CMSC 661 Database Systems Concept

GoAir

Final Report

Dr. Yelena Yesha

Ajinkya Wakhale
AN 75014 wakhale1@umbc.edu

Deepanjan Bhattacharya
AD 68898 deep8@umbc.edu

Sushant Chaudhari
ZW 77612 sushant1@umbc.edu

Table of Contents

Section 1 Introduction

GoAir is a web based airline reservation system through which users can search the available flights, book a new flight and cancel the existing booking. **GoAir** gives you multiple options to plan your Journey.One has the option to book one-way ticket and if you are planning for a return journey, **GoAir** allows you to book return ticket in one go. For one booking,you have option to select class i.e Business, Economy and Premium in which you want to travel. Also, you can book tickets for up to five individuals in one booking.

Once registration is successful, you can plan your journey. There will be multiple flight options for you to select according to your source, destination, type of journey, number of passengers and seat class. After selecting the best option which suits you, you can select the seats on the flight according to the class in which you want to travel and number of passengers travelling. After successful booking, you will get a mail confirming your booking and your ticket.

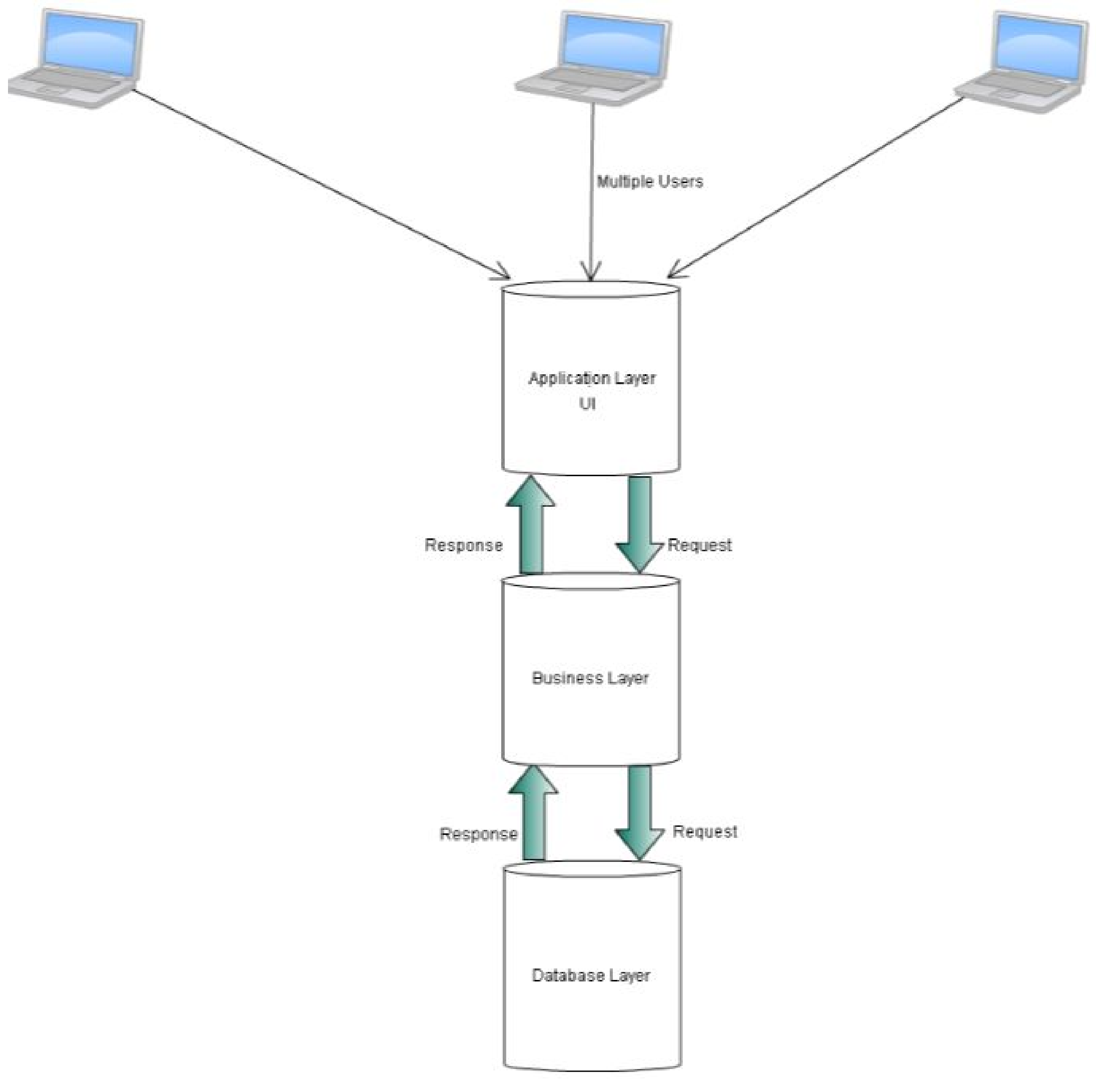
You also have option to cancel your whole booking or individual ticket. Along with that, you can always edit your personal details like updating the email id or changing the phone number before the journey to get real time notifications.

To sum it up, GoAir is an interactive and dynamic web based application which gives you option to book your tickets in a simple and efficient way.

Under the hood Lies a state of the art dedicated 3 tier Architecture. All independent of each other. The UI layer can be swapped and manipulated to custom taste without changing the business layer or database layer. The business layer exposes all functionality through rest api, so easy to consume whether on a desktop based application, web based or even mobile based. And if further custom middle layer is need these services can be consumed by another middle layer application to create a wrapper middle layer. Similarly the database structure and queries are also made with minute details such that they are capable across different types, changing a value in middle layer will make sure that you can connect to any relational database.
Though this system functions as a single unit, but each of the modules are are independently capable of working.

Section 2- System Requirements

2.1 System Architecture Diagram



2.2 Interface Requirements

1. Being a web based architecture so at the interface level, a computer capable of running browser is all the requirement.
2. The application runs on all types of browser, but for best user experience we suggest you to use chrome.
3. As of now we have only tested it on desktop and not on small screen devices.

2.3 Functional Requirements

Registration and Personal details:

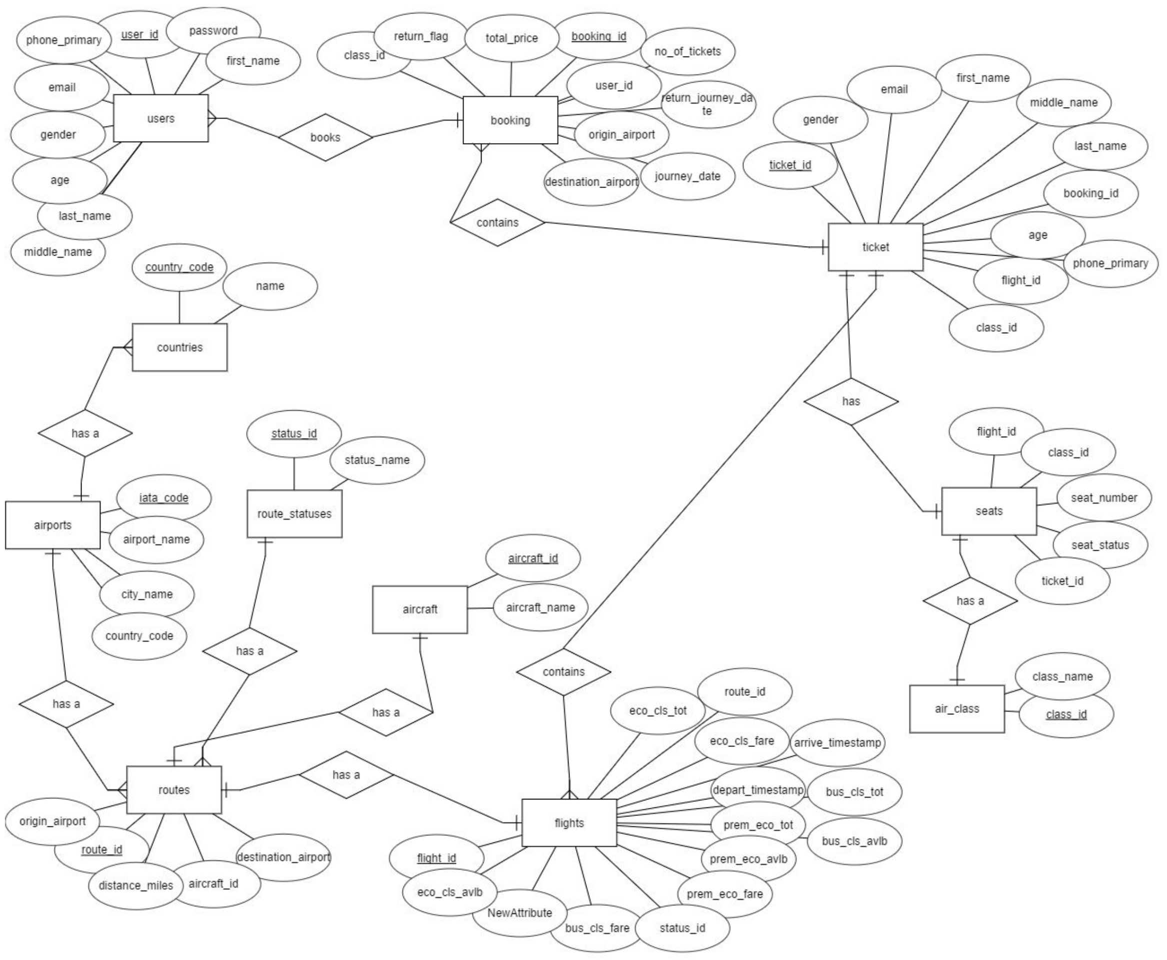
1. User should be able to sign up seamlessly. After successful registration, user should get a mail with his login credentials and there should be an option to change user's personal details if he wants to do so.

Booking

1. User should be able to search for the flights according to his source and destination. Also, there should be options to select seat class in which he wants to travel and number of passengers for whom he wants to book tickets.
2. Return journey option should be made available if user plans for a round trip.
3. For flight select options page, page should display all the available flights between user's source and destination and according to his/her departure date. It should display flight's departure time, arrival time,departure date,arrival date, flight id and importantly Price. Apart from all the direct flights between user' source and destination,page should also display all the available routes with the connecting flight options and connects user's source to the destination. All details for the connecting flight should also be displayed along with the layover time at the layover airport.
4. After selecting flights according to his/her preference, user should be able to select seats according to his/her choice in all the flights he plans to travel even for the round trip.
5. Once the seats are selected, a page should input all the passengers details according to the number of passengers who wants to travel.
6. After user has submitted their details, a confirmation page should come which displays all the flights which user has selected along with the seat and passenger details. Once user confirms these, ticket should be booked and a mail should be sent to the user who is booking tickets and all the passengers who are travelling.
7. There should be an option to view all the bookings which a registered user has done along with the option to cancel it.
8. Also a booking could have multiple tickets depending upon the number of passengers travelling. There should be an option to cancel individual ticket if the user intends to do so.

Section 3- Conceptual Design of the Database

3.1 Entity-Relationship (ER) Model



3.2 Data Dictionary and Business Rules

Data dictionary:

There are eleven tables in the database of this project. Each table is connected to other tables by foreign keys. Following are the tables and their use in the application.

1. Users: User_id is the primary key in this table and it is used to store the login credentials and other information related to the system user.
2. Booking: Booking table is used to store the booking details for the bookings made by the user. Each user_id may have multiple bookings and it is the foreign key here. Booking_id is the primary key of this table.
3. Ticket: This table is used to store the ticket details corresponding to the booking. Ticket_id is the primary key.
4. Seats: The seats table has seat details in it for each flight. For this project each flight has total 60 seats in it. There are three classes amongst which the seats are divided equally.

5. Air_class: This table has three classes stored in it viz. Business , Economy, and Premium Economy. Class_id is the primary key in this table.
6. Flights: Flight_id is the primary key in this table. It stores information regarding the flights such as total number of available seats, total seats booked.
7. Routes: Route table stores the routes for each flight. Routes are uniquely defined and has Route_id as its primary key.
8. Aircraft: Each route has a aircraft which flies on that route. Aircraft name is stored in this table. Aircraft_id is the primary key here.
9. Route_statuses: The status for each route in stored in this table. It has details like if a route is currently cancelled or is delayed. Status_id is the primary key in this table.
10. Airports: All the airports all over the world are stored in this table. Route has the iata_code which is the primary key for airports table as the source and destination locations.
11. Countries: Each airport is in one of the countries given in the countries table. Country_code is the primary key for this table.

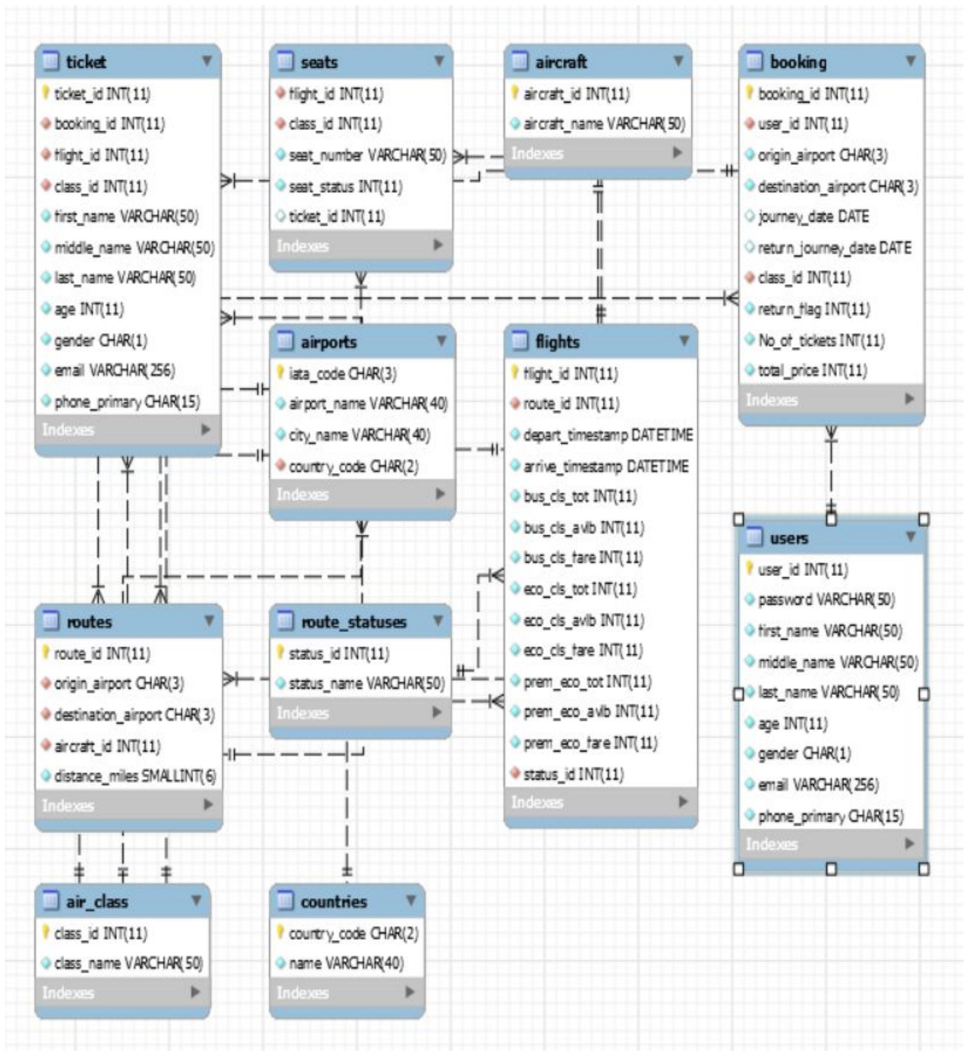
Business rules:

Following are the business rules that are the backbone of this application:

1. Each user can book maximum five tickets at a time for a flight using the user_id.
2. The user can cancel the ticket at any point once the ticket is booked before the departure.
3. User has the privilege to edit the details stored corresponding to the booking_id.
4. The ticket cannot be booked until the user has registered on the site.
5. User needs to provide all the basic details in order to do the booking.
6. Each class can have maximum of 20 tickets that can be booked.
7. The details entered during registration has to fulfill the constraints like the primary_contact should only be digits and no string allowed. Email should be a valid email id. The password and confirm password should match.
8. Each route has a unique flight that runs only on that route and not on any other route.
9. Only ‘M’ and ‘F’ are considered to be valid entries for gender entry in the database.
10. The distance_miles field should have only positive values.
11. Fare is calculated by multiplying the distance of each route by the class multiplier.

Section 4 Logical Database Schema

4.1 Schema of the Database



4.2 SQL Statements Used to Construct the Schema

DDL for creating the database:

```
CREATE TABLE IF NOT EXISTS users (
        user_id  INTEGER                          NOT NULL AUTO_INCREMENT,
        password       VARCHAR(50)          NOT NULL,
        first_name                       VARCHAR(50)          NOT NULL,
        middle_name               VARCHAR(50)          NOT NULL,
        last_name                        VARCHAR(50)          NOT NULL,
        age  INTEGER                              NOT NULL,
        gender              CHAR(1)               NOT NULL,
        email               VARCHAR(256)          NOT NULL,
        phone_primary               CHAR(15)      NOT NULL,
        CONSTRAINT valid_gender   CHECK (gender = 'M' OR gender = 'F'),
        PRIMARY KEY (user_id)
);

CREATE TABLE IF NOT EXISTS countries (
         country_code         CHAR(2)               NOT NULL UNIQUE,
         name                         VARCHAR(40)           NOT NULL,
         PRIMARY KEY(country_code)
);

CREATE TABLE IF NOT EXISTS airports (
         iata_code            CHAR(3)               NOT NULL UNIQUE,
         airport_name         VARCHAR(40)           NOT NULL,
         city_name           VARCHAR(40)           NOT NULL,
         country_code         CHAR(2)               NOT NULL,
         PRIMARY KEY (iata_code),
         FOREIGN KEY(country_code) REFERENCES countries(country_code)
);


CREATE TABLE IF NOT EXISTS air_class (
        class_id        INTEGER                               NOT NULL,
        class_name  VARCHAR(50)   NOT NULL,
        PRIMARY KEY (class_id)
        );

CREATE TABLE IF NOT EXISTS aircraft (
        aircraft_id INTEGER             NOT NULL,
        aircraft_name VARCHAR(50) NOT NULL,
        PRIMARY KEY (aircraft_id)
        );

CREATE TABLE IF NOT EXISTS route_statuses (
        status_id       INTEGER       NOT NULL,
   status_name VARCHAR(50)          NOT NULL,
   primary key(status_id)
);

CREATE TABLE IF NOT EXISTS routes (
         route_id                               INTEGER               NOT NULL,
         origin_airport               CHAR(3)               NOT NULL,
         destination_airport          CHAR(3)               NOT NULL,
         aircraft_id                     INTEGER             NOT NULL,
         distance_miles                          SMALLINT            NOT NULL CHECK
(distance_miles > 0),
         CONSTRAINT diff_orig_dest_airport CHECK(origin_airport != destination_airport),
         PRIMARY KEY (route_id),
         FOREIGN KEY (origin_airport)                REFERENCES airports(iata_code),
         FOREIGN KEY (destination_airport)           REFERENCES airports(iata_code),
         FOREIGN KEY (aircraft_id)   REFERENCES aircraft(aircraft_id)
);

CREATE TABLE IF NOT EXISTS flights (
         flight_id                       INTEGER             NOT NULL,
         route_id                        INTEGER             NOT NULL,
         depart_timestamp        TIMESTAMP   NOT NULL,
         arrive_timestamp        TIMESTAMP   NOT NULL,
         bus_cls_tot             INTEGER                             NOT NULL,
         bus_cls_avlb            INTEGER                             NOT NULL,
         bus_cls_fare            INTEGER                             NOT NULL,
         eco_cls_tot             INTEGER                             NOT NULL,
         eco_cls_avlb            INTEGER                             NOT NULL,
         eco_cls_fare            INTEGER                             NOT NULL,
         prem_eco_tot            INTEGER                             NOT NULL,
         prem_eco_avlb                   INTEGER                             NOT NULL,
         prem_eco_fare                   INTEGER                             NOT NULL,
         status_id                       INTEGER                             NOT NULL,
         PRIMARY KEY (flight_id),
         FOREIGN KEY (route_id)              REFERENCES routes(route_id),
         FOREIGN KEY (status_id)             REFERENCES route_statuses(status_id)
);

CREATE TABLE IF NOT EXISTS booking (
        booking_id  INTEGER                 NOT NULL AUTO_INCREMENT,
        user_id          INTEGER                             NOT NULL,
        flight_id        INTEGER                             NOT NULL,
        PRIMARY KEY (booking_id),
        FOREIGN KEY (user_id)               REFERENCES users(user_id),
        FOREIGN KEY (flight_id)             REFERENCES flights(flight_id)
        );

CREATE TABLE IF NOT EXISTS ticket (
        ticket_id        INTEGER         NOT NULL AUTO_INCREMENT,
        booking_id       INTEGER                             NOT NULL,
        class_id         INTEGER                             NOT NULL,
        first_name                      VARCHAR(50)         NOT NULL,
        middle_name              VARCHAR(50)         NOT NULL,
        last_name                       VARCHAR(50)         NOT NULL,
        age      INTEGER                              NOT NULL,
        gender                  CHAR(1)               NOT NULL,
        email                   VARCHAR(256)          NOT NULL,
        phone_primary                   CHAR(15)      NOT NULL,
        CONSTRAINT valid_gender   CHECK (gender = 'M' OR gender = 'F'),
        PRIMARY KEY (ticket_id),
        FOREIGN KEY (booking_id)            REFERENCES booking(booking_id),
        FOREIGN KEY (class_id)              REFERENCES air_class(class_id)
);


CREATE TABLE IF NOT EXISTS seats (
        flight_id          INTEGER                NOT NULL,
        class_id           INTEGER                NOT NULL,
        seat_number VARCHAR(50)  NOT NULL,
        seat_status INTEGER                       NOT NULL,
        ticket_id   INTEGER,
        FOREIGN KEY (flight_id)       REFERENCES flights(flight_id),
        FOREIGN KEY (class_id)              REFERENCES air_class(class_id)
        );
```

Section 5

Functional Dependencies and Database Normalization

5.1 Functional Dependencies

Following functional dependencies exist in the database of our application:

User_id is the superkey for the users relation.
User_id → password, first_name, middle_name, last_name, age, gender, email, phone_primary

Flight_id is the superkey in the flights relation.
Flight_id → route_id, depart_timestamp, arrive_timestamp, bus_cls_tot, bus_cls_avlb bus_cls_fare, eco_cls_tot, eco_cls_avlb, eco_cls_fare, prem_eco_tot, prem_eco_avlb, prem_eco_fare, status_id

Route_id is the superkey for route relation which uniquely identifies the following attributes.

Route_id → origin_airport, destination_airport, distance_miles

Iata_code is the superkey in airports relation
Iata_code → airport_name, city_name

Ticket_id is the superkey in the tickets relation
Ticket_id → first_name, middle_name, last_name, age, gender, email, phone_primary

Country_code is the superkey for the countries relation
Country_code → name

Class_id is the superkey in the air_class relation
Class_id → class_name

Aircraft_id is the superkey for the aircraft relation
Aircraft_id → aircraft_name

Status_id is the superkey for the route_statuses
Status_id → status_name

5.2 SQL Statements for Constructing the Table

```
CREATE TABLE IF NOT EXISTS users (
        user_id  INTEGER                                NOT NULL AUTO_INCREMENT,
        password        VARCHAR(50)           NOT NULL,
        first_name                   VARCHAR(50)          NOT NULL,
        middle_name             VARCHAR(50)          NOT NULL,
        last_name                    VARCHAR(50)          NOT NULL,
        age  INTEGER                                    NOT NULL,
        gender                  CHAR(1)                 NOT NULL,
        email                   VARCHAR(256)            NOT NULL,
        phone_primary                CHAR(15)         NOT NULL,
        CONSTRAINT valid_gender   CHECK (gender = 'M' OR gender = 'F'),
        PRIMARY KEY (user_id)
);

CREATE TABLE IF NOT EXISTS countries (
         country_code           CHAR(2)                 NOT NULL UNIQUE,
         name                         VARCHAR(40)          NOT NULL,
         PRIMARY KEY(country_code)
);
```

```
CREATE TABLE IF NOT EXISTS airports (
        iata_code               CHAR(3)             NOT NULL UNIQUE,
        airport_name            VARCHAR(40)         NOT NULL,
        city_name               VARCHAR(40)         NOT NULL,
        country_code            CHAR(2)             NOT NULL,
        PRIMARY KEY (iata_code),
        FOREIGN KEY(country_code) REFERENCES countries(country_code)
);

CREATE TABLE IF NOT EXISTS air_class (
        class_id        INTEGER                             NOT NULL,
        class_name  VARCHAR(50)   NOT NULL,
        PRIMARY KEY (class_id)
        );

CREATE TABLE IF NOT EXISTS aircraft (
        aircraft_id INTEGER             NOT NULL,
        aircraft_name VARCHAR(50) NOT NULL,
        PRIMARY KEY (aircraft_id)
        );

CREATE TABLE IF NOT EXISTS route_statuses (
        status_id       INTEGER         NOT NULL,
   status_name VARCHAR(50)          NOT NULL,
   primary key(status_id)
);

CREATE TABLE IF NOT EXISTS routes (
        route_id                                INTEGER                 NOT NULL,
        origin_airport                  CHAR(3)                 NOT NULL,
        destination_airport             CHAR(3)                 NOT NULL,
        aircraft_id                     INTEGER                 NOT NULL,
        Distance_miles SMALLINT                 NOT NULL CHECK (distance_miles > 0),
        CONSTRAINT diff_orig_dest_airport CHECK(origin_airport != destination_airport),
        PRIMARY KEY (route_id),
        FOREIGN KEY (origin_airport)                    REFERENCES airports(iata_code),
        FOREIGN KEY (destination_airport)               REFERENCES airports(iata_code),
        FOREIGN KEY (aircraft_id)   REFERENCES aircraft(aircraft_id)
);

CREATE TABLE IF NOT EXISTS flights (
        flight_id                       INTEGER                 NOT NULL,
        route_id                        INTEGER                 NOT NULL,
        depart_timestamp        TIMESTAMP   NOT NULL,
        arrive_timestamp        TIMESTAMP   NOT NULL,
        bus_cls_tot             INTEGER                                 NOT NULL,
        bus_cls_avlb        INTEGER                             NOT NULL,
        bus_cls_fare        INTEGER                             NOT NULL,
        eco_cls_tot         INTEGER                             NOT NULL,
        eco_cls_avlb        INTEGER                             NOT NULL,
        eco_cls_fare        INTEGER                             NOT NULL,
        prem_eco_tot        INTEGER                             NOT NULL,
        prem_eco_avlb           INTEGER                             NOT NULL,
        prem_eco_fare           INTEGER                             NOT NULL,
        status_id               INTEGER                             NOT NULL,
        PRIMARY KEY (flight_id),
        FOREIGN KEY (route_id)          REFERENCES routes(route_id),
        FOREIGN KEY (status_id)         REFERENCES route_statuses(status_id)
);

CREATE TABLE IF NOT EXISTS booking (
        booking_id  INTEGER         NOT NULL AUTO_INCREMENT,
        user_id         INTEGER                         NOT NULL,
        flight_id       INTEGER                         NOT NULL,
        PRIMARY KEY (booking_id),
        FOREIGN KEY (user_id)           REFERENCES users(user_id),
        FOREIGN KEY (flight_id)         REFERENCES flights(flight_id)
        );

CREATE TABLE IF NOT EXISTS ticket (
        ticket_id       INTEGER         NOT NULL AUTO_INCREMENT,
        booking_id      INTEGER                         NOT NULL,
        class_id        INTEGER                         NOT NULL,
        first_name                      VARCHAR(50)     NOT NULL,
        middle_name             VARCHAR(50)     NOT NULL,
        last_name                       VARCHAR(50)     NOT NULL,
        age     INTEGER                         NOT NULL,
        gender                  CHAR(1)         NOT NULL,
        email                   VARCHAR(256)    NOT NULL,
        phone_primary                   CHAR(15)        NOT NULL,
        CONSTRAINT valid_gender   CHECK (gender = 'M' OR gender = 'F'),
        PRIMARY KEY (ticket_id),
        FOREIGN KEY (booking_id)        REFERENCES booking(booking_id),
        FOREIGN KEY (class_id)          REFERENCES air_class(class_id)
);


CREATE TABLE IF NOT EXISTS seats (
        flight_id       INTEGER             NOT NULL,
        class_id        INTEGER             NOT NULL,
        seat_number VARCHAR(50)  NOT NULL,
        seat_status INTEGER                 NOT NULL,
        ticket_id   INTEGER,
        FOREIGN KEY (flight_id)       REFERENCES flights(flight_id),
        FOREIGN KEY (class_id)                 REFERENCES air_class(class_id)
        );
```

Section 6- The Use of the Database System

6.1 System Installation Description

For Database, Mysql needs to installed
Two scripts are provided.

1. Script_Schema.sql. This contains all the database schemas
2. Script_Metadata.sql. This contains all relevant insert scripts

Create the Database schema and run the above scripts to setup the Database.

For the Middle layer, Node Js needs to be installed.

1. Navigate to the Middle layer folder via cmd prompt and and run “npm install package”
2. Open the “businesslogic.js” and fill in the appropriate connection for the database.

For the UI layer No extra setup is needed.

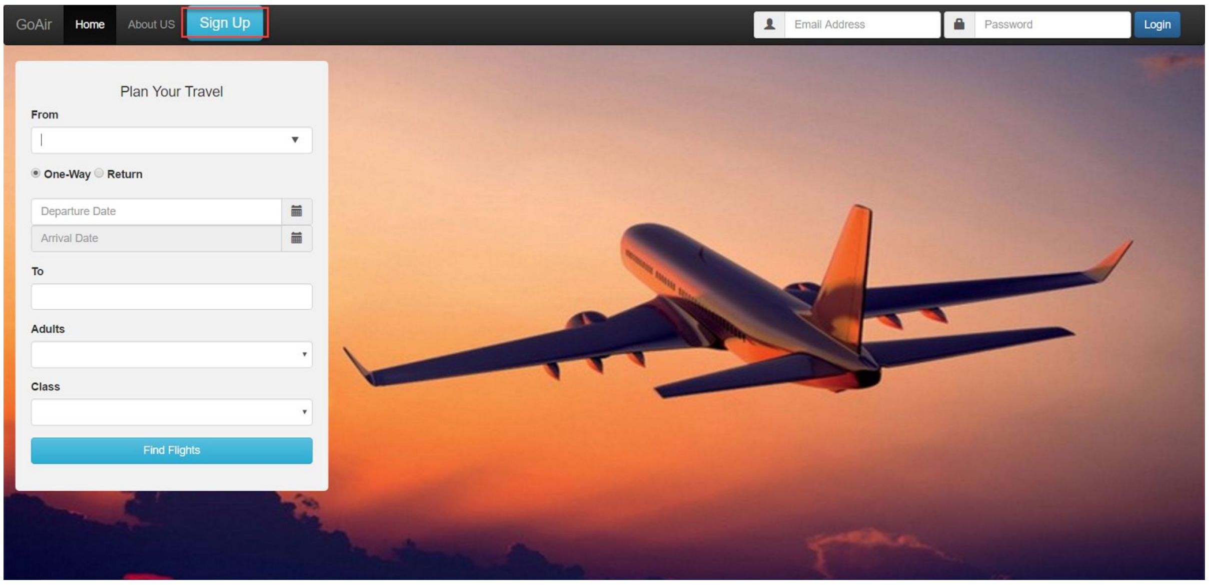
1. Index.html starts up the application.

6.2 The Use of the System

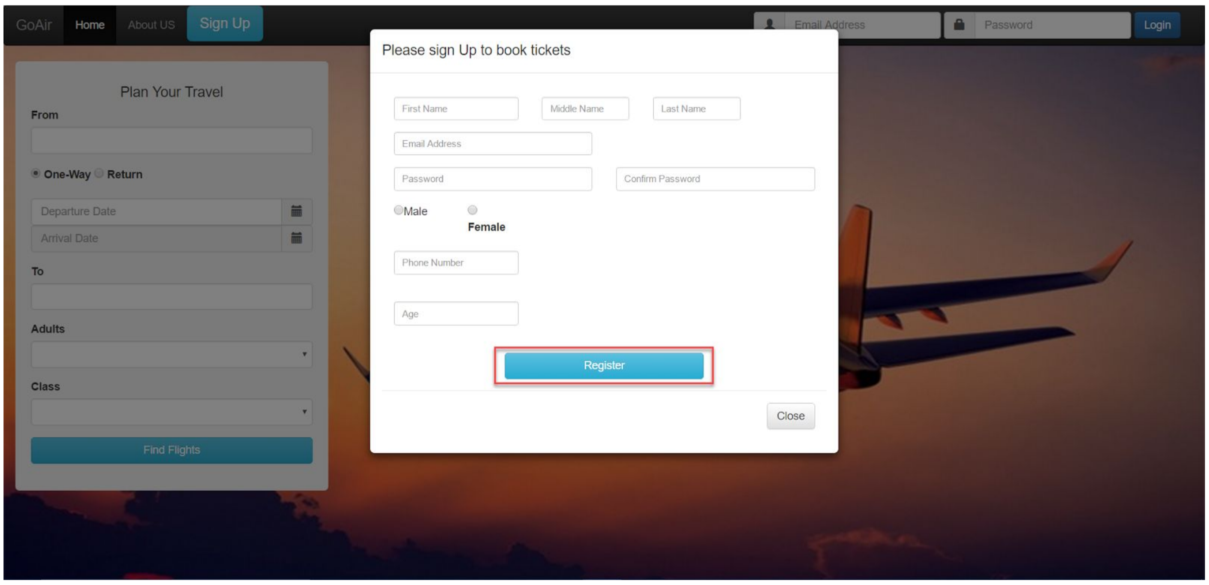
- The system can be used to search for flights based on the journey date
- Select one way or return journey
- Select seat of choice
- Book a maximum of 5 tickets per user
- Edit passenger details
- Overall the system can be used a real time airline reservation system as it has all the features of most of the applications that are being used today.

Below are few screenshots of the system user interface for reference:

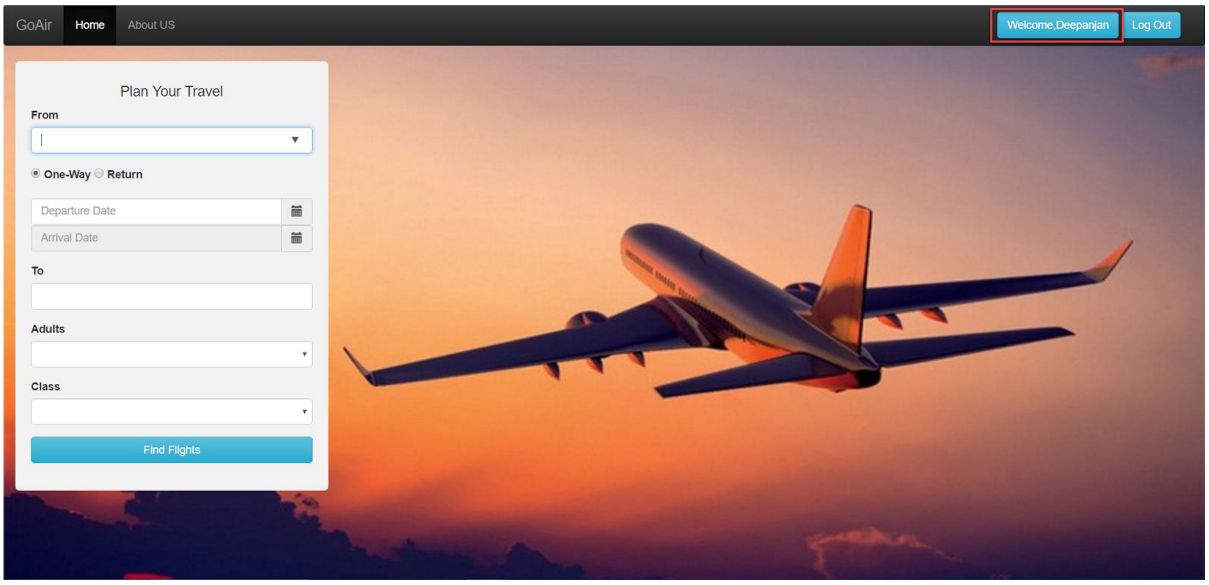
Home page:



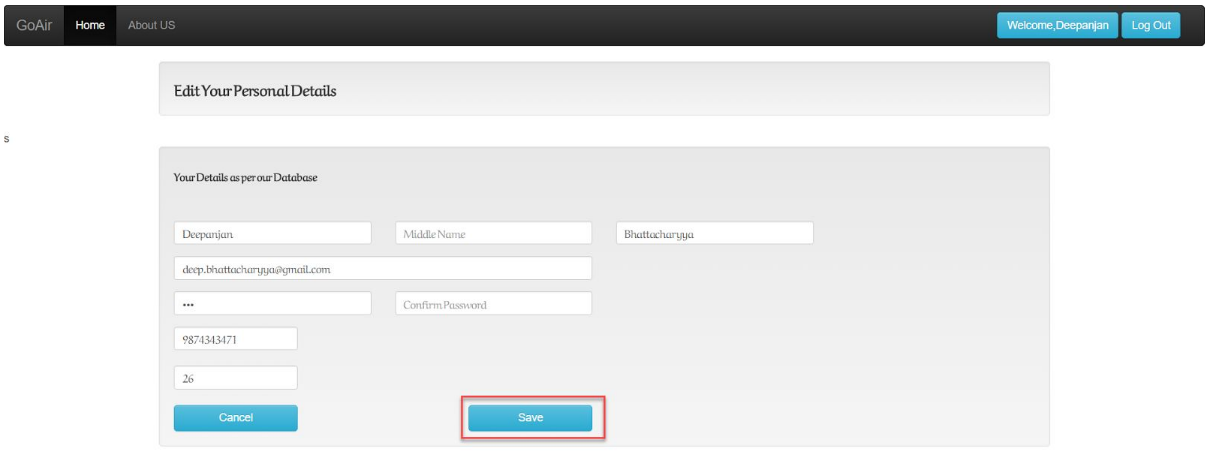
Registration page:



Search page:



Personal details Page:



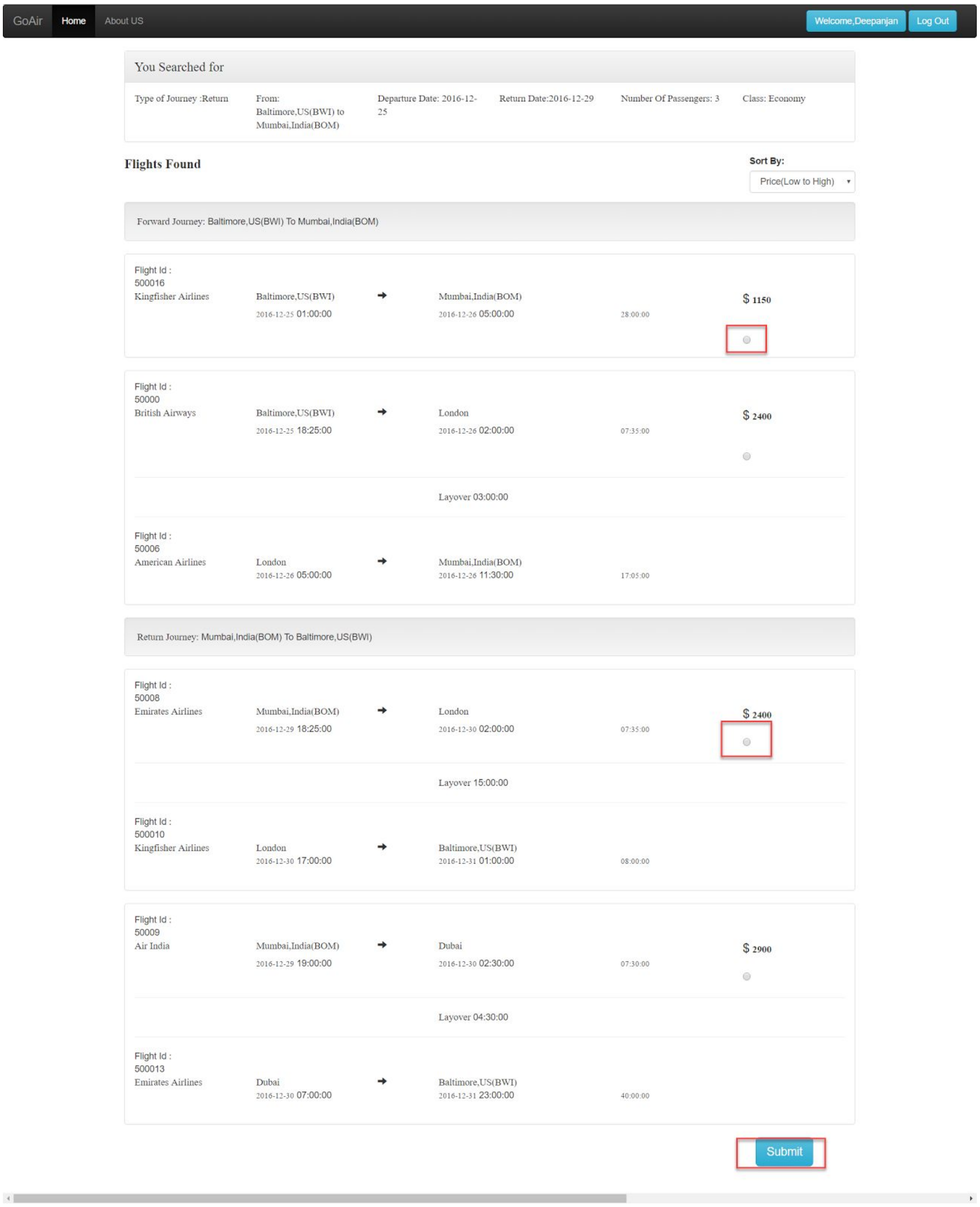
Edit Your Personal Details

s

Your Details as per our Database

Deepanjan
Middle Name
Bhattacharyya
deep.bhattacharyya@gmail.com
•••
Confirm Password
9874343471
26

Result Page:



You Searched for

Type of Journey :Return | From: Baltimore,US(BWI) to Mumbai,India(BOM) | Departure Date: 2016-12-25 | Return Date:2016-12-29 | Number Of Passengers: 3 | Class: Economy

Flights Found

Sort By: Price(Low to High)

Forward Journey: Baltimore,US(BWI) To Mumbai,India(BOM)

Flight Id : 500016
Kingfisher Airlines | Baltimore,US(BWI) 2016-12-25 01:00:00 → Mumbai,India(BOM) 2016-12-26 05:00:00 | 28:00:00 | $ 1150

Flight Id : 50000
British Airways | Baltimore,US(BWI) 2016-12-25 18:25:00 → London 2016-12-26 02:00:00 | 07:35:00 | $ 2400

Layover 03:00:00

Flight Id : 50006
American Airlines | London 2016-12-26 05:00:00 → Mumbai,India(BOM) 2016-12-26 11:30:00 | 17:05:00

Return Journey: Mumbai,India(BOM) To Baltimore,US(BWI)

Flight Id : 50008
Emirates Airlines | Mumbai,India(BOM) 2016-12-29 18:25:00 → London 2016-12-30 02:00:00 | 07:35:00 | $ 2400

Layover 15:00:00

Flight Id : 500010
Kingfisher Airlines | London 2016-12-30 17:00:00 → Baltimore,US(BWI) 2016-12-31 01:00:00 | 08:00:00

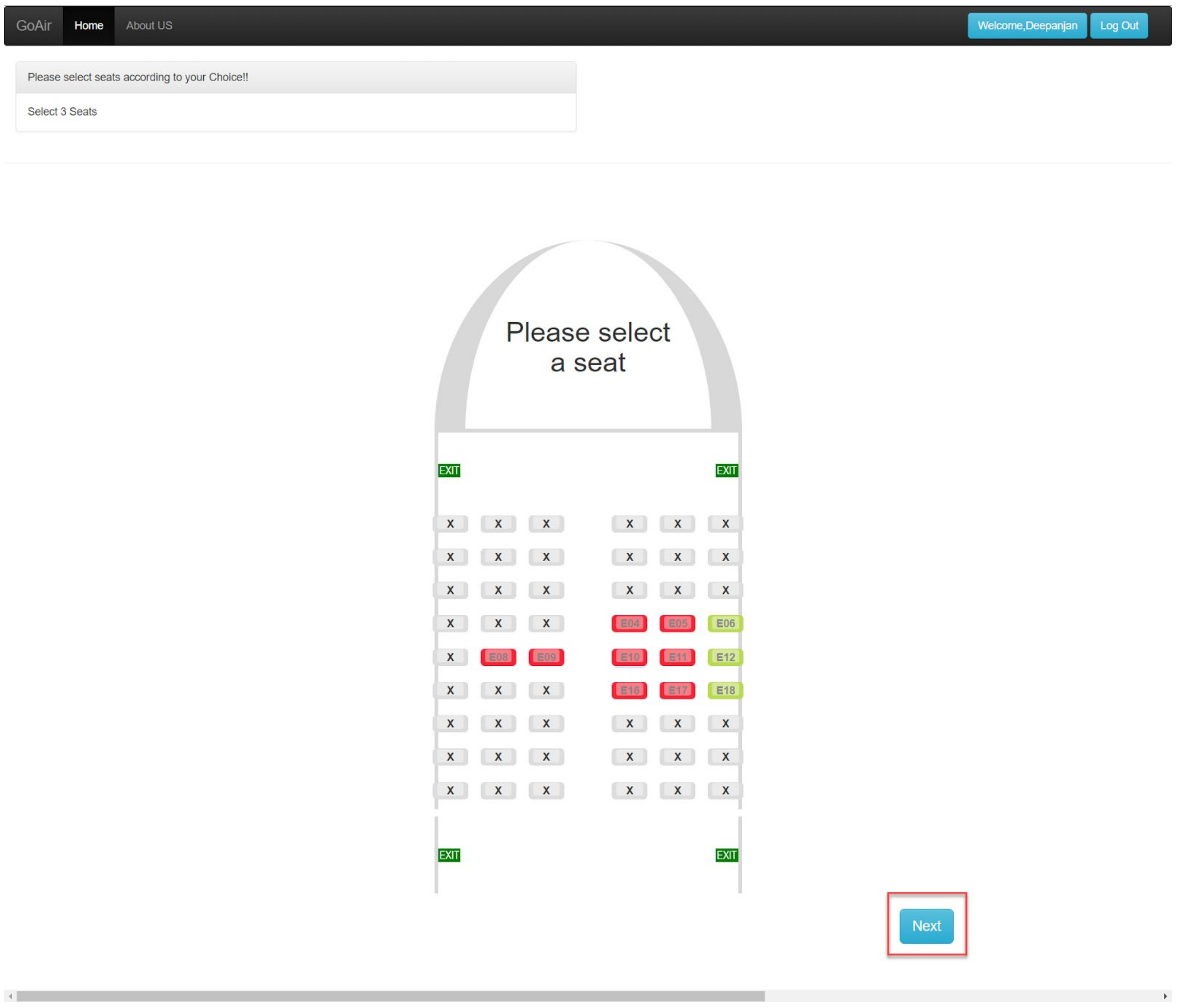
Flight Id : 50009
Air India | Mumbai,India(BOM) 2016-12-29 19:00:00 → Dubai 2016-12-30 02:30:00 | 07:30:00 | $ 2900

Layover 04:30:00

Flight Id : 500013
Emirates Airlines | Dubai 2016-12-30 07:00:00 → Baltimore,US(BWI) 2016-12-31 23:00:00 | 40:00:00

Submit

Seat selection Page:



Please select seats according to your Choice!!

Select 3 Seats

Please select a seat

Section 7 Suggestions of Database Tuning

1. Using procedures for some portion of business logic, where in dynamic query generation and multiple query could be avoided while taking advantage of transaction features.
2. Using functions for repetitive works, like conversion of date format.
3. Feature to support distributed Databases.

Section 8 Additional Queries and Views

#Search query for a non direct flight:

# For business class:

```
SELECT
    t1.origin_airport,
    t2.destination_airport,
    t1.depart_timestamp,
    t2.arrive_timestamp,
    t1.destination_airport AS stop_airport,
    t1.arrive_timestamp AS stop_airport_arrival,
    t2.depart_timestamp AS stop_airport_departure,
    t1.aircraft_name AS source_to_stop,
    t1.flight_id AS flight_id1,
    t2.aircraft_name AS stop_to_destination,
    t2.flight_id AS flight_id2,
    SEC_TO_TIME(TIMESTAMPDIFF(SECOND,
                t1.arrive_timestamp,
                t2.depart_timestamp)) AS layover,
    (t1.bus_cls_fare * t1.distance_miles) + (t2.bus_cls_fare * t2.distance_miles) AS Total_fare,
    '' AS flight_id,
    '' AS aircraft_name
FROM
    (SELECT
        r.origin_airport,
            r.destination_airport,

            f.depart_timestamp,

            f.arrive_timestamp,

            a.aircraft_name,

            f.flight_id,

            f.bus_cls_fare,

            r.distance_miles

    FROM

        routes r

    JOIN flights f

    JOIN aircraft a ON r.route_id = f.route_id

        AND r.aircraft_id = a.aircraft_id

    WHERE

        origin_airport = 'JFK'

            AND DATE(f.depart_timestamp) = '2016-12-25') t1,

    (SELECT

        r.destination_airport,

            r.origin_airport,

            f.arrive_timestamp,

            f.depart_timestamp,

            a.aircraft_name,

            f.flight_id,
            f.bus_cls_fare,

            r.distance_miles

        FROM

            routes r

        JOIN flights f

        JOIN aircraft a ON r.route_id = f.route_id

            AND r.aircraft_id = a.aircraft_id

            AND destination_airport = 'BOM') t2

    WHERE

        t2.origin_airport = t1.destination_airport

            AND t2.depart_timestamp > t1.arrive_timestamp

            AND (DATE(t2.depart_timestamp) = '2016-12-25'

            OR DATE(t2.depart_timestamp) = DATE_ADD('2016-12-25', INTERVAL 1 DAY));
```

# For Economy Class:

```
SELECT

    t1.origin_airport,

    t2.destination_airport,

    t1.depart_timestamp,

    t2.arrive_timestamp,

    t1.destination_airport AS stop_airport,
    t1.arrive_timestamp AS stop_airport_arrival,
    t2.depart_timestamp AS stop_airport_departure,
    t1.aircraft_name AS source_to_stop,
    t1.flight_id AS flight_id1,
    t2.aircraft_name AS stop_to_destination,
    t2.flight_id AS flight_id2,
    SEC_TO_TIME(TIMESTAMPDIFF(SECOND,
                t1.arrive_timestamp,
                t2.depart_timestamp)) AS layover,
    (t1.eco_cls_fare * t1.distance_miles) + (t2.eco_cls_fare * t2.distance_miles) AS Total_fare,
    '' AS flight_id,
    '' AS aircraft_name
FROM
    (SELECT
        r.origin_airport,
            r.destination_airport,
            f.depart_timestamp,
            f.arrive_timestamp,
            a.aircraft_name,
            f.flight_id,
            f.eco_cls_fare,
            r.distance_miles
```

```
FROM
    routes r
JOIN flights f
JOIN aircraft a ON r.route_id = f.route_id
    AND r.aircraft_id = a.aircraft_id
WHERE
    origin_airport = 'JFK'
        AND DATE(f.depart_timestamp) = '2016-12-25') t1,
(SELECT
    r.destination_airport,
        r.origin_airport,
        f.arrive_timestamp,
        f.depart_timestamp,
        a.aircraft_name,
        f.flight_id,
        f.eco_cls_fare,
        r.distance_miles
FROM
    routes r
JOIN flights f
JOIN aircraft a ON r.route_id = f.route_id
    AND r.aircraft_id = a.aircraft_id
```

```
        AND destination_airport = 'BOM') t2
WHERE
    t2.origin_airport = t1.destination_airport
        AND t2.depart_timestamp > t1.arrive_timestamp
        AND (DATE(t2.depart_timestamp) = '2016-12-25'
        OR DATE(t2.depart_timestamp) = DATE_ADD('2016-12-25', INTERVAL 1 DAY));
```

#For Premium economy class:

```
SELECT
    t1.origin_airport,
    t2.destination_airport,
    t1.depart_timestamp,
    t2.arrive_timestamp,
    t1.destination_airport AS stop_airport,
    t1.arrive_timestamp AS stop_airport_arrival,
    t2.depart_timestamp AS stop_airport_departure,
    t1.aircraft_name AS source_to_stop,
    t1.flight_id AS flight_id1,
    t2.aircraft_name AS stop_to_destination,
    t2.flight_id AS flight_id2,
    SEC_TO_TIME(TIMESTAMPDIFF(SECOND,
                t1.arrive_timestamp,
            t2.depart_timestamp)) AS layover,
    (t1.prem_eco_fare * t1.distance_miles) + (t2.prem_eco_fare * t2.distance_miles) AS Total_fare,
    '' AS flight_id,
    '' AS aircraft_name
FROM
    (SELECT
        r.origin_airport,
            r.destination_airport,
            f.depart_timestamp,
            f.arrive_timestamp,
            a.aircraft_name,
            f.flight_id,
            f.prem_eco_fare,
            r.distance_miles
    FROM
        routes r
    JOIN flights f
    JOIN aircraft a ON r.route_id = f.route_id
        AND r.aircraft_id = a.aircraft_id
    WHERE
        origin_airport = 'JFK'
            AND DATE(f.depart_timestamp) = '2016-12-25') t1,
    (SELECT
        r.destination_airport,
            r.origin_airport,
            f.arrive_timestamp,
            f.depart_timestamp,
            a.aircraft_name,
            f.flight_id,
            f.prem_eco_fare,
            r.distance_miles
    FROM
        routes r
    JOIN flights f
    JOIN aircraft a ON r.route_id = f.route_id
        AND r.aircraft_id = a.aircraft_id
        AND destination_airport = 'BOM') t2
WHERE
    t2.origin_airport = t1.destination_airport
        AND t2.depart_timestamp > t1.arrive_timestamp
        AND (DATE(t2.depart_timestamp) = '2016-12-25'
        OR DATE(t2.depart_timestamp) = DATE_ADD('2016-12-25', INTERVAL 1 DAY));
```

#Search query For direct flights:

#For Business class:

```
SELECT

    origin_airport,

    destination_airport,

    depart_timestamp,

    arrive_timestamp,

    '' AS stop_airport,

    '' AS stop_airport_arrival,

    '' AS stop_airport_departure,

    '' as source_to_stop,

    '' as flight_id1,

    '' as stop_to_destination,

    '' as flight_id2,

    '' as layover,

    bus_cls_fare * distance_miles AS Total_fare,

    flight_id,

    aircraft_name

FROM

    routes r

        JOIN

    flights f

        JOIN
    aircraft a ON r.route_id = f.route_id
        AND r.aircraft_id = a.aircraft_id
WHERE
    origin_airport = 'JFK'
        AND destination_airport = 'LHR'
        AND DATE(depart_timestamp) = '2016-12-25';
```

#For Economy class:

```
SELECT
    origin_airport,
    destination_airport,
    depart_timestamp,
    arrive_timestamp,
    '' AS stop_airport,
    '' AS stop_airport_arrival,
    '' AS stop_airport_departure,
    '' as source_to_stop,
    '' as flight_id1,
    '' as stop_to_destination,
    '' as flight_id2,
    '' as layover,

    eco_cls_fare * distance_miles AS Total_fare,

    flight_id,

    aircraft_name

FROM

    routes r

        JOIN

    flights f

        JOIN

    aircraft a ON r.route_id = f.route_id

        AND r.aircraft_id = a.aircraft_id

WHERE

    origin_airport = 'JFK'

        AND destination_airport = 'LHR'

        AND DATE(depart_timestamp) = '2016-12-25';
```

#For premium economy class

```
SELECT

    origin_airport,

    destination_airport,

    depart_timestamp,
    arrive_timestamp,
    '' AS stop_airport,
    '' AS stop_airport_arrival,
    '' AS stop_airport_departure,
    '' as source_to_stop,
    '' as flight_id1,
    '' as stop_to_destination,
    '' as flight_id2,
    '' as layover,
    prem_eco_fare * distance_miles AS Total_fare,
    flight_id,
    aircraft_name
FROM
    routes r
        JOIN
    flights f
        JOIN
    aircraft a ON r.route_id = f.route_id
        AND r.aircraft_id = a.aircraft_id
WHERE
    origin_airport = 'JFK'
        AND destination_airport = 'LHR'
        AND DATE(depart_timestamp) = '2016-12-25';
```

# User creation:

```
insert into users values
(user_id,'password','first_name','last_name','date_of_birth','gender','email',phone_primary);
```

#update seat status after booking and add ticket_id

```
start transaction;
```

#lock rows for transaction

```
select * from seats where flight_id = $flight_id and seat_number = $seat_number for update;
```

#updating booking table:

```
insert into booking values (booking_id,user_id,flight_id);
```

#Booking a ticket:

```
insert into ticket values
(ticket_id,booking_id,class_id,first_name,middle_name,last_name,age,gender,email,phone_primary);
```

#update seat table

```
update seats

set seat_status = 2, ticket_id = $ticket_id

where flight_id = $flight_id and seat_number = $seat_number;
```

#update available seats after each booking:

#for business class

update flights

set bus_cls_avlb = bus_cls_avlb - 1

where flight_id = $flight_id;

#for economy class

update flights

set eco_cls_avlb = eco_cls_avlb - 1

where flight_id = $flight_id;

#for premium economy class

update flights

set prem_eco_avlb = prem_eco_avlb - 1

where flight_id = $flight_id;

Commit;

#cancelling a booking:

start transaction;

#lock rows for transaction

select * from seats where flight_id = $flight_id and seat_number = $seat_number for update;

#delete from booking table

delete from booking where booking_id = 'booking_id';

#delete from ticket table

```
delete from ticket where ticket_id = 'ticket_id';
#update seat table
update seats
set seat_status = 1, ticket_id = 0
where flight_id = $flight_id and seat_number = $seat_number;
#update available seats after each cancellation:
#for business class
update flights
set bus_cls_avlb = bus_cls_avlb + 1
where flight_id = flight_id;
#for economy class
update flights
set eco_cls_avlb = eco_cls_avlb + 1
where flight_id = flight_id;
#for premium economy class
update flights
set prem_eco_avlb = prem_eco_avlb + 1
where flight_id = flight_id;
Commit;

#Booking details based on booking_id
SELECT
```

```
    b.booking_id,

    t.ticket_id,

    f.flight_id,

    t.first_name,

    t.last_name,

    r.origin_airport,

    date_format(f.depart_timestamp,'%Y-%m-%d') as depart_date,

    date_format(f.depart_timestamp,'%H-%m-%s') as depart_time,

    r.destination_airport,

    date_format(f.arrive_timestamp,'%Y-%m-%d') as arrive_date,

    date_format(f.arrive_timestamp,'%H-%m-%s') as arrive_time,

    a.class_name,

    s.seat_number,

    CASE

        WHEN t.class_id = 1 THEN (f.bus_cls_fare * r.distance_miles)

        WHEN t.class_id = 2 THEN (f.eco_cls_fare * r.distance_miles)

        ELSE (f.prem_eco_fare * r.distance_miles)

    END AS fare

FROM

    booking b

        JOIN

    ticket t
```

```
        JOIN

    flights f

        JOIN

    air_class a

        JOIN

    routes r

                    JOIN

    seats s ON b.booking_id = t.booking_id

        AND b.flight_id = f.flight_id

        AND t.class_id = a.class_id

        AND r.route_id = f.route_id

WHERE

    b.booking_id = 0;
```

Section 9 Conclusions and Future Work

Conclusions:

Overall the system works well and it has many features that a working airline reservation application has. A person can book multiple tickets, sort the tickets as per the price, book his choice of seat, edit the user details, send the ticket details by mail to the user. We have also implemented the row level locking mechanism for concurrency control to avoid locking of tables when two or more users try to book the same seat. The application also displays the layover time if there is a non-direct flight. So the GoAir application provides user with lot of features which gives a feel of a real time user application.

Future Work:

For future work we wish to enhance the system with more user functionalities. The user should be given a choice to book itinerary, if the layover time is more than 6 hours user should be given a choice to book a hotel or book a car to travel in the layover time.

The system can be made more secure in order to prevent it from security threats like SQL injection attack and securing the user’s personal details in the database.
One more module can be added to the system to handle the payment for booking. Currently the system has no payment gateway to process the payments.

References

https://dev.mysql.com/doc/refman/5.5/en/internal-locking.html

https://en.wikipedia.org/wiki/Concurrency_control
http://www.britishairways.com/travel/home/public/en_us

Appendix

Layover - It is the time calculated from flight arrival to the next flight departure for a passenger.