Project Specification Document

FlyAway An Airline Booking Portal

End of Phase 2-Become a back-end expert Project

Student: Thirumavalaven

vthiru97@gmail.com

Full Stack Java Developer

Master's Program

Project Objective

As a Full Stack Developer, design and develop an airline booking portal named as FlyAway

Developer Details

The project is developed by Thirumavalaven. I worked as a Service Delivery Specialist at IBM India Pvt. Ltd.

The code for this project is hosted at https://github.com/Thiru97/Simplilearn2023-FlyAway

Project Details

FlyAway is a ticket-booking portal that lets people book flights on their website. As a Full Stack Developer, design and develop an airline booking portal named as FlyAway. Use the GitHub repository to manage the project artefacts.

The Website should meet the following requirements

- A search form in the homepage to allow entry of travel details, like the date of travel, source, destination, and the number of persons.
- Based on the travel details entered, it will show the available flights with their ticket prices.
- Once a person selects a flight to book, they will be taken to a register page where they must fill in their personal details. In the next page, they are shown the flight details of the flight that they are booking, and the payment is done via a dummy payment gateway. On completion of the payment, they are shown a confirmation page with the details of the booking.

For the above features to work, there will be an admin backend with the following features:

- An admin login page where the admin can change the password after login, if he wishes
- A master list of places for source and destination
- A master list of airlines
- A list of flights where each flight has a source, destination, airline, and ticket price

The goal of the company is to deliver a high-end quality product as early as possible.

Sprint Planning and Tasks Achieved

The project is planned to be completed in 1 sprint. Tasks assumed to be completed inthe sprint are:

- Creating the flow of the application
- Initializing git repository to track changes as development progresses.
- Writing the Java program to fulfill the requirements of the project.
- Testing the Java program with different kinds of User input
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Project Overview

The main objectives of this Projects are

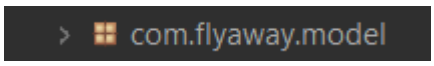
- To gain an understanding of core concepts of the Java Programming Language (abstraction,polymorphism, inheritance, and encapsulation),
- Embrace the Eclipse Integrated Development Environment (IDE),
- Understand the Agile software development life cycle
- Gain familiarity with Java data structures for object-oriented applications.
- Familiarize with concepts of Servlets , Servlet Filter, HTTP Methods
- Familiarize with concepts of Web Application Development using Servlets
- Learn Maven, A build Automation tool and handle dependencies
- To learn SQL and do CRUD Operation(Create, Read, Update, Delete)
- To learn Hibernate and design a Web Application with Servlets and perform CRUD Operation (Create, Read, Update, Delete).
- To learn Hibernate's Association and perform @OnetoOne , @OnetoMany @ManytoMany Associations with different tables of a database
- Learn Cookie and HTTP Session Management and Perform User's Session Management in a Web Application
- Learn MVC Architecture and build a Web Application using Servlets, JSP, Hibernate and perform CRUD Operations

Implemented Java Concepts

This section will highlight the Java concepts used to create the FlyAway Airline website. HTML, CSS, JSP,JSTL, Servlets, Servlet Filters, HTTP Methods, SQL, JDBC Connection for SQL , Java Persistence API, Hibernate, HTTP Sessions and Cookie , Servlet MVC Architecture ,Collections framework, Flow Control, Exception Handling, are the core concepts used in this program. ***The entire Application was built in JAVA 19***

Packages

I chose to create a package dedicated to the FlyAway Airlines as per the naming standards.

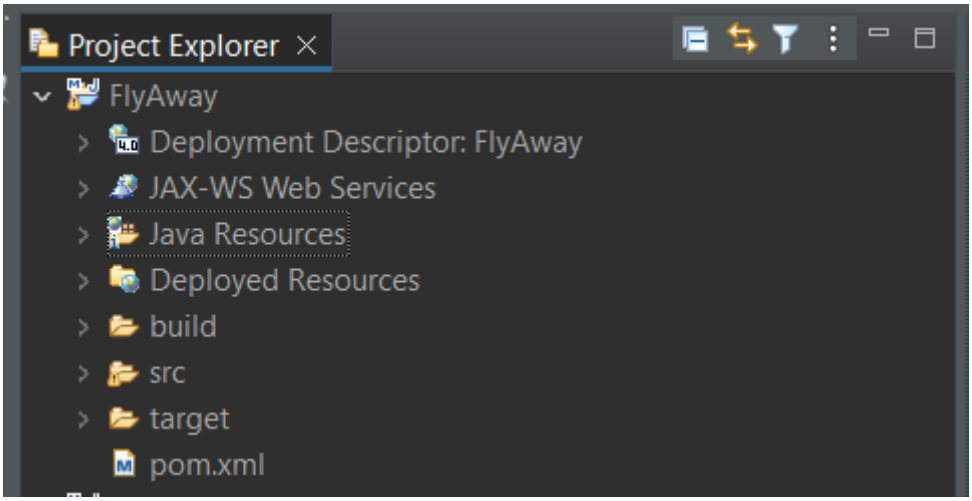


Maven

Maven is an open-source build tool from Apache Group. It is a widely-used and popular tool to build, publish, and deploy software development projects. It is used to build and manage diverse Java-based projects. A powerful project management tool, it is based on the project object model (POM) concept.

STEP 1:

Create a new Maven project via **File** >**New**> **Other>** Maven> **Maven Project**. Project name as FlyAway. Once Done Maven will build the project and include a pom.xml file



STEP 2:

Add the Following dependencies in pom.xml

```
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.0.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.14.Final</version>
    </dependency>
        <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
</dependencies>
```

Servlet MVC Architecture

We will create our web application that implements the Model View Controller (MVC) design pattern, using basic Servlets and JSPs.

Model-View-Controller (MVC) is a pattern used in software engineering to separate the application logic from the user interface. As the name implies, the MVC pattern has three layers.

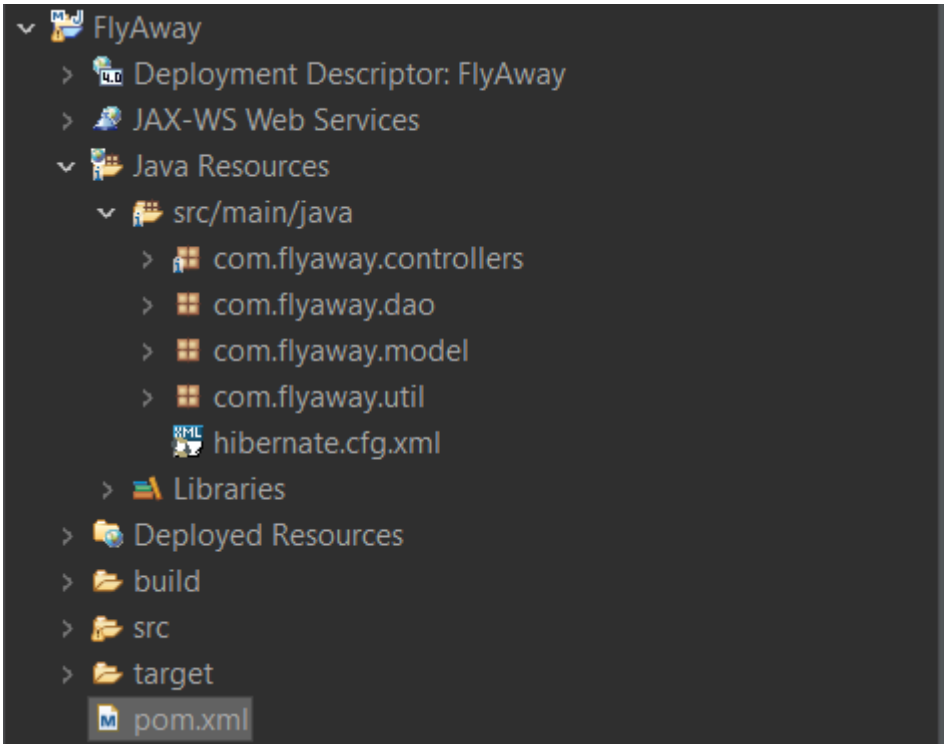
The Model defines the business layer of the application, the Controller manages the flow of the application, and the View defines the presentation layer of the application.

Although the MVC pattern isn't specific to web applications, it fits very well in this type of applications. In a Java context, the Model consists of simple Java classes, the Controller consists of servlets and the View consists of JSP pages.

Here're some key features of the pattern:

- It separates the presentation layer from the business layer
- The Controller performs the action of invoking the Model and sending data to View
- The Model is not even aware that it is used by some web application or a desktop application

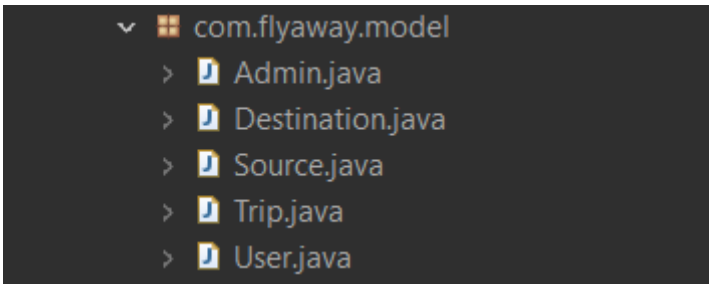
Below is the Maven and MVC Architecture based package Structure of FlyAway Web Application



Let's have a look at each layer.

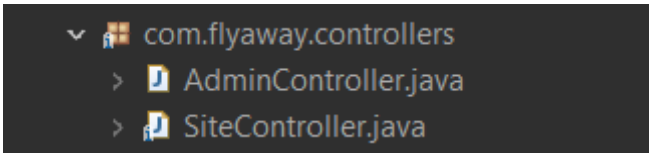
The Model Layer

This is the data layer which contains business logic of the system, and also represents the state of the application. It’s independent of the presentation layer, the controller fetches the data from the Model layer and sends it to the View layer.



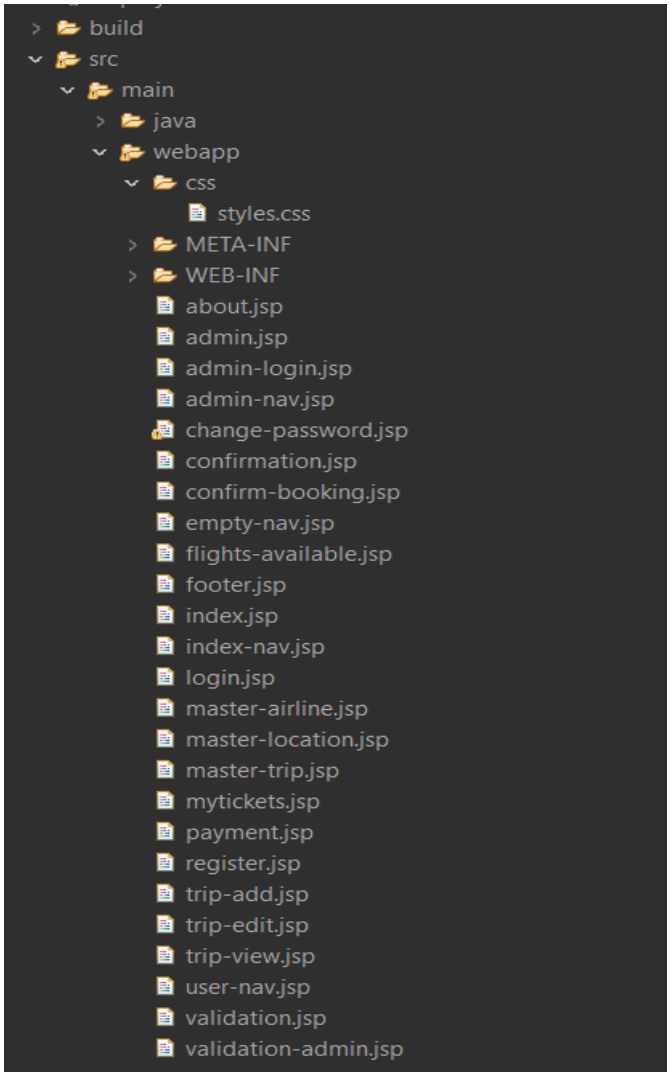
The Controller Layer

Controller layer acts as an interface between View and Model. It receives requests from the View layer and processes them, including the necessary validations. The requests are further sent to Model layer for data processing, and once they are processed, the data is sent back to the Controller and then displayed on the View.



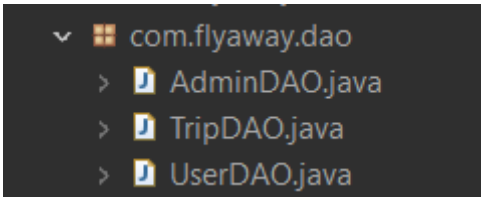
The View Layer

This layer represents the output of the application, usually some form of UI. The presentation layer is used to display the Model data fetched by the Controller.



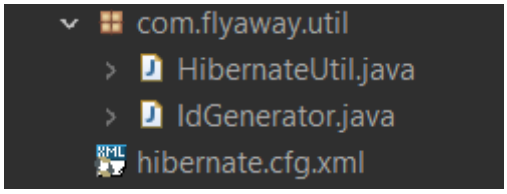
The DAO Layer

DAO stands for Data Access Object. DAO Design Pattern is used to separate the data persistence logic in a separate layer. This way, the service remains completely in dark about how the low-level operations to access the database is done. This is known as the principle of **Separation of Logic**.



The Util Package

The Util package consists of helper methods that our web application needs. We have a HibernateUtil.java files which creates a session factory with the help of hibernatecfg.xml. The xml file hibernatecfg.xml contains the necessary configuration needed for our hibernate to run. The IdGenerator.java files create a random customized ID number for our flights and Trips



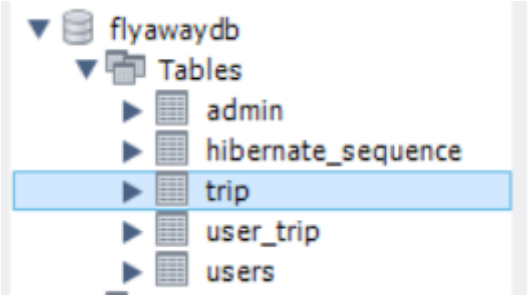
Database

I have used MySQL database. I also used an ORM (Object Relational Mapping) tool like Hibernate. Hibernate makes our project loosely coupled with respect to database. In future we can easily change from MySQL database to other database like Postgres Database. All we need to do is change the configuration file

MySQL

MySQL is a widely used relational database management system (RDBMS).MySQL is free and open-source. MySQL is ideal for both small and large applications.

I have used MySQL 8.0.33 in this application and below is the schema



Hibernate

Hibernate is an open source Object-Relational Persistence and Query service for any Java Application. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from most common data persistence related programming tasks. Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.

Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

In order to connect our database with Hibernate we need to define the configurations which includes the database port number, username and password. We define these configuration in hibernatecfg.xml file

```
  -//Hibernate/Hibernate Configuration DTD 3.0//EN (doctype with catalog)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5
6● <hibernate-configuration>
7●     <session-factory>
8         <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
9         <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/flyawayDB</property>
10        <property name="hibernate.connection.username">root</property>
11        <property name="hibernate.connection.password">root@thiru1234</property>
12
13        <property name="hibernate.show_sql">true</property>
14        <property name="hibernate.format_sql">true</property>
15        <property name="hbm2ddl.auto">update</property>
16    </session-factory>
17 </hibernate-configuration>
```

Hibernate Association

Association in hibernate tells the relationship between the objects of POJO classes i.e. how the entities are related to each other. Association or entities relationship can be unidirectional or bidirectional.

Below is the associations used by Flyaway project

@ManytoMany Association

Two items are said to be in Many-to-Many relationship if many occurrence of item are belong to the many occurrences of other item and vice versa

In this application Trip and Users are mapped with many to many association as user can book multiple tickets

```java
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(name = "user_trip",
joinColumns = @JoinColumn(name = "trip_id"), inverseJoinColumns = @JoinColumn(name = "user_id"))
List<User> users = new ArrayList<User>();
```

```java
@ManyToMany(mappedBy = "users", fetch = FetchType.EAGER)
List<Trip> trips = new ArrayList<Trip>();
```

@Embeddable and @Embedded

The @Embeddable and @Embedded annotations in Hibernate are used to map an object's properties to columns in a database table. These annotations are used in combination to allow the properties of one class to be included as a value type in another class and then be persisted in the database as part of the containing class.

```java
@Embedded
private Source source;

@Embedded
private Destination destination;
```

```java
@Embeddable
public class Destination {
    @Column(name = "destination_country")
    protected String destinationCountryName;

    @Column(name = "destination_city")
    protected String destinationCityName;

    @Column(name = "destination_airport")
    protected String destinationAirportName;

    public String getDestinationCountryName() {
        return destinationCountryName;
    }
```

```java
@Embeddable
public class Source {

    @Column(name = "source_country")
    protected String sourceCountryName;

    @Column(name = "source_city")
    protected String sourceCityName;

    @Column(name = "source_airport")
    protected String sourceAirportName;
```

Servlets

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, and then send a response back to the webserver.

Properties of Servlets are as follows:

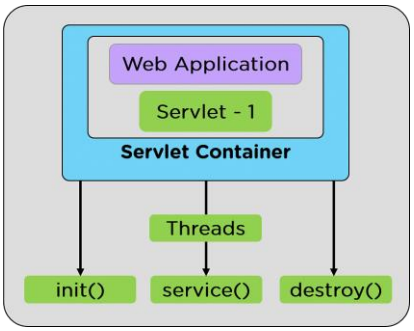
- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Execution of Servlets basically involves six basic steps:

- The clients send the request to the webserver.
- The web server receives the request.
- The web server passes the request to the corresponding servlet.
- The servlet processes the request and generates the response in the form of output.
- The servlet sends the response back to the webserver.
- The web server sends the response back to the client and the client browser displays it on the screen.

There are several varieties of interfaces and classes available in the Servlet API. Some of them are as follows:

- HTTP Servlet
- Generic Servlet
- Servlet Request
- Servlet Response



To write a Servlet, the user needs first to implement the Servlet Interface, directly or indirectly, using the following import command.

```
import javax.servlet.*;
```

Once the Servlet interface is imported, and we inherit the HTTP Class, we begin with the Java Servlet's life cycle.

In the life cycle of a servlet, we have mainly three stages, which are mentioned below.

- init()
- service()
- destroy()

We call these methods at their respective stages. The methods are resolved by generating the essential threads for the process to get executed.

The service() method is the heart of the life cycle of a Java Servlet. Right after the Servlet's initialization, it encounters the service requests from the client end.

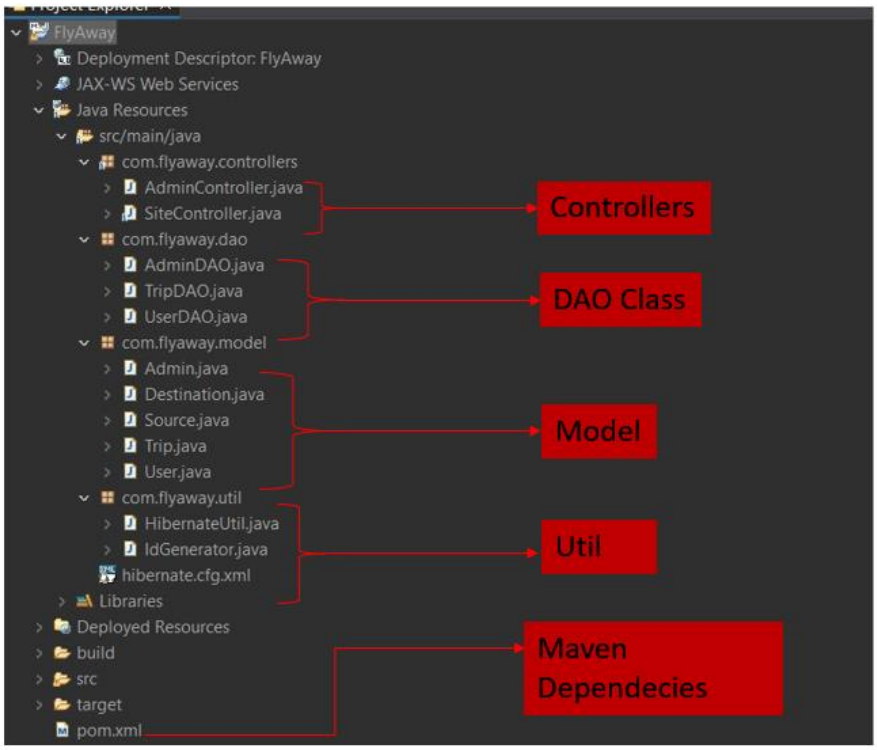
The client may request various services like:

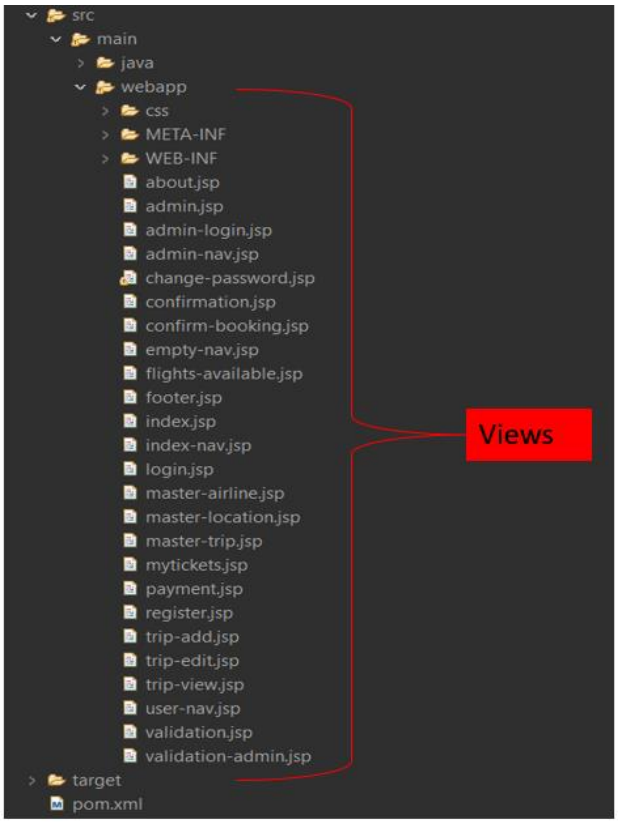
- GET
- PUT
- UPDATE
- DELETE

The service() method takes responsibility to check the type of request received from the client and respond accordingly by generating a new thread or a set of threads per the requirement and implementing the operation through the following methods.

- doGet() for GET
- doPut() for PUT
- doUpdate() for UPDATE
- doDelete() for DELETE

Project Structure



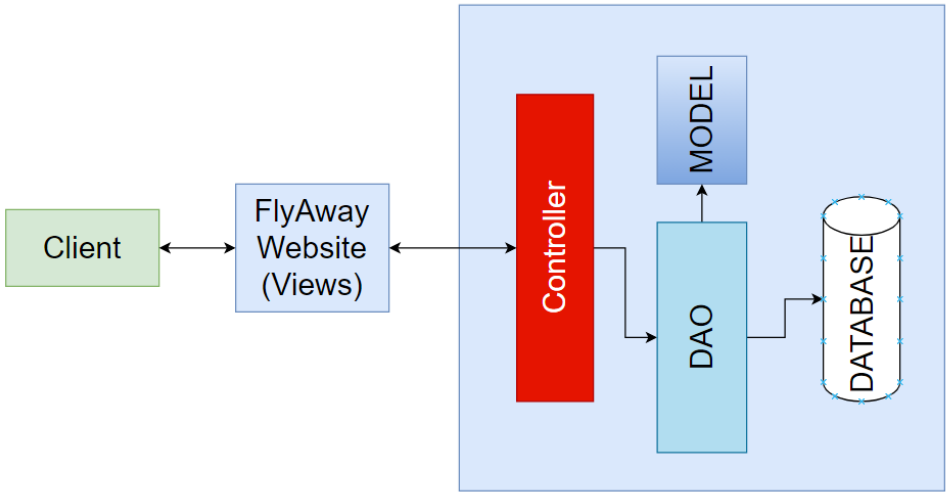


Control Statements

The program utilizes the following control statements to direct the desired logic:

- *while* loop – Controls the program flow by prompting the User for main menu and the business options sub-menu, performing the desired operations, and terminates when theUser wishes to quite the program.
- *Switch* Statement Executes the desired code statements associated with the main menuand the business level options sub-menu based on the value entered by the user.

Flow Chart



Application Flow Chart (page 3:3)

Future Improvement Areas

- Passenger details can also be added by making it as a @onetoMany association with User and Trip models
- User can also change password same as the admin
- As of now user can only select from the list of cities available in the menu in future we can dependent dropdown with the help of JavaScript.
- As of now user can only select a one way trip, Round trip can also be added as a feature in the future.

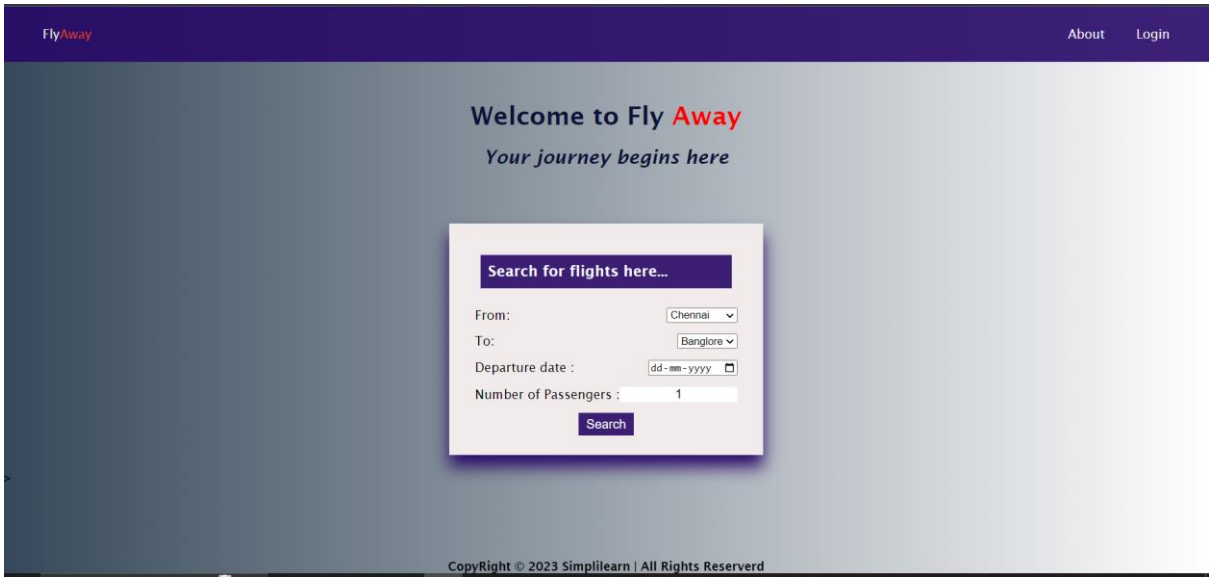
GitHub Repository

I have pushed my code and associated documentation to the following GitHub repository:

https://github.com/Thiru97/Simplilearn2023.git

Web Application Screenshots

Home Page



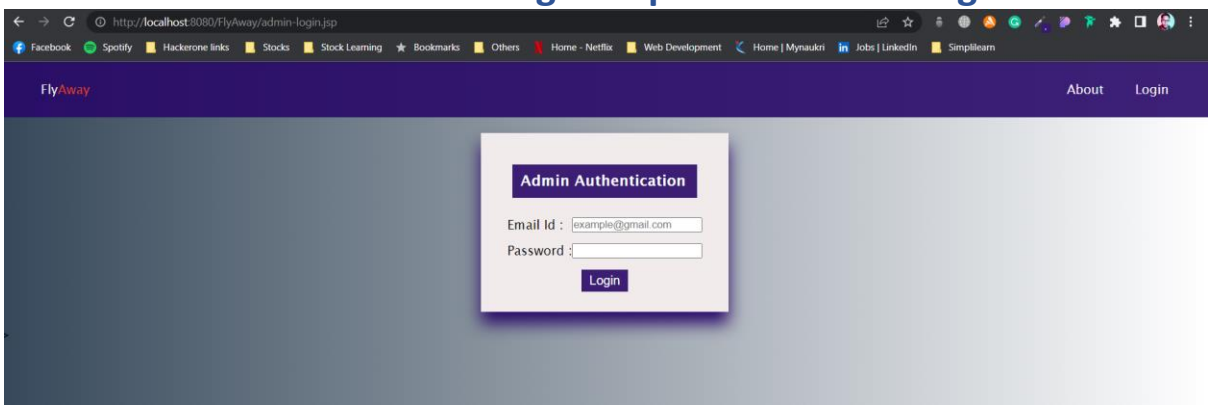
Admin can login by changing the url as localhost:8080/FlyAway/admin-login.jsp

Where he has to enter his credentials

Default Email Id: admin@flyaway.com

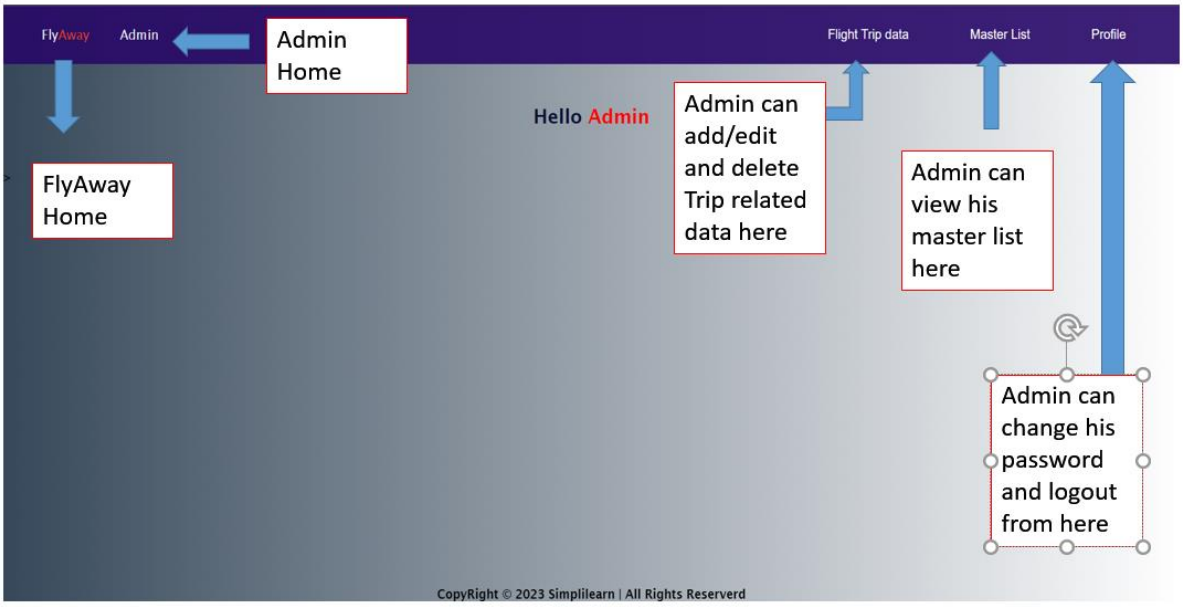
Default Password: admin

However admin should change his password after login

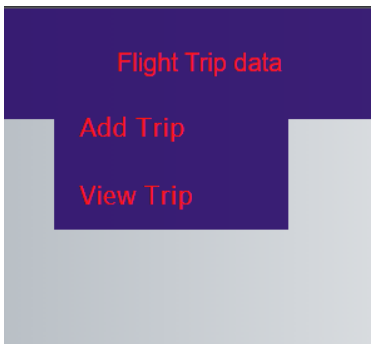


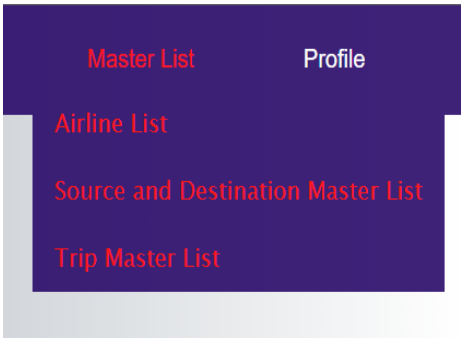
Admin Dashboard

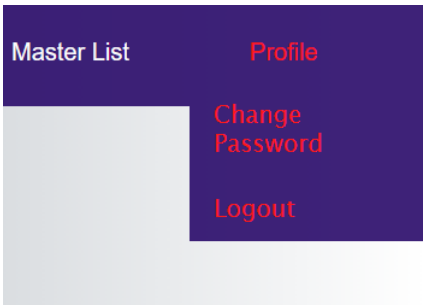
NOTE: If session is inactive for more than 5 minutes admin will be logged out



Admin Navigation Bar Drop Down menu



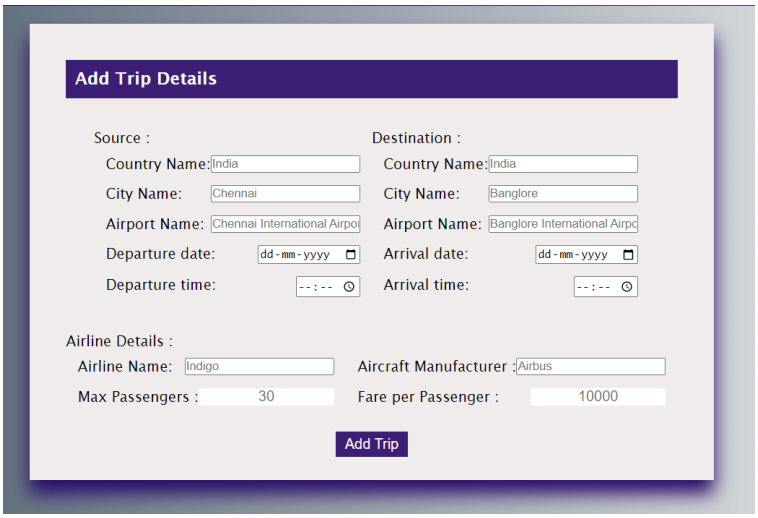




Flight Trip Data

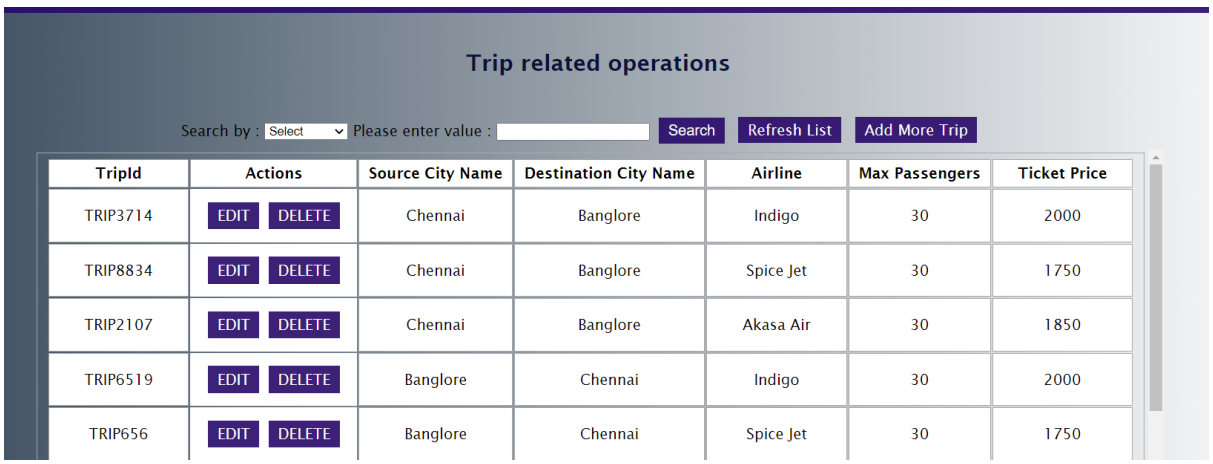
Admin can add trip details here:

Note: Trip ID and Flight number are generated by IdGenerator.java



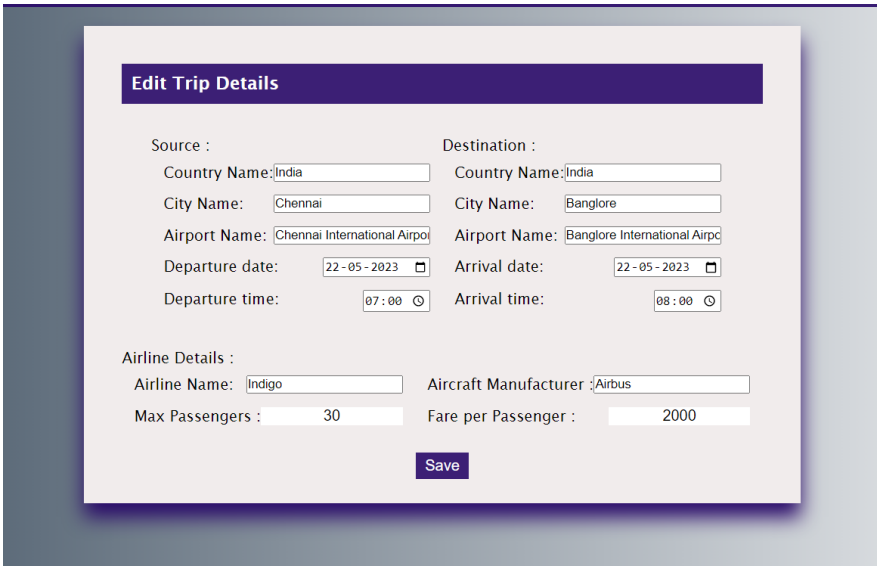
View Trip data

Admin can edit / delete trips from here. He can also use search by to narrow the results

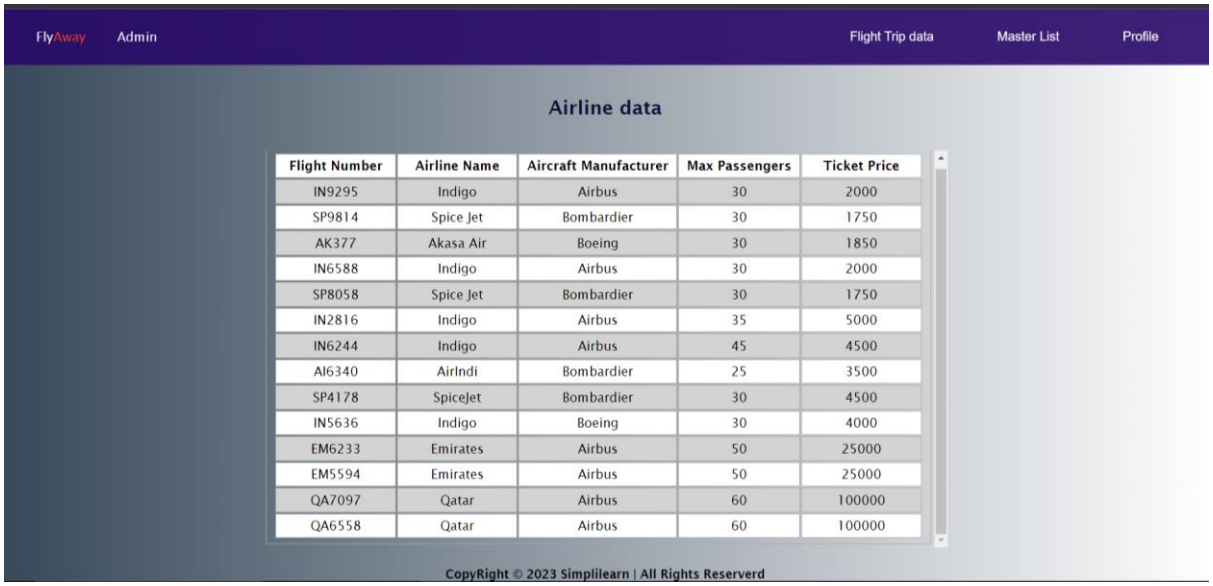


Edit Trip data

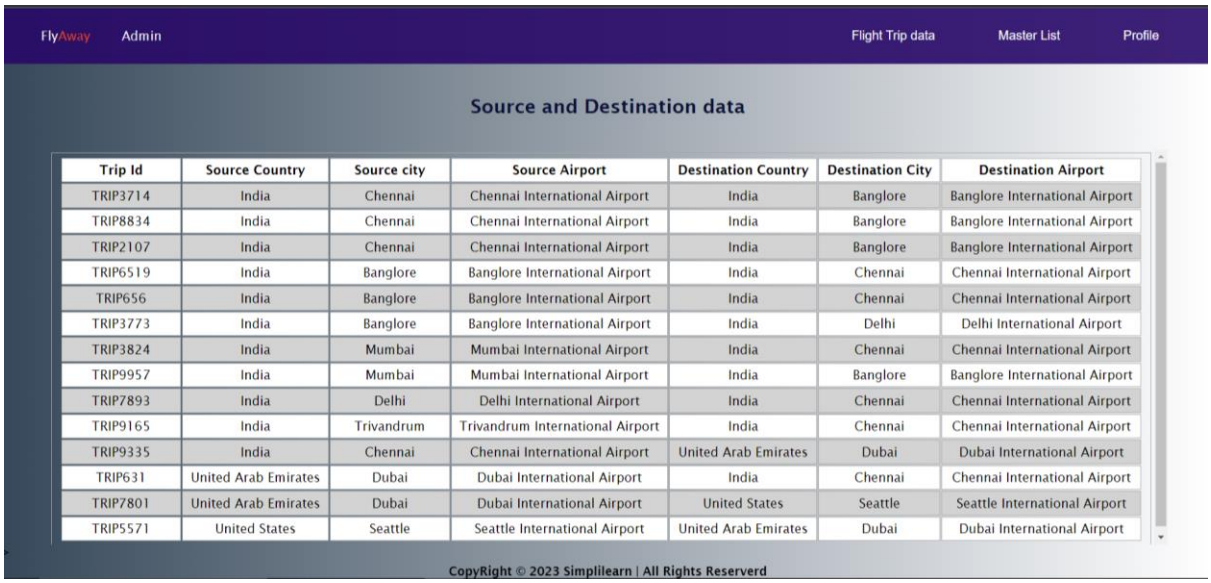
If admin clicks on edit respective trip data will be retrieved and can be edited



Airline Master List



Source and Destination Master List



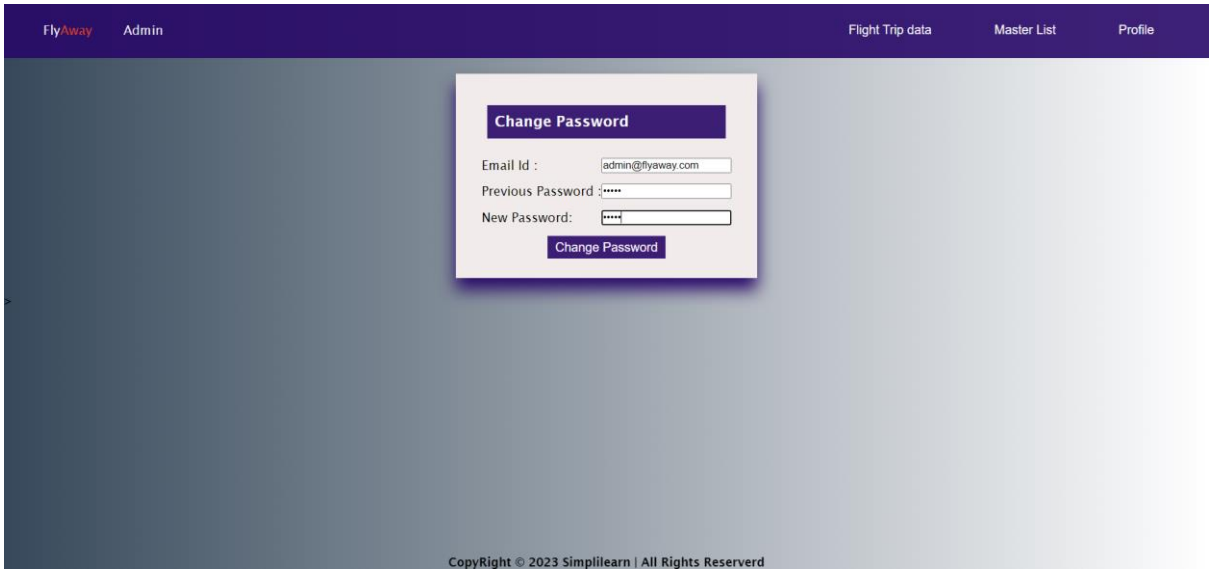
Trip Id	Source Country	Source city	Source Airport	Destination Country	Destination City	Destination Airport
TRIP3714	India	Chennai	Chennai International Airport	India	Banglore	Banglore International Airport
TRIP8834	India	Chennai	Chennai International Airport	India	Banglore	Banglore International Airport
TRIP2107	India	Chennai	Chennai International Airport	India	Banglore	Banglore International Airport
TRIP6519	India	Banglore	Banglore International Airport	India	Chennai	Chennai International Airport
TRIP656	India	Banglore	Banglore International Airport	India	Chennai	Chennai International Airport
TRIP3773	India	Banglore	Banglore International Airport	India	Delhi	Delhi International Airport
TRIP3824	India	Mumbai	Mumbai International Airport	India	Chennai	Chennai International Airport
TRIP9957	India	Mumbai	Mumbai International Airport	India	Banglore	Banglore International Airport
TRIP7893	India	Delhi	Delhi International Airport	India	Chennai	Chennai International Airport
TRIP9165	India	Trivandrum	Trivandrum International Airport	India	Chennai	Chennai International Airport
TRIP9335	India	Chennai	Chennai International Airport	United Arab Emirates	Dubai	Dubai International Airport
TRIP631	United Arab Emirates	Dubai	Dubai International Airport	India	Chennai	Chennai International Airport
TRIP7801	United Arab Emirates	Dubai	Dubai International Airport	United States	Seattle	Seattle International Airport
TRIP5571	United States	Seattle	Seattle International Airport	United Arab Emirates	Dubai	Dubai International Airport

Trip Master List

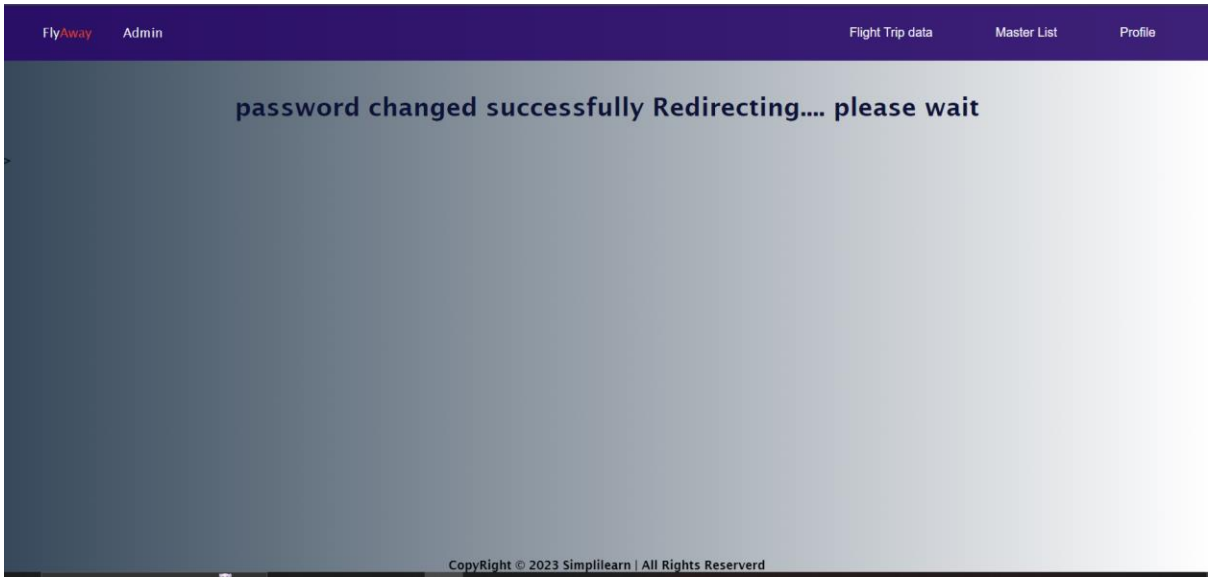
Trip Id	Flight Number	Airline	Source Country Name	Source City Name	Source Airport Name	Departure Date	Departure Time	Destination Country Name	Destination City Name	Destination Airport Name	Arrival Date	Arrival Time	Max Passengers
TRIP3714	IN9295	Indigo	India	Chennai	Chennai International Airport	22-05-2023	07:00	India	Banglore	Banglore International Airport	22-05-2023	08:00	30
TRIP8834	SP9814	Spice Jet	India	Chennai	Chennai International Airport	22-05-2023	07:15	India	Banglore	Banglore International Airport	22-05-2023	08:15	30
TRIP2107	AK377	Akasa Air	India	Chennai	Chennai International Airport	22-05-2023	07:30	India	Banglore	Banglore International Airport	22-05-2023	08:30	30
TRIP6519	IN6588	Indigo	India	Banglore	Banglore International Airport	22-05-2023	08:30	India	Chennai	Chennai International Airport	22-05-2023	09:30	30
TRIP656	SP8058	Spice Jet	India	Banglore	Banglore International Airport	22-05-2023	08:45	India	Chennai	Chennai International Airport	22-05-2023	09:45	30
TRIP3773	IN2816	Indigo	India	Banglore	Banglore International	22-05-2023	08:00	India	Delhi	Delhi International	22-05-2023	11:30	35

Change Password

If the admin changes his password he will be logged out and prompted that he is logged out and redirected to login page within 3 seconds of delay and asked to login again

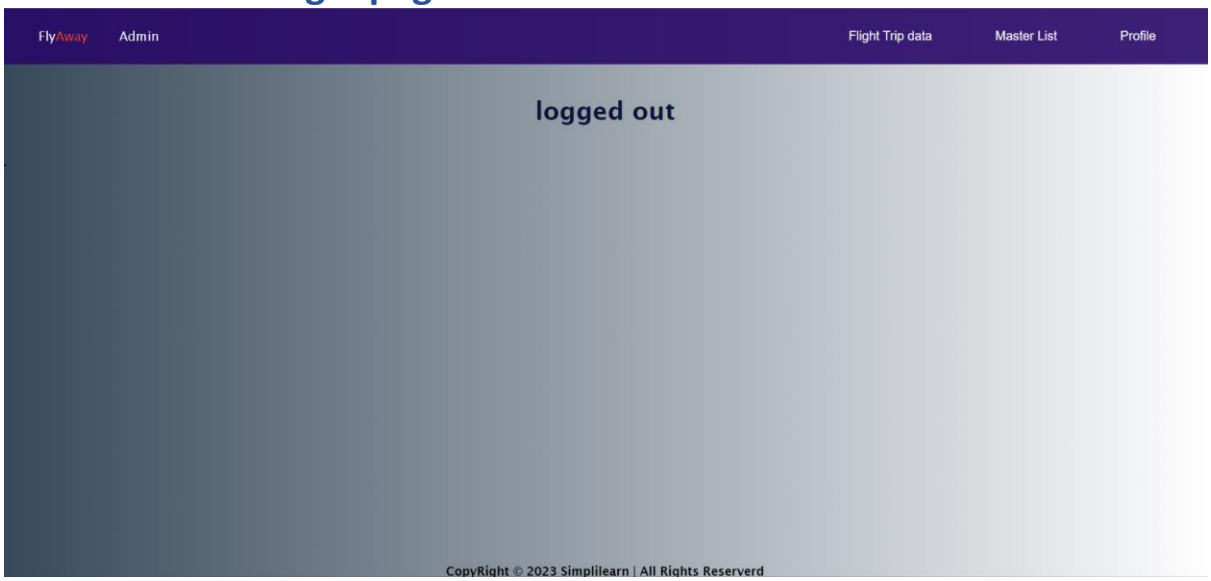


After Changing Password

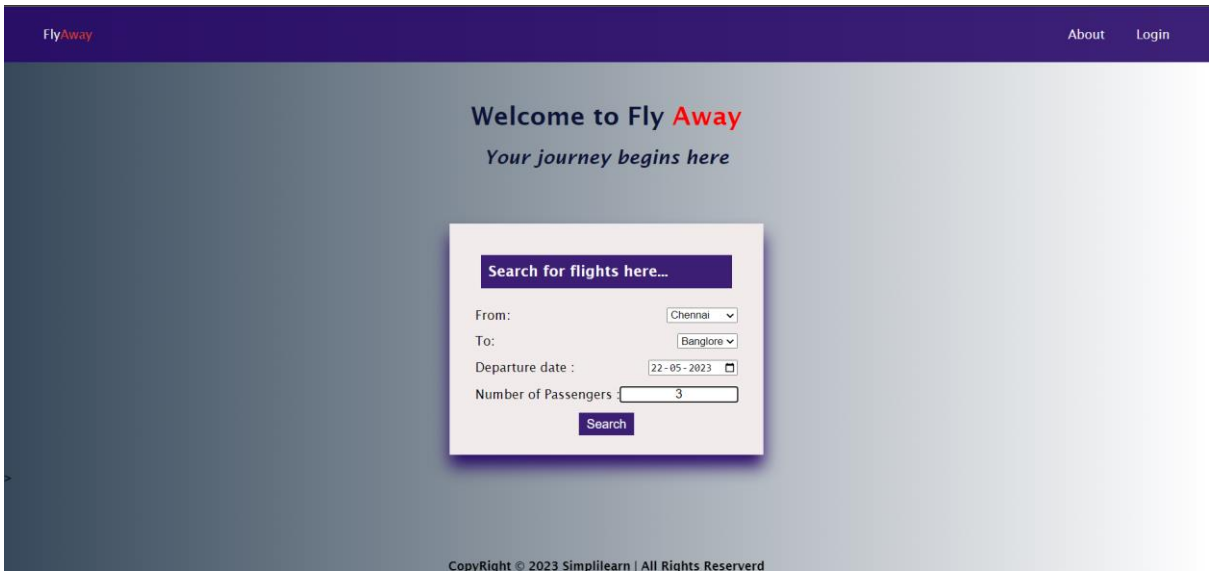


Logout

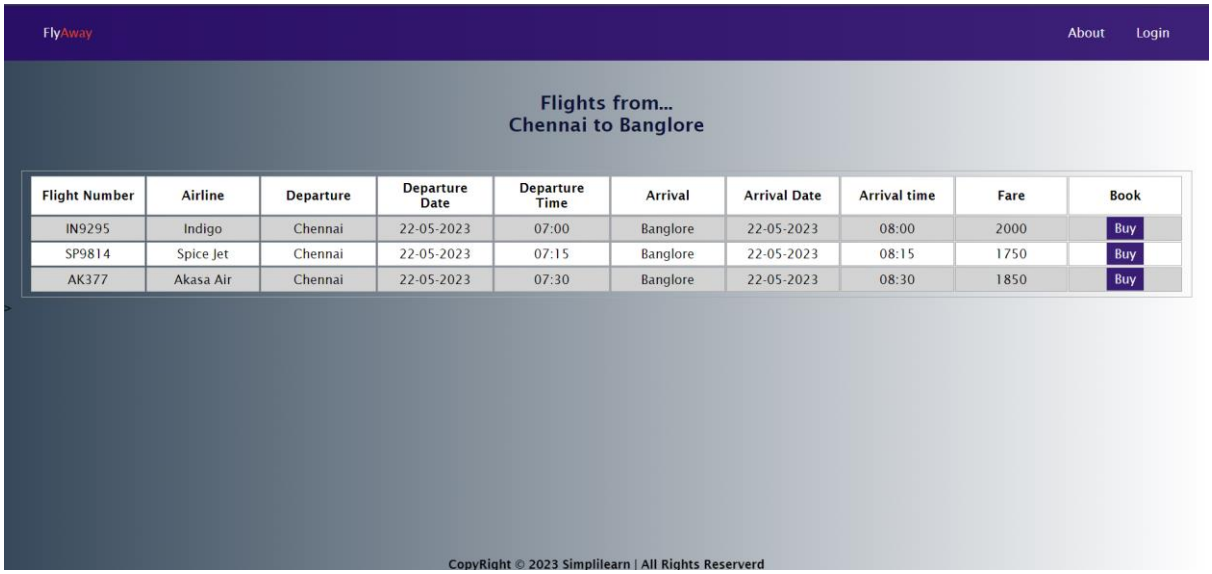
If admin clicks logout he will be prompted and automatically redirected to login page within 3 sec timer



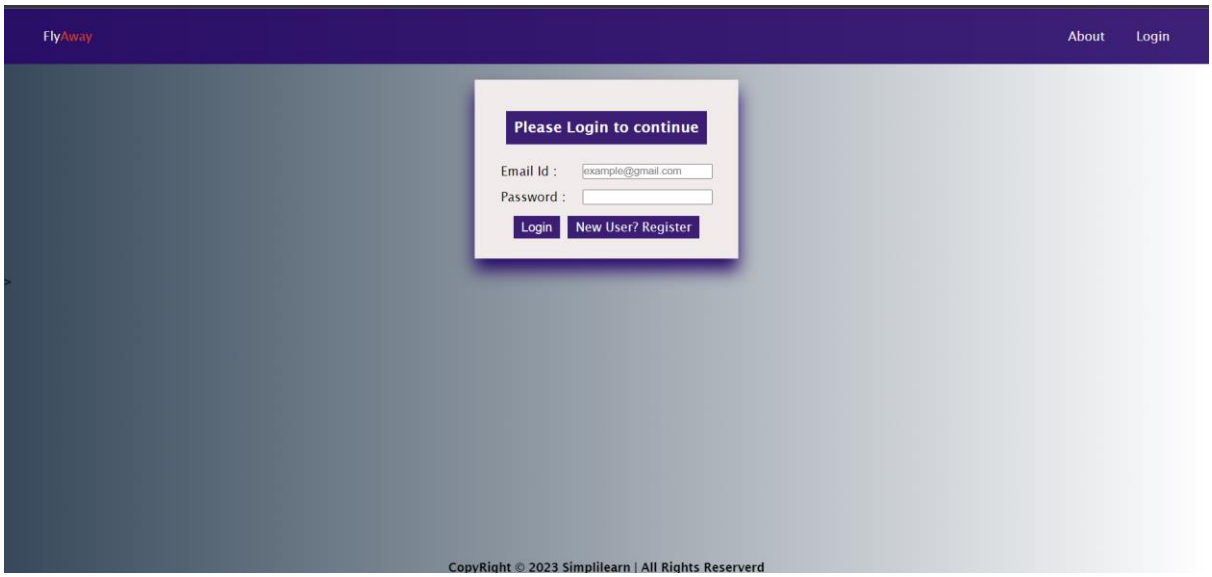
FlyAway Home Page



FlyAway Search Results

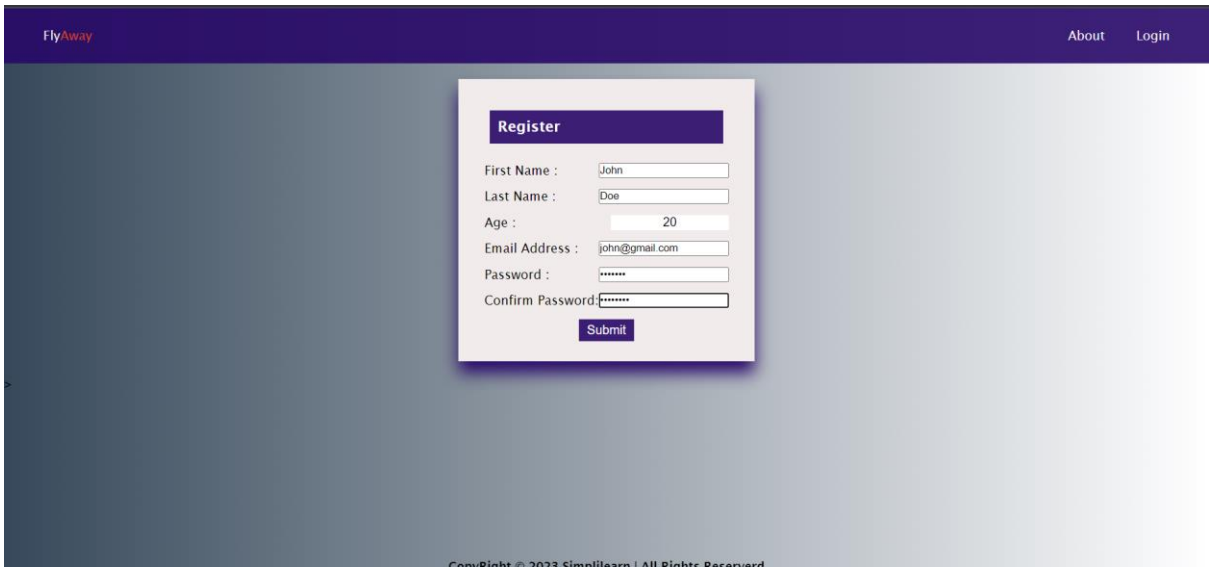


Buying tickets = when user clicks “Buy” he will be automatically asked to either login or register himself. I am selecting Trip 1 where Airline operator is Indigo

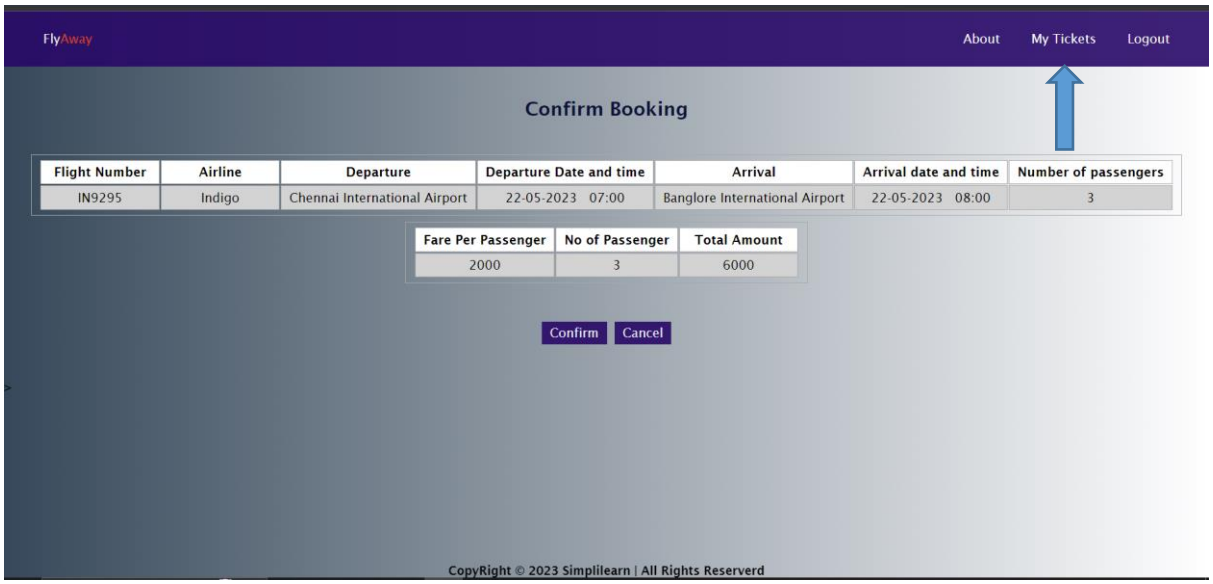


Register

Registering myself as John Doe



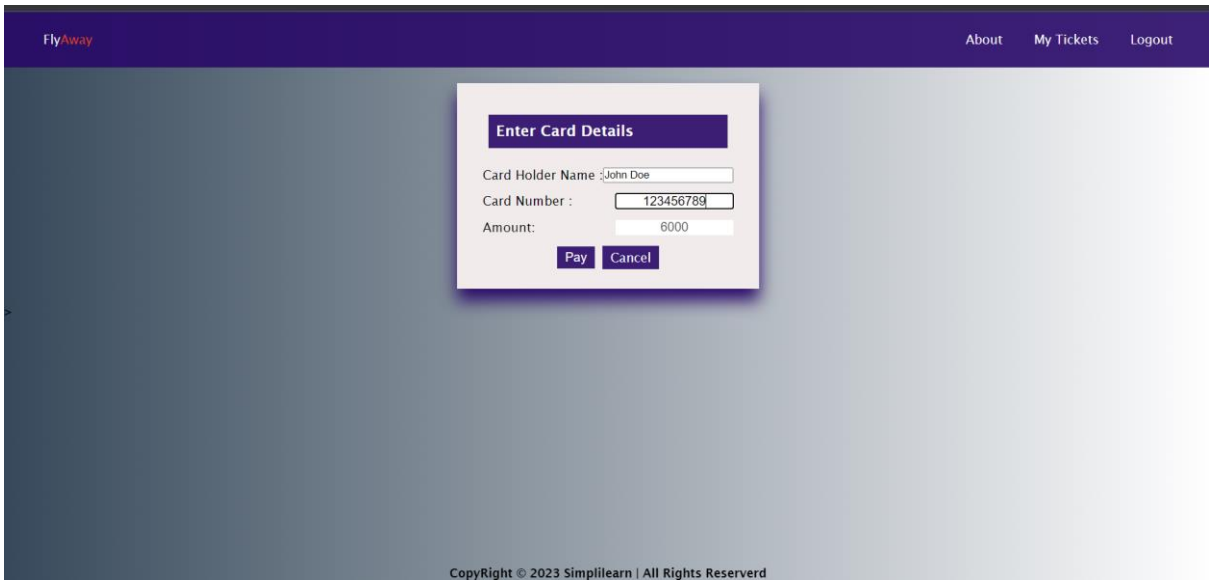
When I click Register I will be logged in with the same credentials and redirected to proceed with my ticket booking. As I am logged in now please note the navigation bar has changed



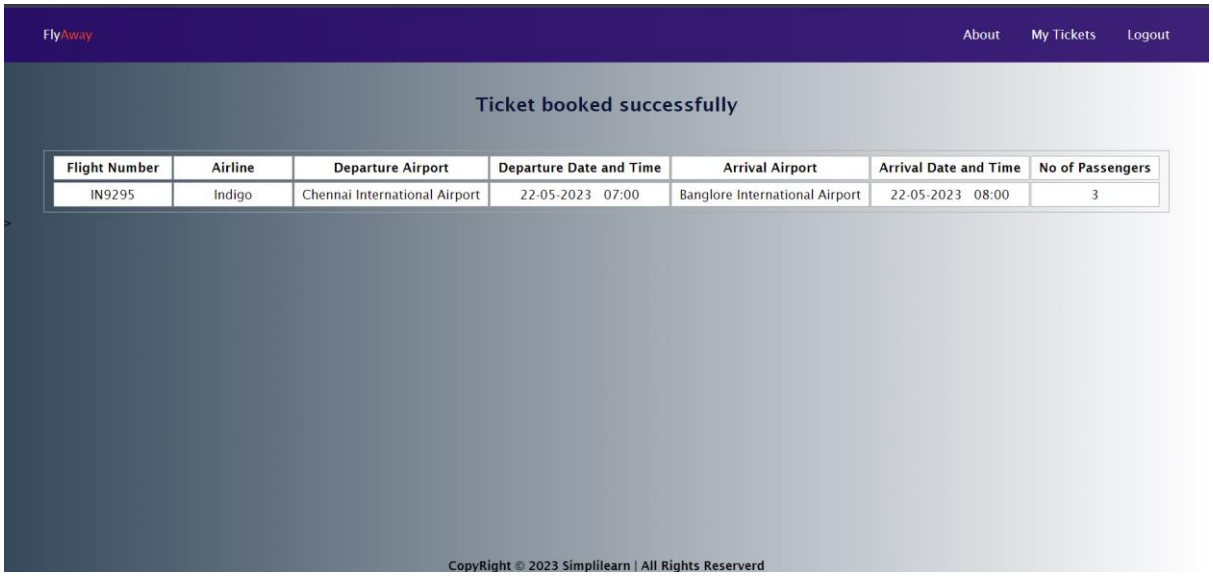
I have chosen no of passengers as 3 in the search menu so I am asked to pay ticket price for all the passengers

Register

Dummy Payment Gateway

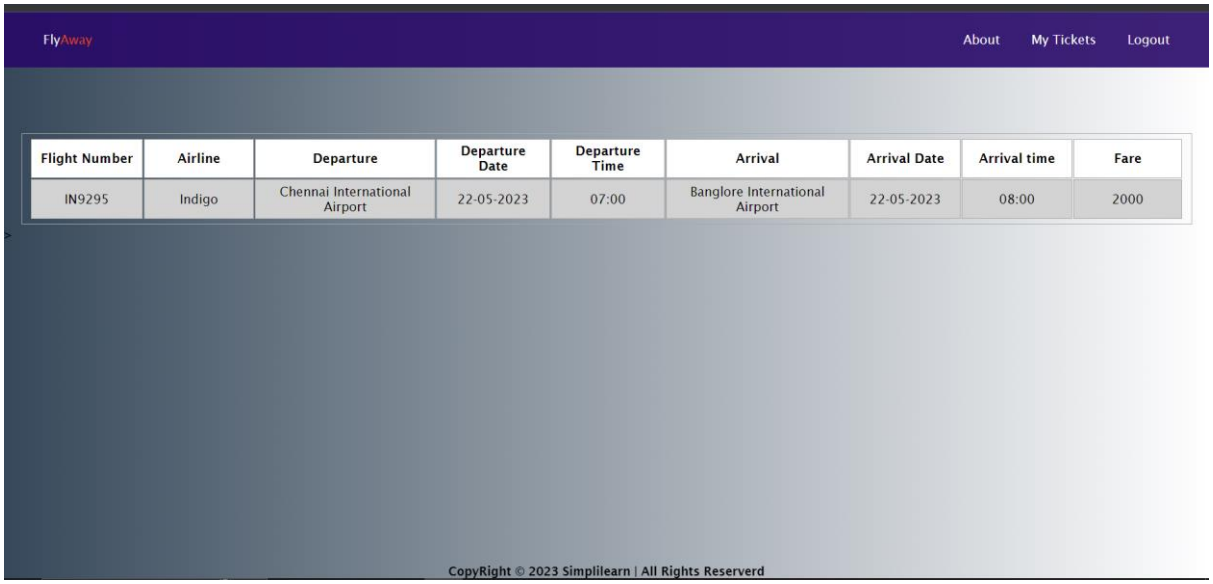


Confirm booking and confirmation status



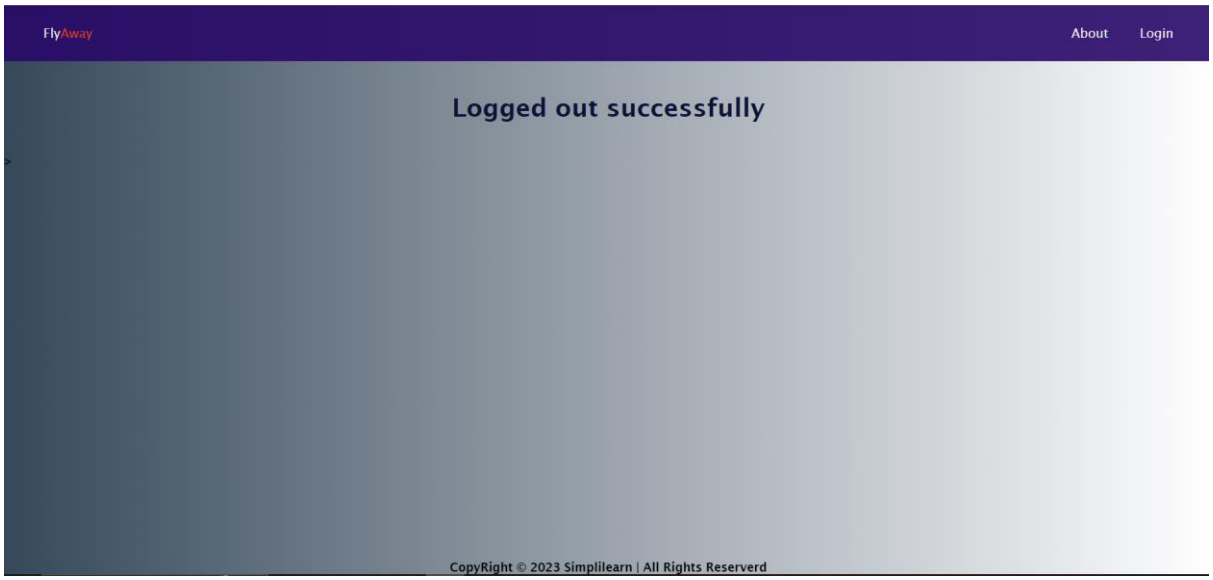
@ManytoMany trip details are also successfully updated in the database

User can also see the ticket details reflected in his My Tickets page



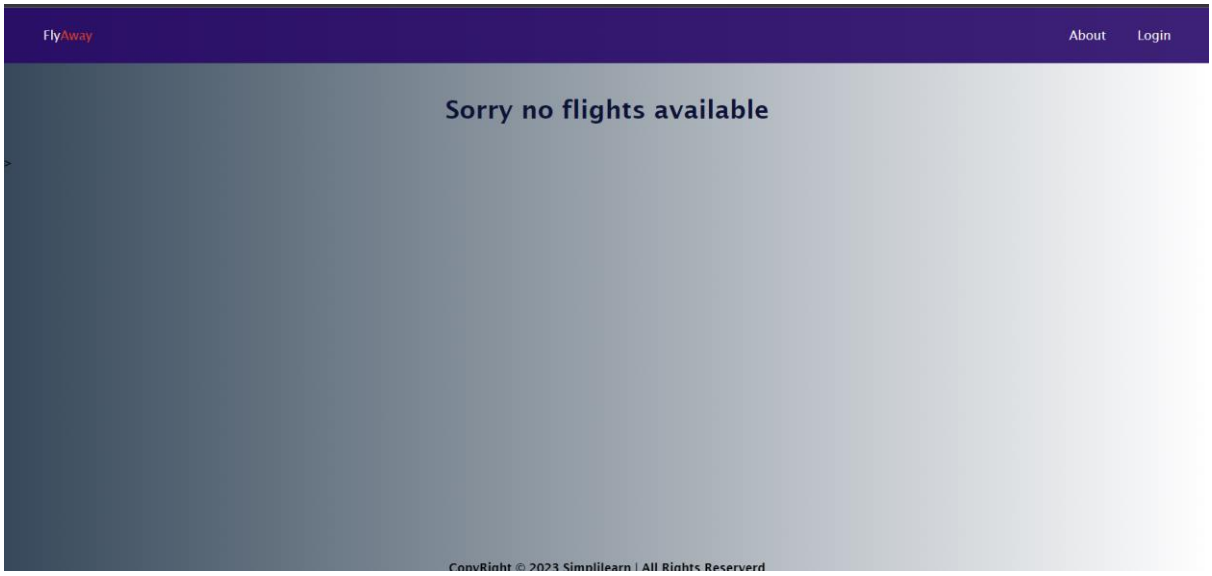
Logout

When user clicks logout he will be logged out and prompted and the navigation bar changes automatically



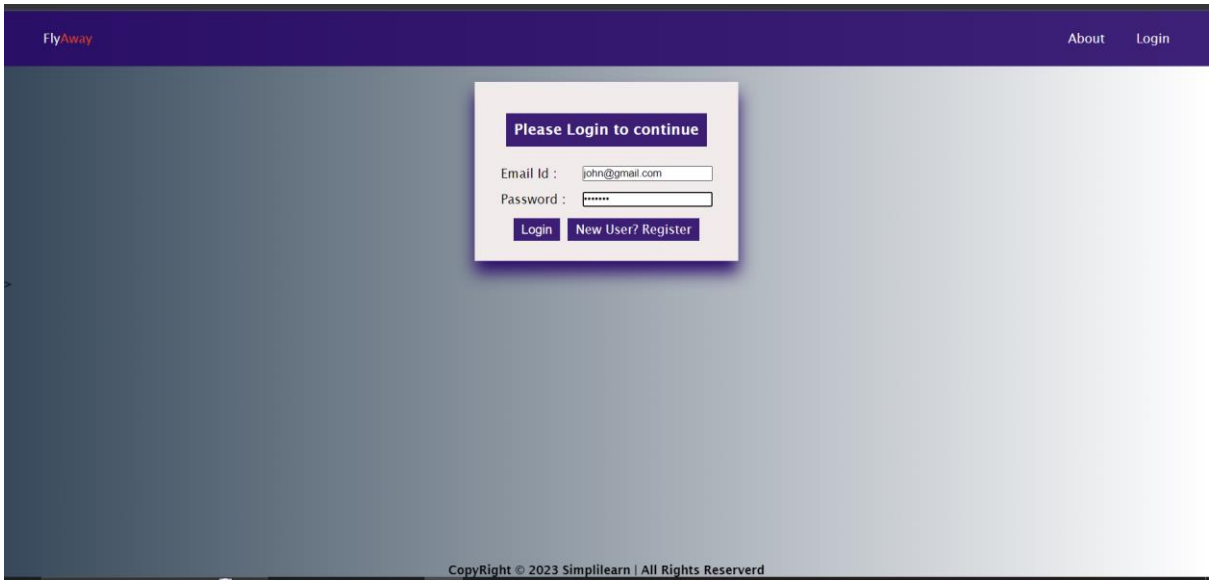
Miscellaneous and Error Pages

If user query flight details are not available

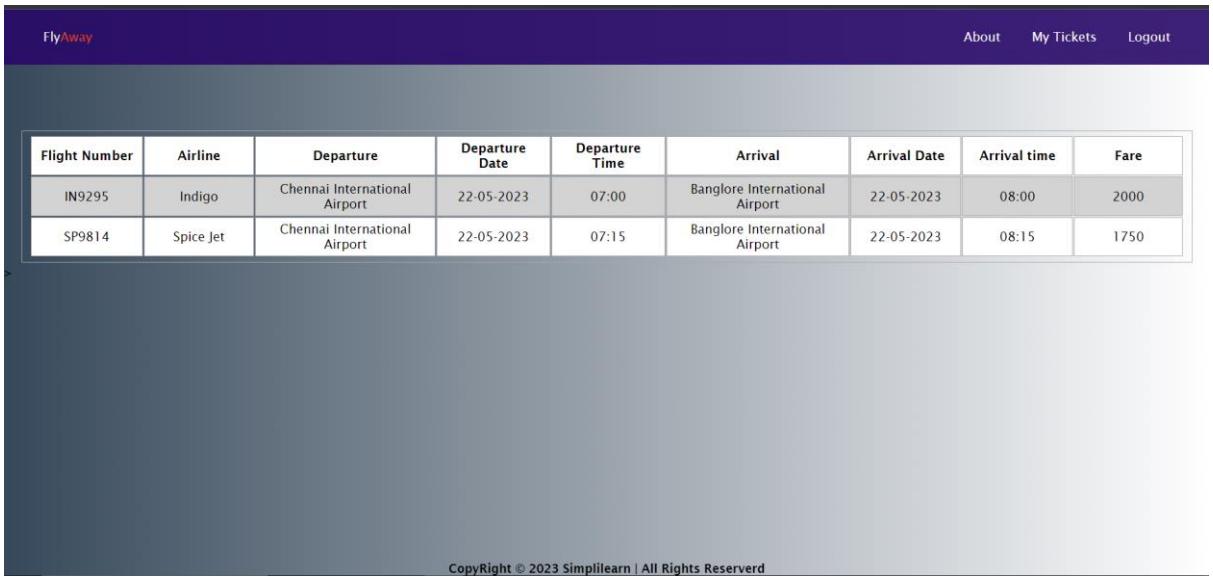


While booking flight user can also log in

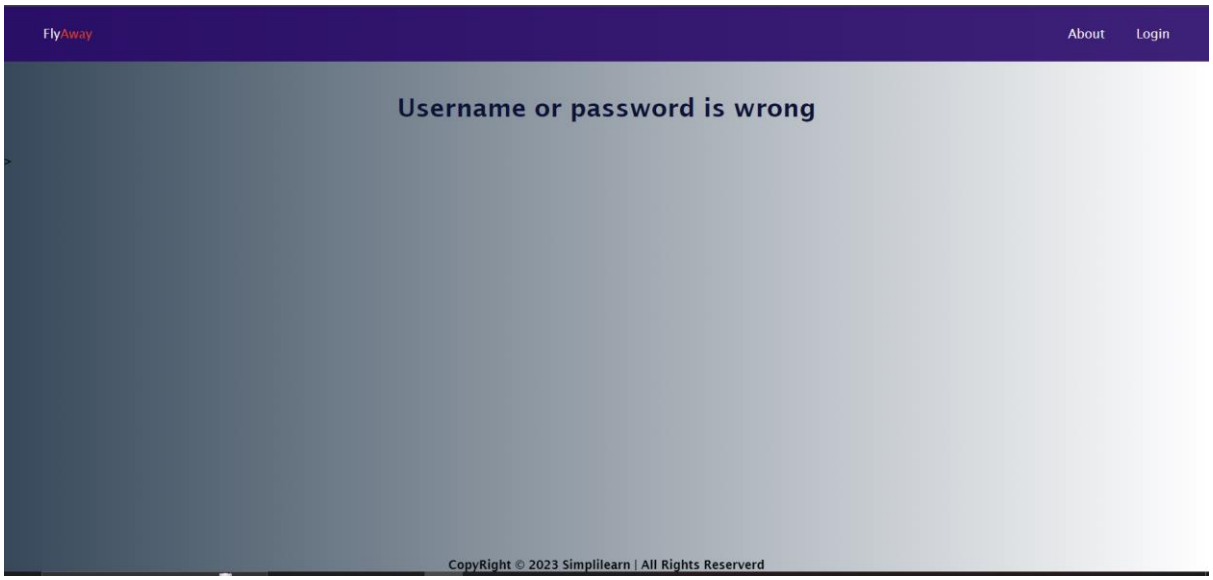
To show this I again logged In as John Doe and booked a new ticket



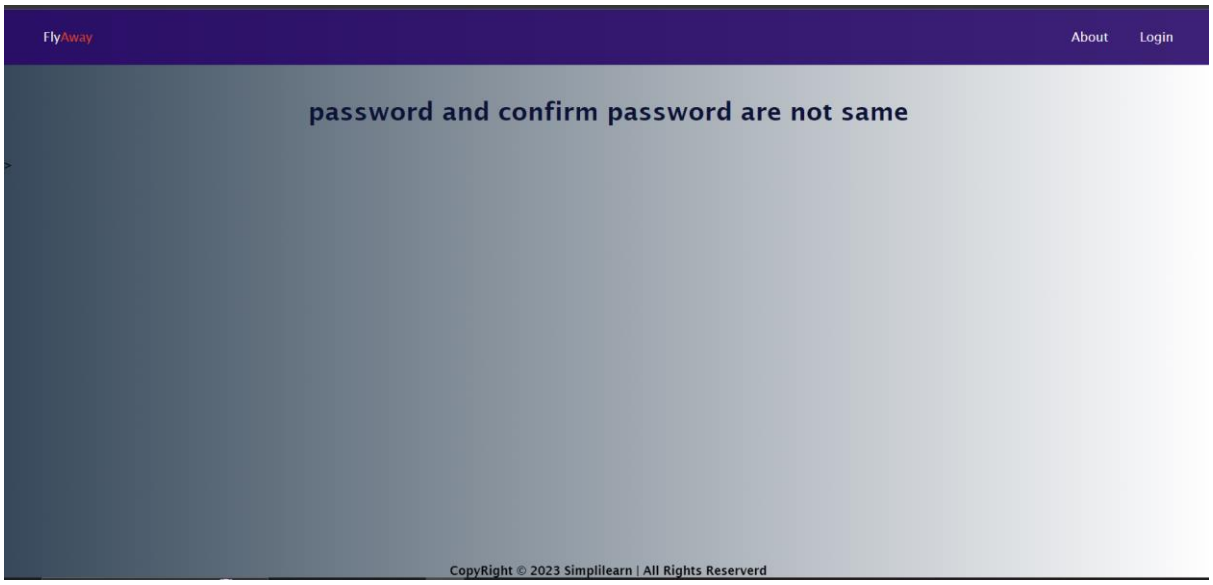
John Doe's Ticket History



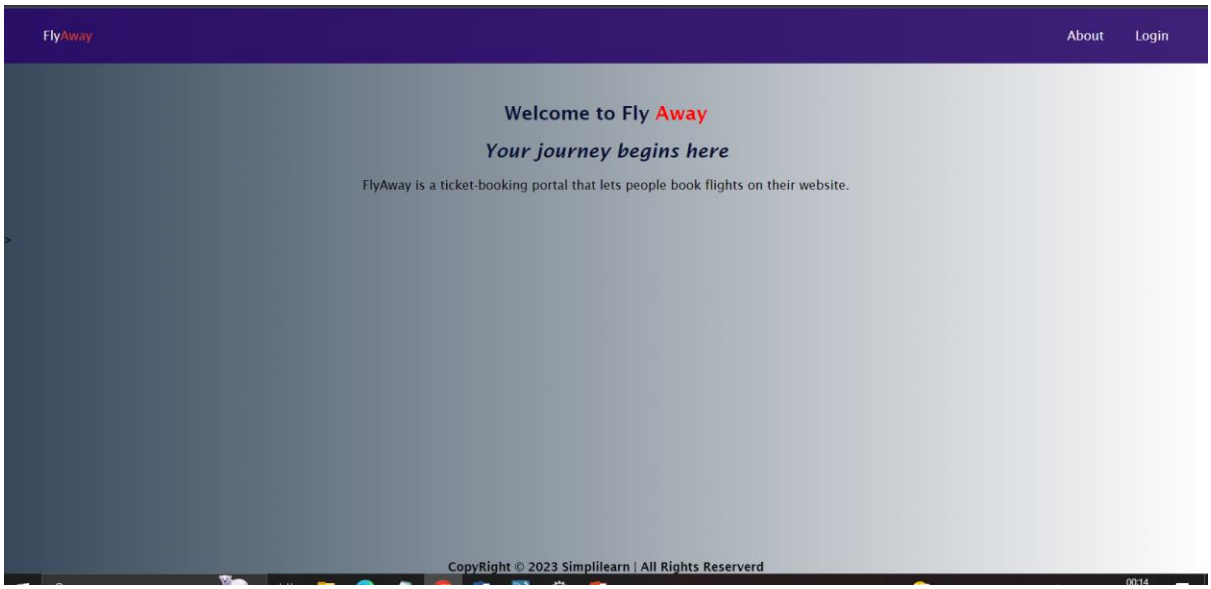
While login in user can enter a wrong password he will be prompted the same



While registering user can enter password and confirm password different

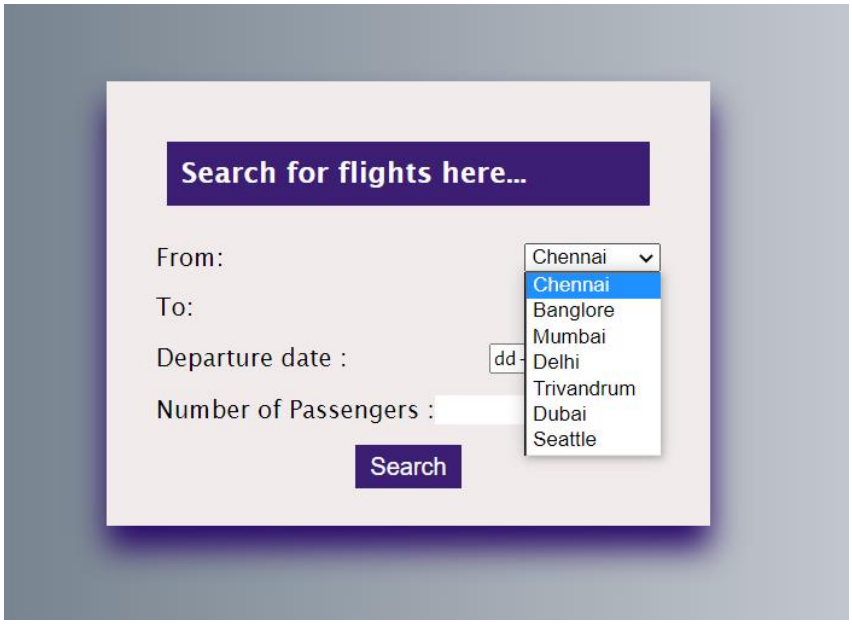


About Page



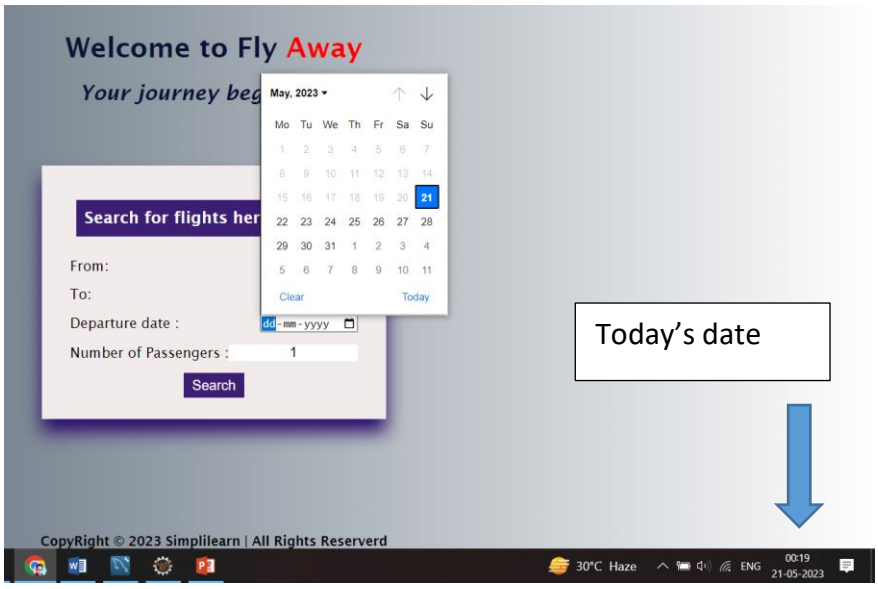
Search Query

When Admin adds Trip data in the backend distinct cities are displayed respectively



Form Date field:

User and Admin can only use present date, past dates are greyed out



Search Operations in Admin field