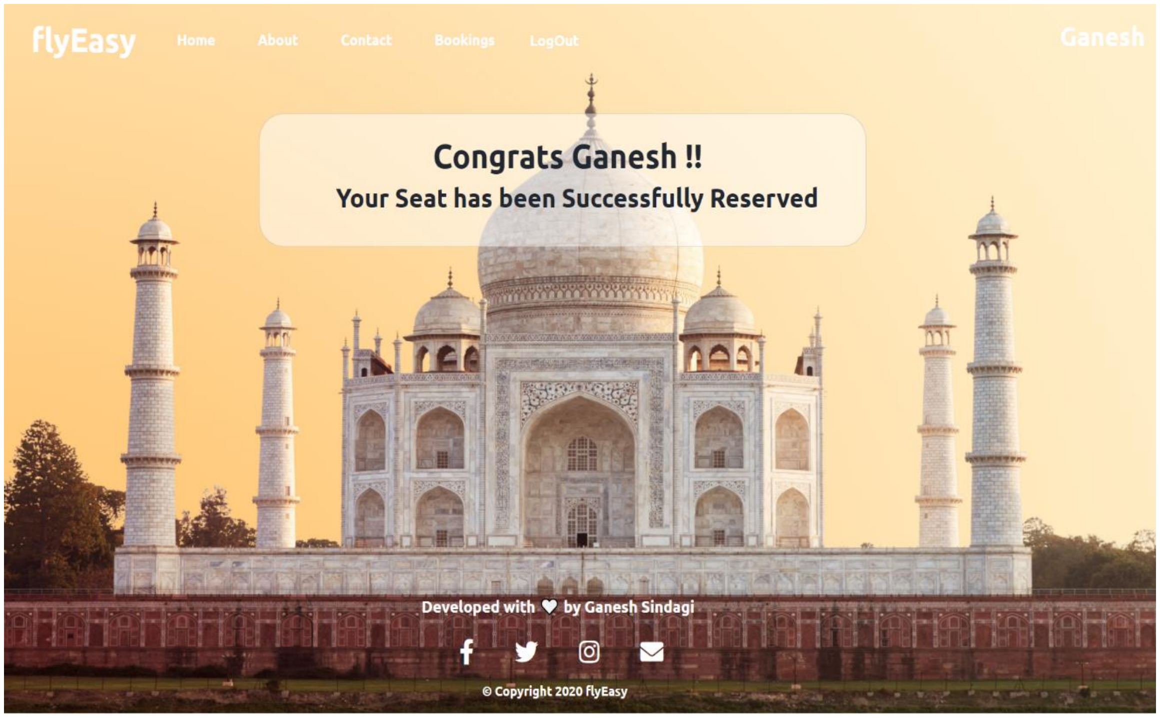
Seat No.	Name	Age
8		

Email	Phone

Fare	Payment Method
₹ 3123	Choose...

Figure 5.7: Passenger Details Page.

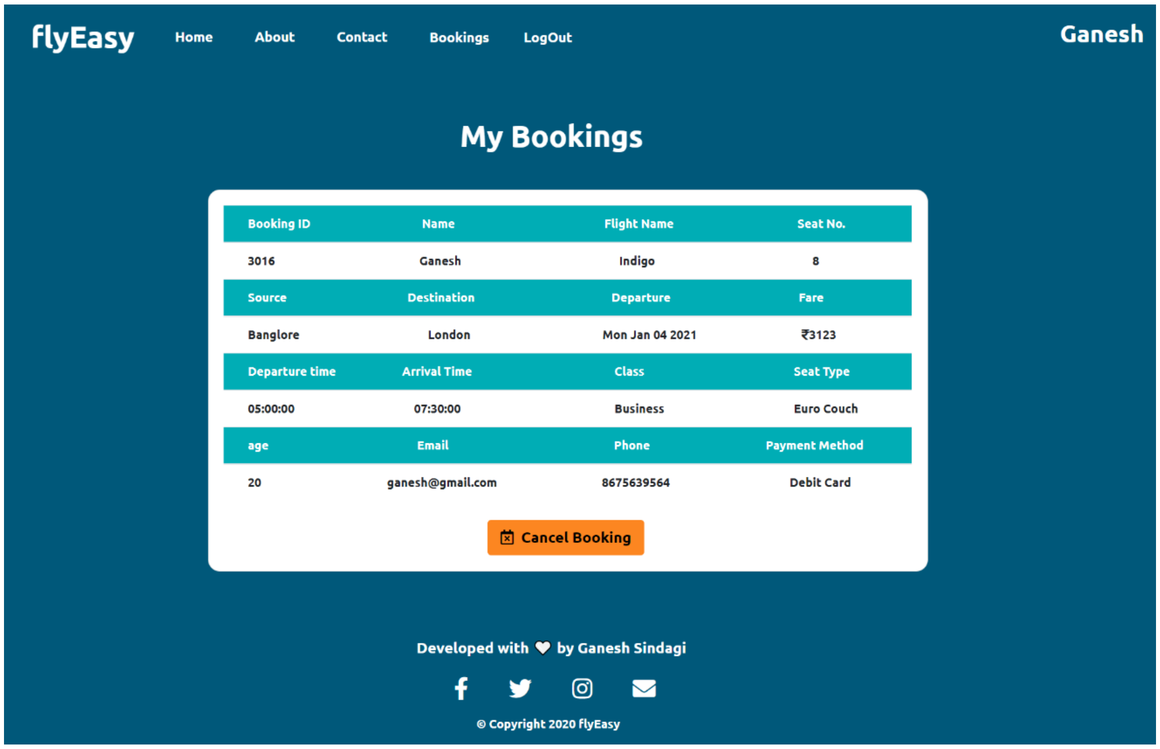
Booking Success Page



Congrats Ganesh !!
Your Seat has been Successfully Reserved

Figure 5.8: Booking Success Page.

User Bookings Page



My Bookings

Booking ID	Name	Flight Name	Seat No.
3016	Ganesh	Indigo	8
Source	**Destination**	**Departure**	**Fare**
Banglore	London	Mon Jan 04 2021	₹3123
Departure time	**Arrival Time**	**Class**	**Seat Type**
05:00:00	07:30:00	Business	Euro Couch
age	**Email**	**Phone**	**Payment Method**
20	ganesh@gmail.com	8675639564	Debit Card

Figure 5.9: User Bookings Page.

Admin Section page

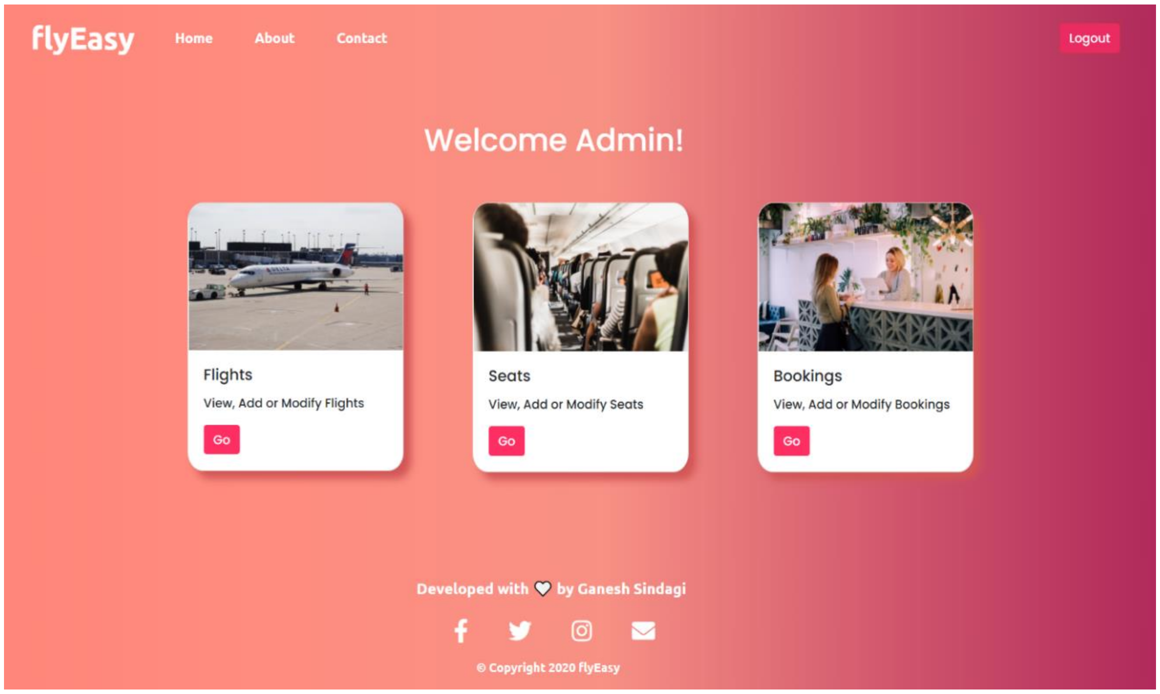
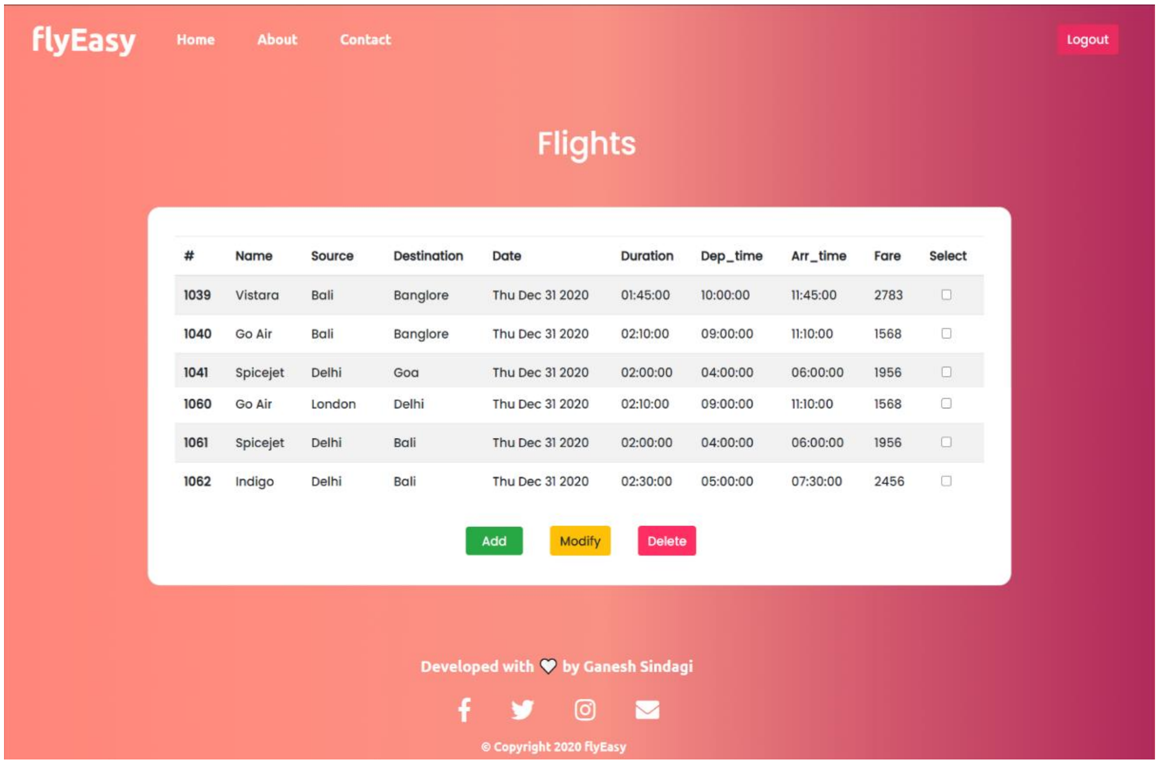

flyEasy
Home
About
Contact
Logout
Welcome Admin!
Flights
View, Add or Modify Flights
Go
Seats
View, Add or Modify Seats
Go
Bookings
View, Add or Modify Bookings
Go
Developed with ♡ by Ganesh Sindagi
© Copyright 2020 flyEasy

Figure 5.10: Admin Section Page.

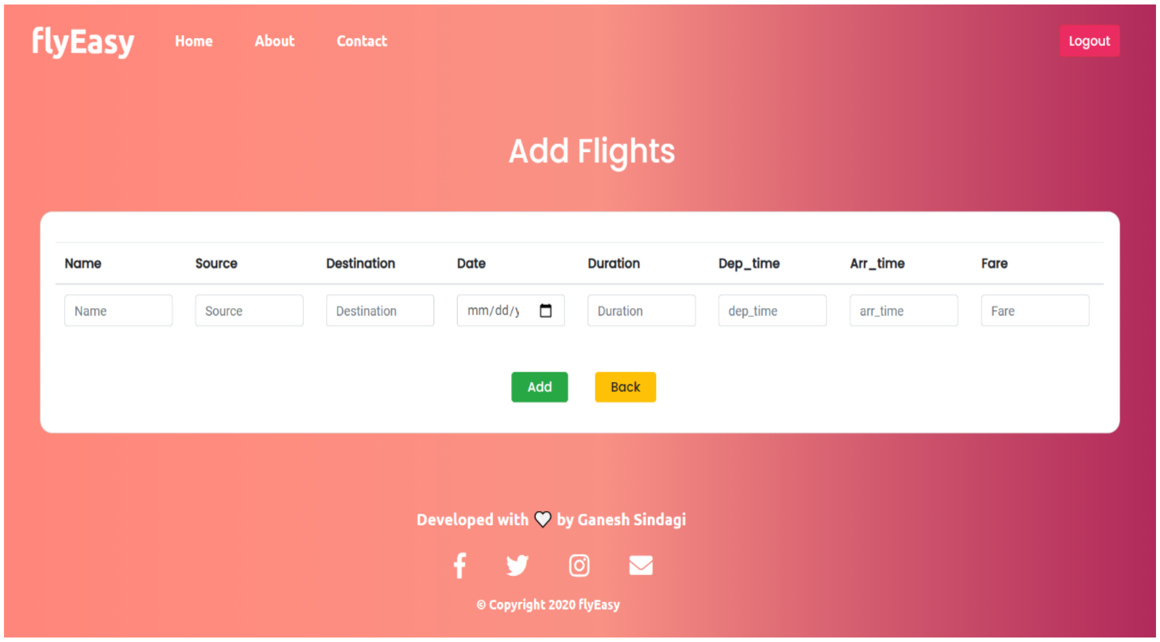
Admin Flights page



#	Name	Source	Destination	Date	Duration	Dep_time	Arr_time	Fare	Select
1039	Vistara	Bali	Banglore	Thu Dec 31 2020	01:45:00	10:00:00	11:45:00	2783	☐
1040	Go Air	Bali	Banglore	Thu Dec 31 2020	02:10:00	09:00:00	11:10:00	1568	☐
1041	Spicejet	Delhi	Goa	Thu Dec 31 2020	02:00:00	04:00:00	06:00:00	1956	☐
1060	Go Air	London	Delhi	Thu Dec 31 2020	02:10:00	09:00:00	11:10:00	1568	☐
1061	Spicejet	Delhi	Bali	Thu Dec 31 2020	02:00:00	04:00:00	06:00:00	1956	☐
1062	Indigo	Delhi	Bali	Thu Dec 31 2020	02:30:00	05:00:00	07:30:00	2456	☐

Figure 5.11: Admin Flights Page.

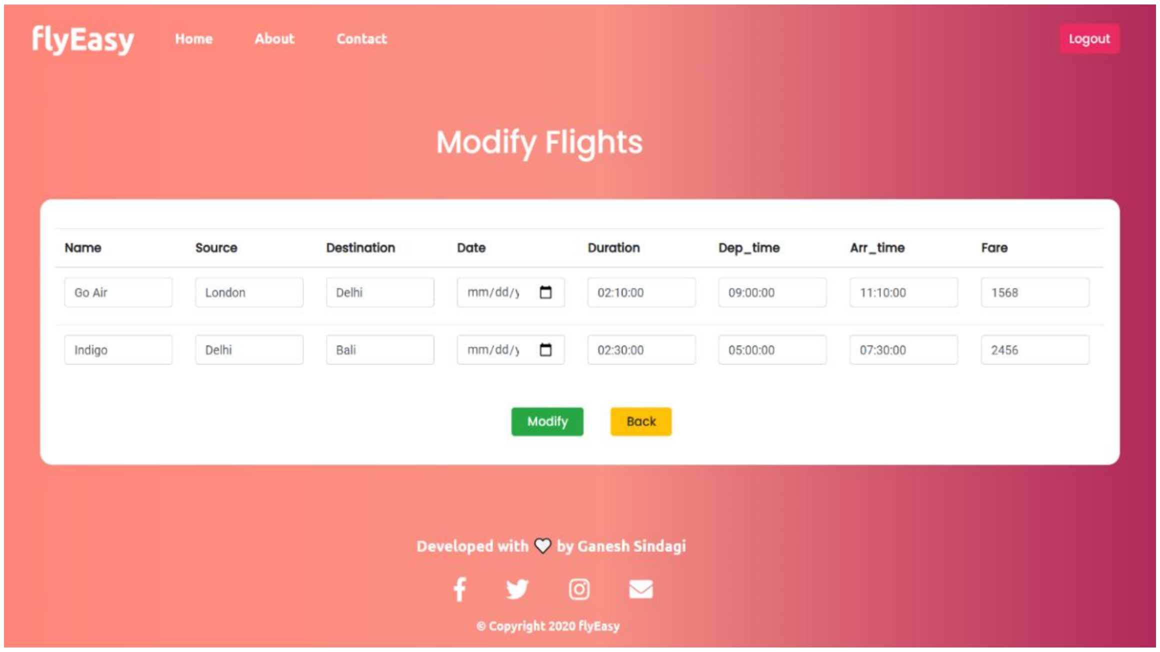
Admin Add Flights Page



Name	Source	Destination	Date	Duration	Dep_time	Arr_time	Fare
Name	Source	Destination	mm/dd/y	Duration	dep_time	arr_time	Fare

Figure 5.12: Admin Add Flights Page.

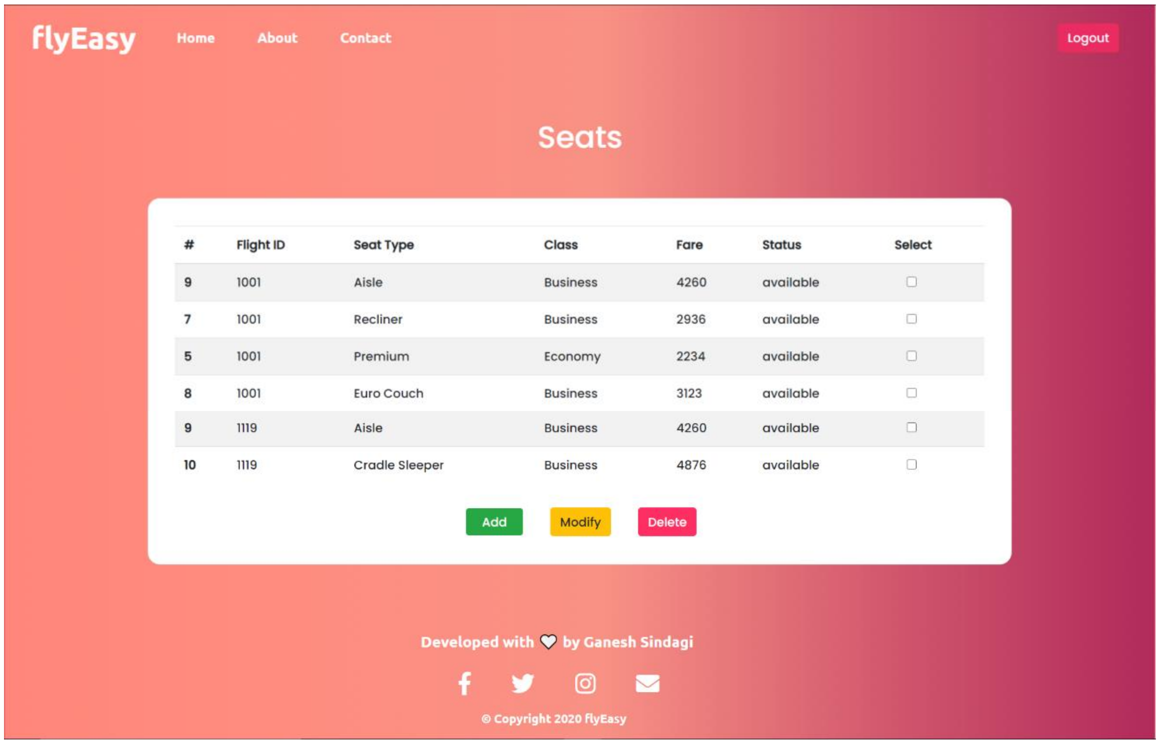
Admin Modify Flights Page



Name	Source	Destination	Date	Duration	Dep_time	Arr_time	Fare
Go Air	London	Delhi	mm/dd/y	02:10:00	09:00:00	11:10:00	1568
Indigo	Delhi	Bali	mm/dd/y	02:30:00	05:00:00	07:30:00	2456

Figure 5.13: Admin Modify Flights Page.

Admin Seats Page

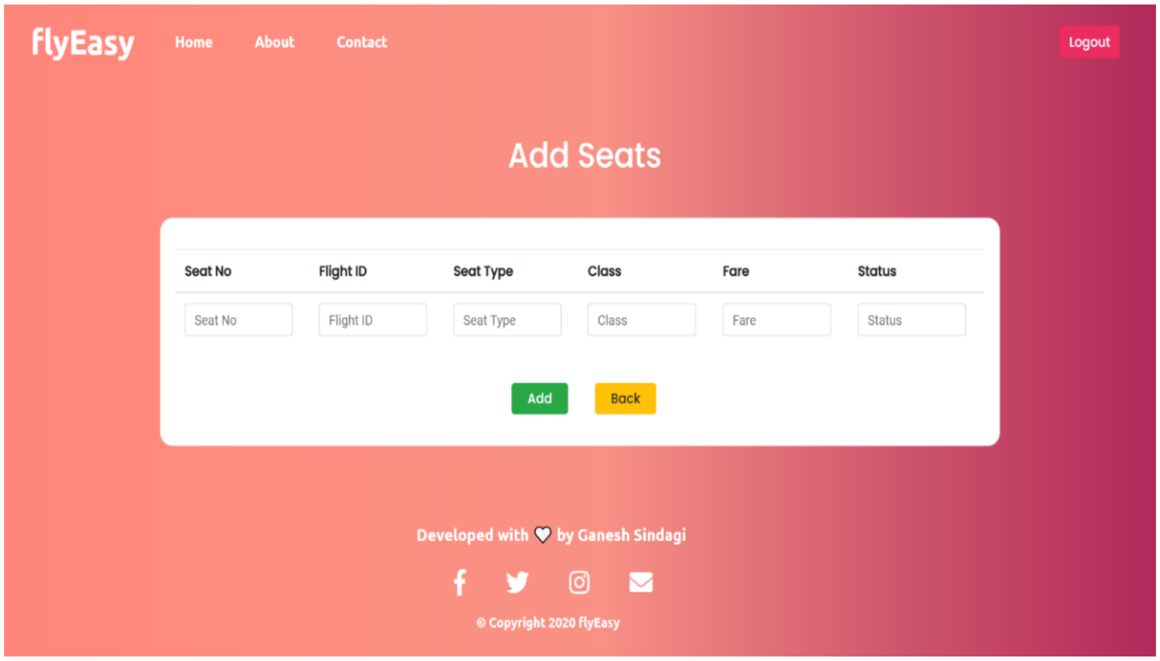


Seats

#	Flight ID	Seat Type	Class	Fare	Status	Select
9	1001	Aisle	Business	4260	available	☐
7	1001	Recliner	Business	2936	available	☐
5	1001	Premium	Economy	2234	available	☐
8	1001	Euro Couch	Business	3123	available	☐
9	1119	Aisle	Business	4260	available	☐
10	1119	Cradle Sleeper	Business	4876	available	☐

Figure 5.14: Admin Seats Page.

Admin Add Seats Page

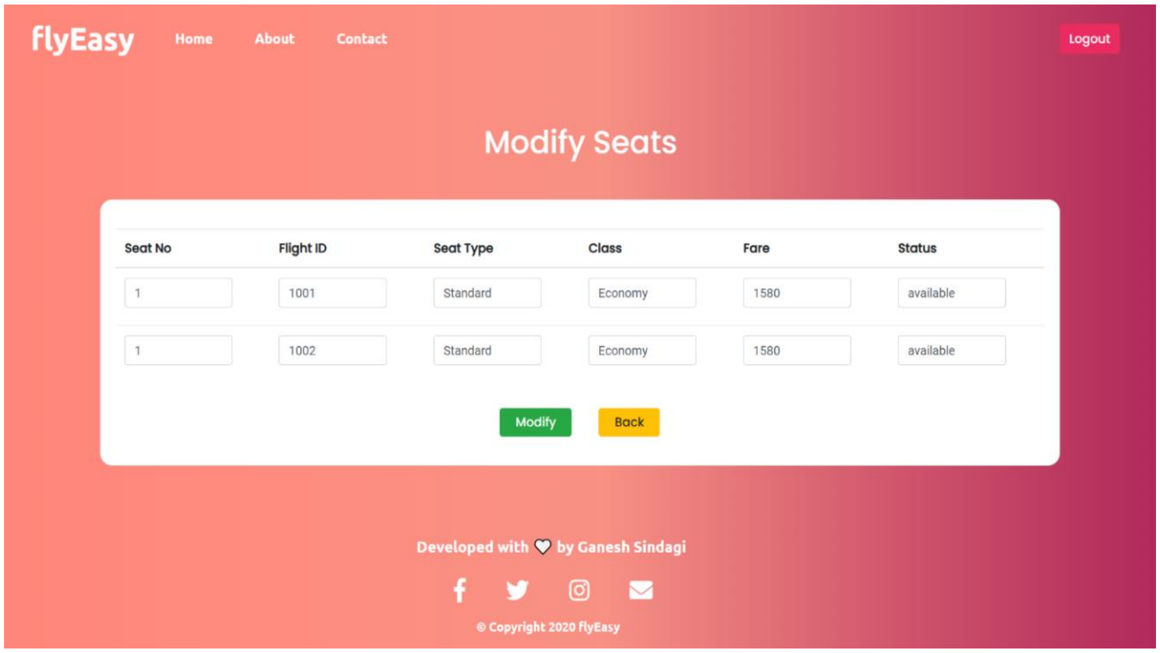


Add Seats

Seat No	Flight ID	Seat Type	Class	Fare	Status
Seat No	Flight ID	Seat Type	Class	Fare	Status

Figure 5.15: Admin Add Seats Page.

Admin Modify Seats Page

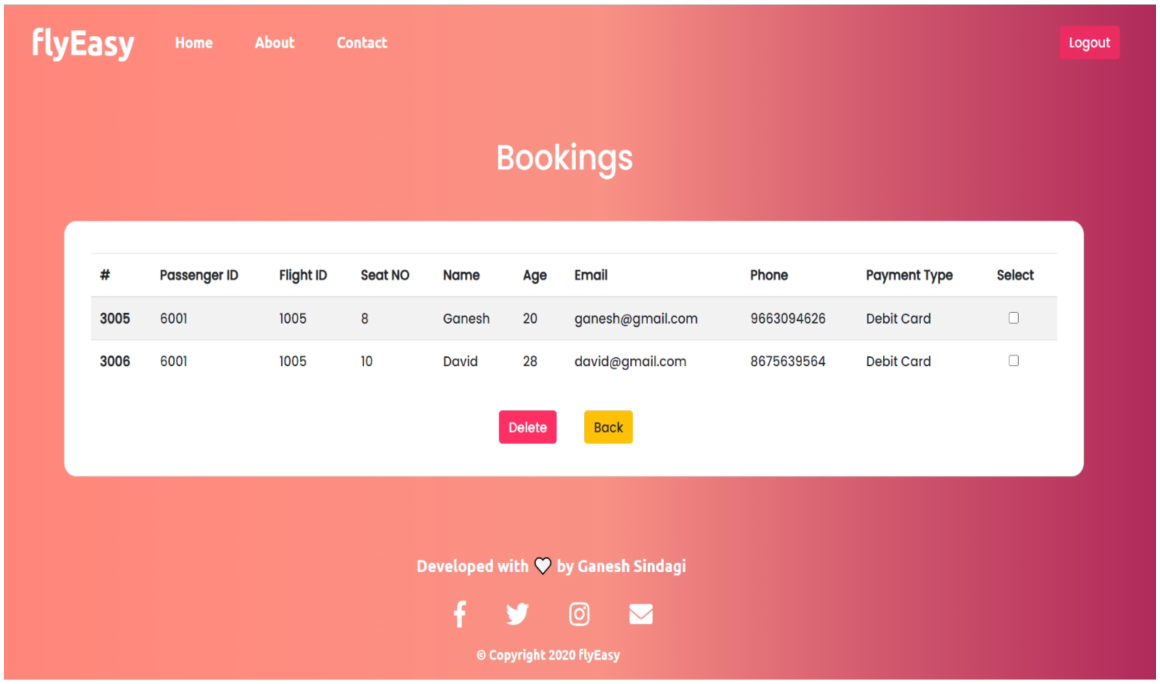


Modify Seats

Seat No	Flight ID	Seat Type	Class	Fare	Status
1	1001	Standard	Economy	1580	available
1	1002	Standard	Economy	1580	available

Figure 5.16: Admin Modify Seats Page.

Admin Bookings Page



Bookings

#	Passenger ID	Flight ID	Seat NO	Name	Age	Email	Phone	Payment Type	Select
3005	6001	1005	8	Ganesh	20	ganesh@gmail.com	9663094626	Debit Card	☐
3006	6001	1005	10	David	28	david@gmail.com	8675639564	Debit Card	☐

Figure 5.17: Admin Bookings Page.

CONCLUSION

The Airline Reservation Database System provides an easy access to the various flights in each airport in different cities and even makes ticket booking easier. The admin can access information related to the flights in different cities and the different flights owned by various airlines. The user can view the various flights and seats available to book tickets, and can book, cancel and also view the ticket information. Thus reduces the time and effort required by the passengers in booking by offline method. In the application the admin and users are provided with different username and passwords to prevent accessing of each other information.

The application takes care of all the requirements and is capable to provide easy and effective storage of information related to admins and users. Only the admin is given the authority to insert, delete and retrieve sensitive information like the flights information, the seats that a flight has and also the bookings made by the passengers. The application is built using suitable back-end and front-end which has made the project more efficient and user-friendly. Hence the application can be chosen over manual flight booking system.

FUTURE ENHANCEMENT

The project is easy to implement and also enhances the application as per future requirements. The table normalized in appropriate manner, so there will not be any ambiguity as the data increases. Hence, the application can be enhanced to meet the growing demands of the market. The future enhancements may include:

- The application can be enhanced by keeping track of user's previous order and payment histories.
- Introducing filters for searches so that user can search flight by price and timings.
- Allowing rescheduling of reservations.
- Increasing the security of the website from bcrypt encryption to google OAuth.
- Introducing cookies and sessions for the better user experience.