**J.S.S. Mahavidyapeetha's**

# SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING, MYSURU

## Topic : Simulation of Airline Reservation System.

(Project work as a part of partial fulfillment of course of Database Management System)


**Project team members :**

Samarth Deyagond
USN : 4JC14CS087
Email Id: deyagondsamarth@gmail.com

Rachana K
USN : 4JC14CS073
Email Id: rachanademilovato@gmail.com

**Guides :**

Prof. Manimala S

Prof. Trisiladevi Nagavi

Dr. Anilkumar K M

Prof. G. Madhusudan

Prof. Vasantha Raghavan

# TABLE OF CONTENTS

## Contents

# CHAPTER 1 : INTRODUCTION

# 1 Introduction

*The economic status of a nation is measured by several parameters and transport sector is one main parameter among all.*

The Airline Reservation System project is an implementation of a general Airline Ticketing website like Orbitz and MakeMyTrip, 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, 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.

The website is named as **ReplicaDB** and is the database to access and book the domestic flights whose cost of service is extremely reasonable.

## 1.1 Aim

To simulate an airline reservation database using which people can find flight schedule and enjoy all sorts of services provided by a particular airline company.

## 1.2 Objective

The main objectives of our project are :

- To explore the features of Database Management System.
- Experimenting basic, intermediate and advanced SQL.
- Entity Relationship Modeling (ER diagram).
- Database design and normalization.
- Accessing database using suitable front-end tool(s).
- Query processing to use databases for applications.
- Gaining knowledge about the internals of the database system.

## 1.3 Scope

The scope of the project is as follows :

- The project gives the provision to book one-way journey ticket to the users.
- The database is exclusively for domestic flights with few destinations.
- The front-end is designed only for the client/user end and not the admin end as linking admin end is not a secure plan because of threats of SQL injections and similar malicious activities.

# CHAPTER 2 : SYSTEM REQUIREMENTS AND ANALYSIS

# 2 System Requirements

## 2.1 Software Requirements

- Operating System : Windows 10/ Ubuntu 16.04 or equivalent.
- Software : Sublime Text editor, MySQL server-client, any stable browser.
- Front-end : Python-Flask web framework
- Back-end : MySQL server-client version 5.7
- Middleware : Python MySQLdb module.

## 2.2 Hardware Requirements

- Processor : 2.2 GHz
- RAM : 4 GB
- Input Device : Standard keyboard and mouse
- Output Device : Monitors with decent resolution.

## 2.3 Functional Requirements

- Collection of flight schedule :

  The schedule of the flight services from different airline companies is to be collected and populated into the database.

- Redundancy check:

  The schedule must be verified for redundancy and clashes in the runway timings.

- Graphical User Interface :

  A graphical user interface has been designed to pass the control to the built system and display the results.

## 2.4 Non-Functional requirements

- Usability :

  The graphical user interface (GUI must be user-friendly.

- Accuracy and perfection :

  The results of the queried operation must generate the required results and make the expected changes as well; with highest accuracy.

## 2.5 Input Requirements

- Input of city names as source and destinations.
- Input of date of journey and number of seats to be booked.

## 2.6 Output Requirements

- Acknowledgement of proper ticket generation and updation of the database.

# CHAPTER 3 : SYSTEM DESIGN AND ANALYSIS

# 3 System Design

## 3.1 Database Implementation

### 3.1.1 Tools used

MySQL server-client was employed for the implementation of the database which has the entities and relations as shown in the ER-diagram in figure 1. MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL 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.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL 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).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

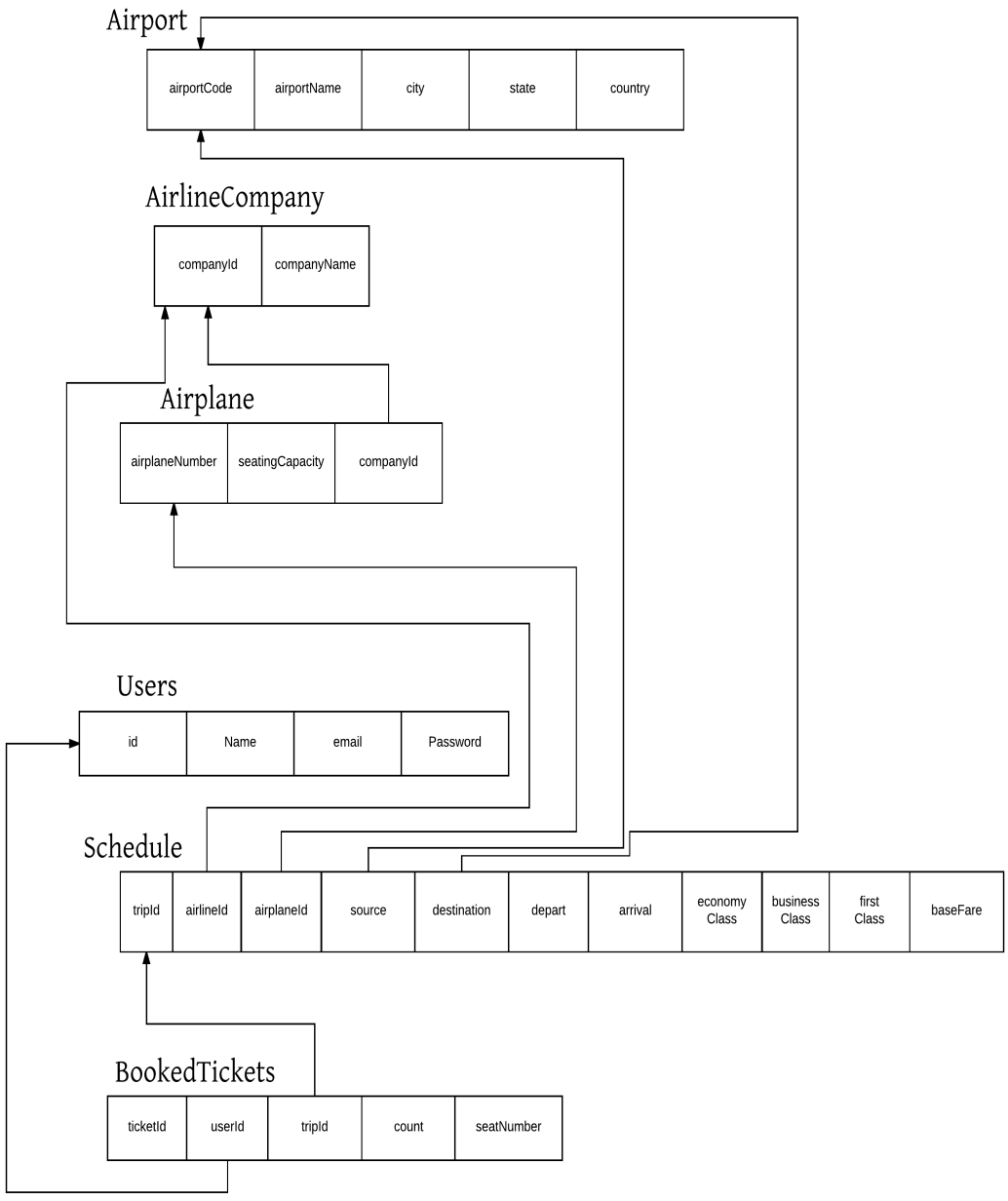


Figure 1: Typical schema Diagram of the proposed database

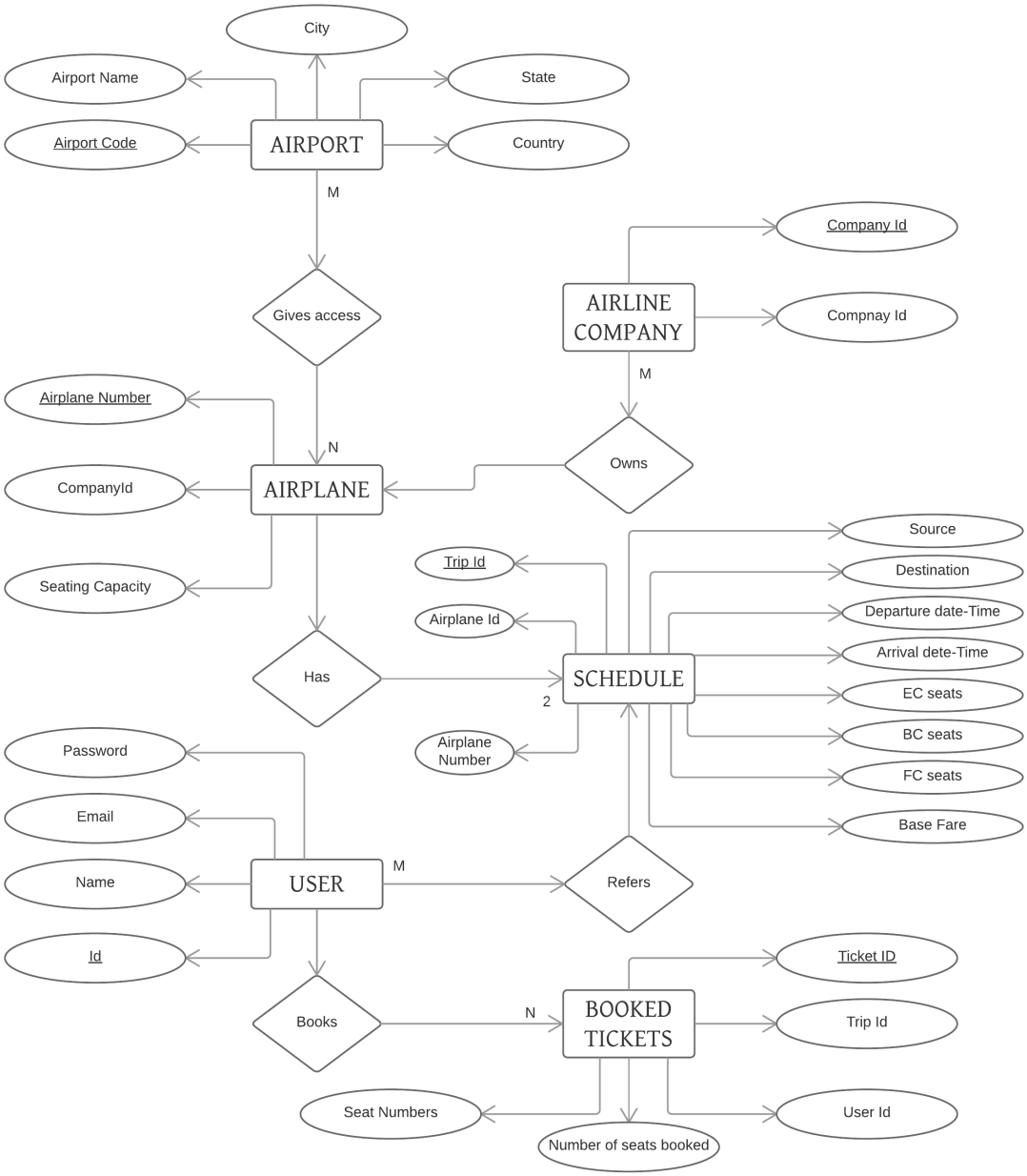


Figure 2: Typical ER Diagram of the proposed database

### 3.1.2 Design and building of database.

The replicaDB has six entities as shown in the ER diagram which are Airline Company, Airplane, Airport, Booked Tickets, Schedule and Users whose attributes go like this:

- Airport : Airport Code, Airport Name, City, State, Country
- Airline Company : Company Id, Company Name
- Airplane : Airplane Number, Seating Capacity, Company Id
- Schedule: Trip Id, Airline Id, Airplane Id, Source, Destination, Depart, Arrival, Remaining seats in Economy Class, Remaining seats in Business Class, Remaining seats in First Class, Base Fare
- Booked Tickets : Ticket Id, user Id, trip Id, Count, Class, Allotted Seat Numbers

The MySQL queries to create the above tables are as follows (in the sequence as above):

```
CREATE airport (airportCode INT PRIMARY KEY AUTO_INCREMENT,
airportName VARCHAR(100) NOT NULL, city VARCHAR(50) NOT NULL,
state VARCHAR(50) NOT NULL, country VARCHAR(50) NOT NULL);

CREATE airlineCompany (companyId INT PRIMARY KEY AUTO_INCREMENT,
companyName VARCHAR(100) NOT NULL);

CREATE airplane (airplaneNumber INT PRIMARY KEY AUTO_INCREMENT,
seatingCapacity INT NOT NULL, companyId INT NOT NULL);

CREATE schedule (tripId INT PRIMARY KEY AUTO_INCREMENT,
airlineId INT NOT NULL, airplaneId INT NOT NULL, source VARCHAR(100)
NOT NULL, destination VARCHAR(100) NOT NULL,
depart DATE, arrival DATE, economyClass INT NOT NULL,
businessClass INT NOT NULL, fisrtClass INT NOT NULL,
baseFare INT NOT NULL);
```

```
CREATE bookedTickets (ticketId INT PRIMARY KEY AUTO_INCREMENT,
userId INT NOT NULL, tripId INT NOT NULL,count INT NOT NULL,
class VARCHAR(50) NOT NULL, seatNumbers VARCHAR(100));
```

- Airport

| Airport Code | Airport Name | City | state | Country |
|---|---|---|---|---|
| 1 | ChatrapathiShivaji International Airport | Mumbai | Maharashtra | India |
| 2 | Kempegowda International Airport | Bengaluru | Karnataka | India |

- Airline Company

| CompanyId | CompanyName |
|---|---|
| 11 | AirAsia |
| 12 | AirIndia |

- Airplane

| Airplane Number | Seating Capacity | CompanyId |
|---|---|---|
| 10 | 100 | 11 |
| 20 | 100 | 12 |

- Users

| Id | Name | Email | Password |
|---|---|---|---|
| 1 | Ria | rria67@gmail.com | flyingsquirrel |
| 2 | Ishu | ishuchetty14@yahoo.com | rose |
| 3 | Dummakka | dummakkademilovato@gmail.com | Dummi |

- Schedule

| TripId | AirlineId | AirplaneId | Source | Destination | Depart | Arrival | Economy | Business | First | Basefare |
|---|---|---|---|---|---|---|---|---|---|---|
| 90 | 11 | 10 | Mumbai | Bengaluru | 2016-11-09 | 2016-11-09 | 60 | 15 | 10 | 5000 |
| 91 | 12 | 20 | Bengaluru | Delhi | 2016-11-09 | 2016-11-09 | 60 | 15 | 15 | 6000 |

- Booked Ticket

| TicketId | UserId | TripId | Count | Class | SeatNumbers |
|---|---|---|---|---|---|
| 111 | Ria | 90 | 3 | Economy | 58 to 60 |
| 2222 | Ishu | 91 | 1 | Business | 75 |

Typical view of populated tables in the database.

#### 3.1.3 Normal form description of the designed database

**Normalization :**

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible. There are many type of anomalies :

- **Update anomalies** - When the data items are scattered and are not linked properly, then strange things may happen. Such a situation is update anomaly.
- **Delete anomalies** - A data is deleted and a part of it is remaining undeleted also consuming memory because of unawareness. This leads to Delete Anomalies.
- **Insert anomalies** - Trying to insert data in a record that doesn't exist leads to Insert anomalies.

Normalization is a method to remove all these anomalies and bring the database to a consistent state. ***Our Database is in Boyce-Codd Normal Form as per our analysis***. The justification goes like this -

- DB holds First Normal Form because all the attributes are atomic in nature.
- DB also holds Second Normal Form because every non-primary attribute is fully functionally dependent on the prime attribute in the given table.
- The Boyce-Codd Normal Form also holds good as for any non-trivial functional dependency X->A, X is a super key.

Thus replicaDB is in BCNF.

## 3.2 Front-end Implementation and Connectivity

### 3.2.1 Tools used

- **Python-Flask** :

  Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

- **What is framework ?**

  Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

- **WSGI :**

  Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

- **Werkzeug :**

  It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

- **Jinja2 :**

  Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application.

# CHAPTER 4 : SYSTEM IMPLEMENTATION

# 4 System Implementation

## 4.1 Introduction

Software design of a system is a process of problem solving and planning for a software solution. It is the process or art of dening the hardware and software architecture, components, modules, interfaces, and data for a computer system to satisfy specied requirements. One could see it as the application of systems theory to computing. The design of the system is essentially a blueprint, or a plan for a solution for the system. We consider a system to be set of components with the clear and dened behavior, which interacts in a xed, dened manner, to produce some behaviour or services to its environment.

Our system is used to query the schedule of flights and book tickets, cancel them as well.

## 4.2 Overview Of the project :

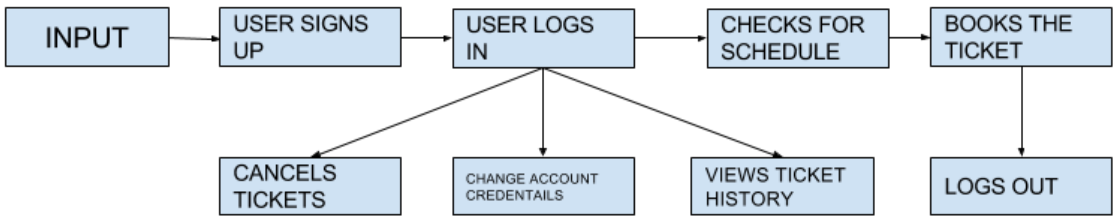


Figure 3: Block Diagram Representing Overview of Our Project

### 4.3 Implementation Steps

The entire application is coded in Sublime Text Editor and the tables were populated using the terminal of MySQL server-client interface. We have achieved all the above mentioned objectives in section 1.2

- The database was populated with the schedule and other empty tables.
- The empty table **users** was populated as and when people signed up.
- The source-destination query was handled by the application and further then exceeded to booking handling which populated the **bookedTickets** table.
- Then 'cancel ticket 'operation was implemented to delete the tuple from the bookedTickets table. Because canceling a ticket will require the booking history of the particular user, the operation of *fetch booking history* was also implemented.

The screen-shots of different windows for specific operation are as follows :

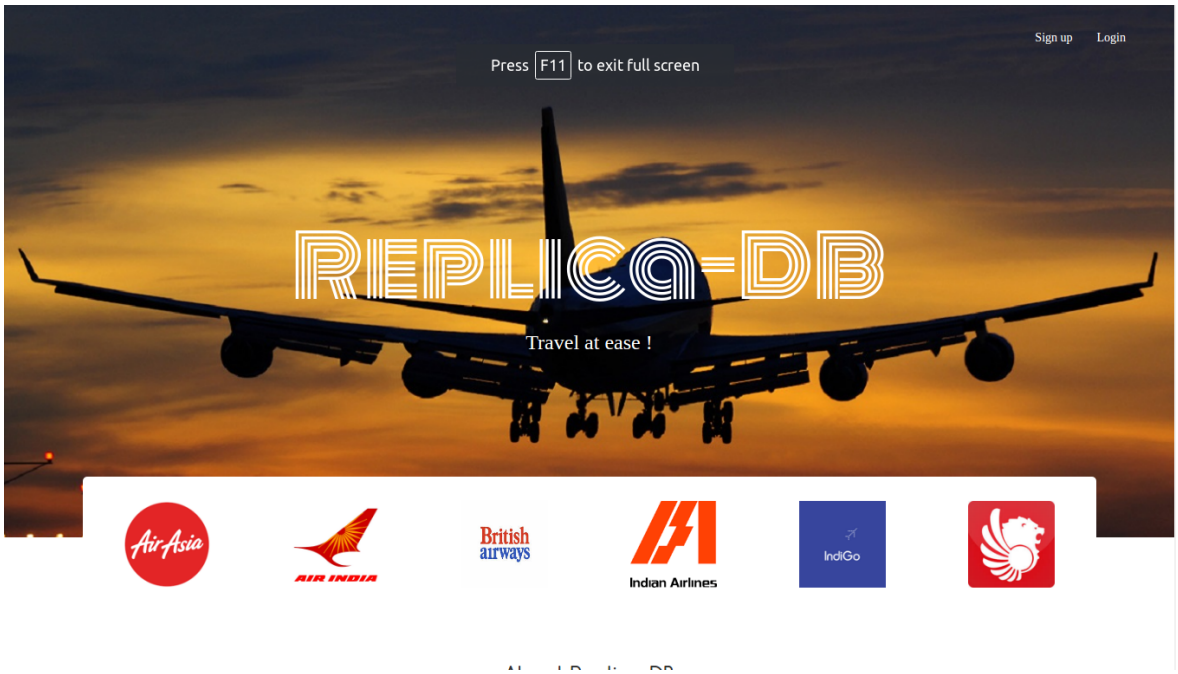


Figure 4: The home page of ReplicaDB

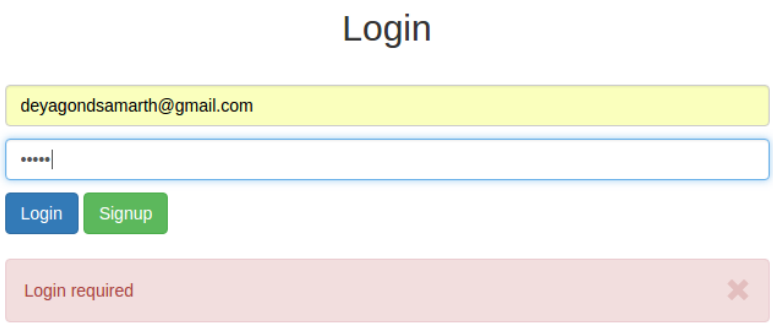


Figure 5: The Login page for the user

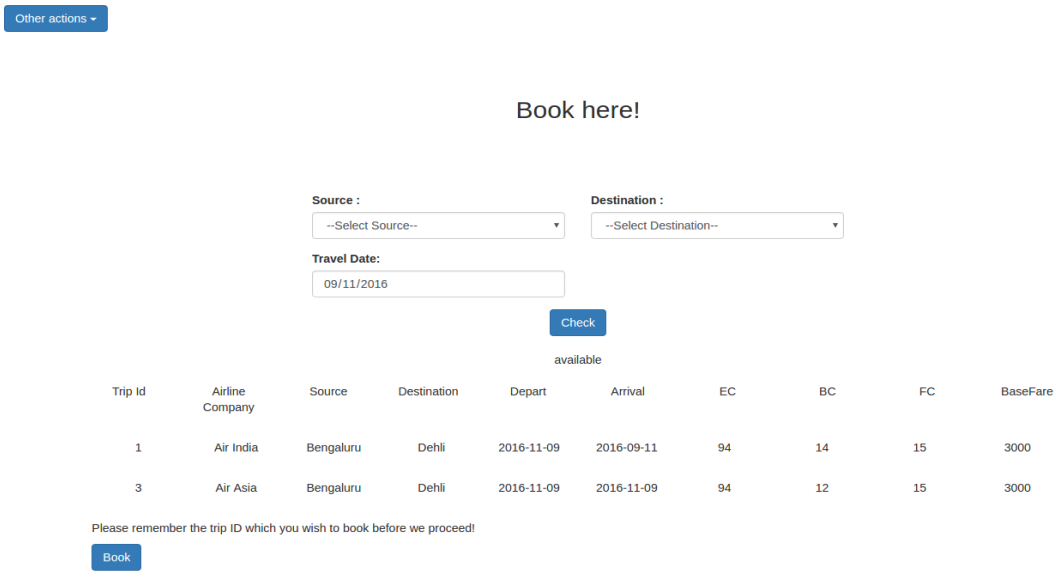


Figure 6: The booking page for the user along with schedule display

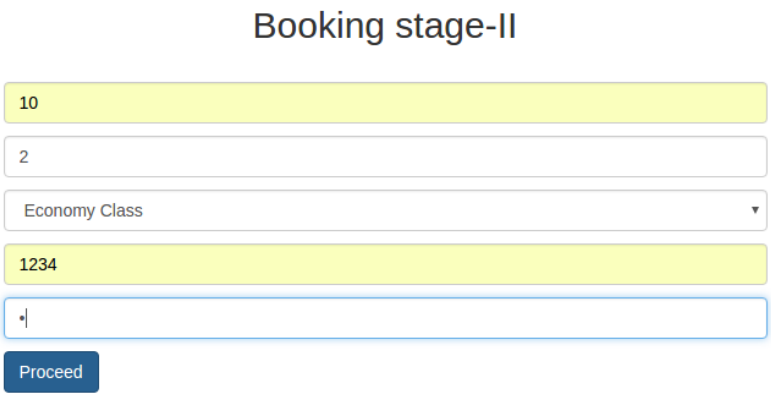


Figure 7: The stage-2 booking page for the user along with schedule display

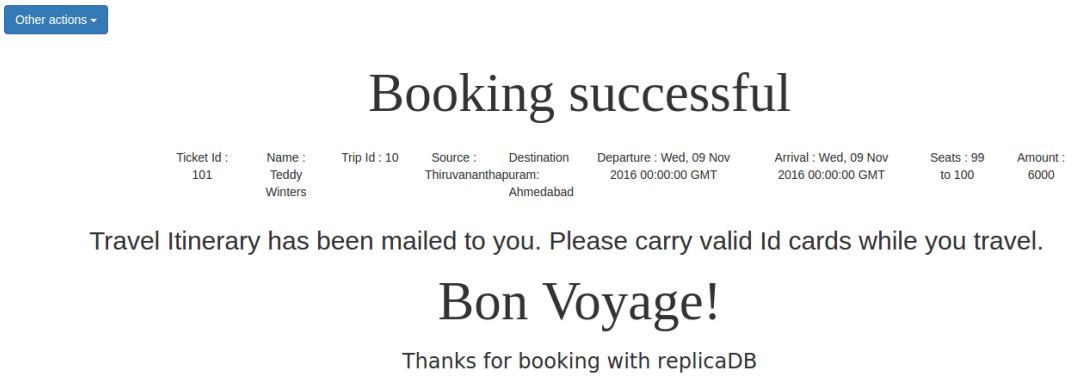


Figure 8: The generated acknowledgment and travel itinerary

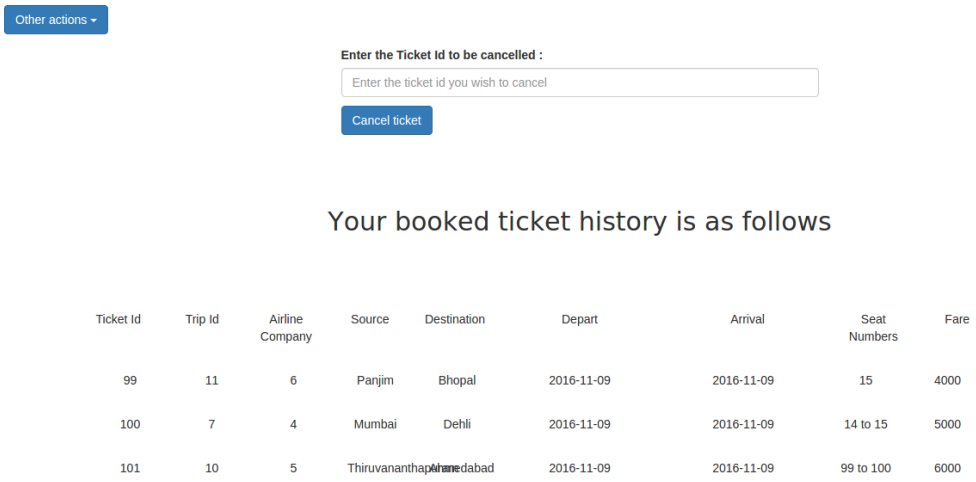


Figure 9: The interface to cancel the booked tickets along with display of booking history.

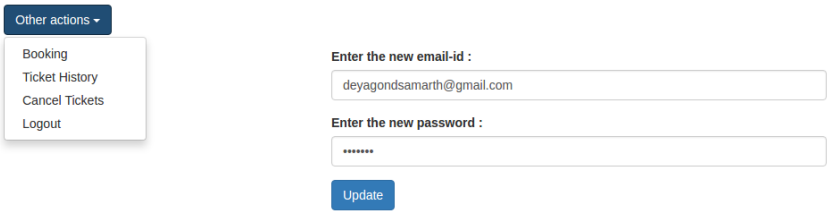


Figure 10: A simple interface to update the account credentials. Also navigation drop-down from page to every other page.

This way the database so created was connected with the front-end application and it was tested for the reflection of the changes from the front-end onto the database.

# CHAPTER 5 : SYSTEM TESTING AND CRITICAL ANALYSIS OF RESULTS

# 5 System Testing and Critical Analysis of results

## 5.1 Uniting testing and result analysis

Incremental unit testing involves the design of test cases that validate integral program logic is working properly, and the program input produces valid output.

| Serial no | Test Case | Description | Expected result | Actual result | Remark |
|---|---|---|---|---|---|
| 1 | Populating database | Checks for INSERT queries | successful insertion | insertion successful | PASS |
| 2 | Updating database | Checks for UPDATE queries and constraint handling | successful update | updation successful | PASS |
| 3 | Deletion of tuples | Checks the DELETE queries | successful deletion | Deletion successful | PASS |

Figure 11: Unit testing result

## 5.2 Integration testing

After the above tests were successfully carried out, the integration testing was conducted where the database (the back-end) was connected to the front-end application which was developed using Python-Flask frame work. It was checked for the same operations which were verified in unit testing with the only change that Python MySQLdb module was used as the middle ware. The functioning of the front-end was pretty good and every expected result was observed. ***So, integration testing as a whole was given the remark as PASS.***

# CHAPTER 6 : CONCLUSION AND FUTURE WORK

# 6 Conclusion and Future Work

The main aim of this project ReplicaDB was to simulate the airline reservation system similar to existing systems like MakeMyTrip or Orbitz. However, in future we are looking forward to have the following enhancements :

- Implementation of package booking including return tickets or round-trip booking concept.
- To suggest the schedule as per the rating of the passengers who have traveled in a particular airline company's flight. (*For this; to implement, we need lots of dataset to have the recommender system ready using machine learning algorithms*). To accomplish this,we need to run this system for some period of time which is actually decided by the extent of usage of the system by people.

# References


- Python Flask documentation - `http://flask.pocoo.org/docs/0.11/`
- MySQL documentation - `https://help.ubuntu.com/lts/serverguide/mysql.html`
- MySQLdb beginner's documentation - `https://wiki.python.org/moin/BeginnersGuide`