**<Book My Flight>**

**<Air-Ticket Reservation System>**

**Application Design**

**<Version No.0 01>**



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By** | **Reviewed by** | **Approved By** |
| **Name** |  |  |  |
| **Role** |  | Project Leader | Project Manager |
| **Signature** |  |  |  |
| **Date** |  |  |  |

Table of Contents

[1.0 Introduction 3](#_Toc157835546)

[1.1 Purpose of this document 3](#_Toc157835547)

[1.2 Scope 3](#_Toc157835548)

[1.3 Intended Audience 3](#_Toc157835549)

[1.4 Definition & Acronyms 3](#_Toc157835550)

[1.5 References 3](#_Toc157835551)

[2.0 Conventions and Standards Followed 4](#_Toc157835552)

[3.0 Assumptions, Dependencies and Risks 5](#_Toc157835553)

[4.0 Use Case Realization 6](#_Toc157835554)

[4.1 Flow of Events – Design 6](#_Toc157835555)

[4.2 Derived Requirements 7](#_Toc157835556)

[5.0 Package & Sub-system Design 8](#_Toc157835557)

[5.1 Package Design 8](#_Toc157835558)

[5.1.1 Packages 1…N 8](#_Toc157835559)

[5.2 Sub-system Design 9](#_Toc157835560)

[6.0 Generic Components Design 10](#_Toc157835561)

[*7.0* Object Model 11](#_Toc157835562)

[7.1 System Object Model 11](#_Toc157835563)

[7.2 Object Descriptions 11](#_Toc157835564)

[7.2.1 Object <Name> 11](#_Toc157835565)

[8.0 Database Design 12](#_Toc157835566)

[8.1 Data Model 12](#_Toc157835567)

[8.1.1 Identification of Tables 12](#_Toc157835568)

[8.1.2 Table Details 12](#_Toc157835569)

[8.1.3 Referential Integrity Constraints 12](#_Toc157835570)

[8.1.4 Computations/processing in Database 12](#_Toc157835571)

[8.1.5 Access Control Details 12](#_Toc157835572)

[8.2 Storage Characteristics 13](#_Toc157835573)

[8.3 Database Performance Improvement Measures 13](#_Toc157835574)

[9.0 References 13](#_Toc157835575)

[10.0 Change Log 14](#_Toc157835576)

# Introduction

## Purpose of this document

The detailed design contains the design of every module / sub-system. This covers the following:

* Design of common / generic components
* Module / Sub-system design
* Unit / Class / Program Specifications

From a content consideration, the above three have been combined in this document. However depending on the scale of design or projects, from manageability and configurability point of view, the above three may be split and maintained

## Scope

* User login/register
* Content validation
* Generate SSN type
* Maintain flight information.
* Provision for searching available flights
* Provision for viewing available seats
* Provision for searching and booking the flight ticket
* Provision for generating booking id
* Maintaining booking information
* Update and maintain the customer information
* Create and maintain the registered user information.
* Create and update the flight information.
* Provision for cancelling the tickets.
* Calculate the calculation and refundable amount.
* Update the availability of seats.

## Intended Audience

The intended audience for this document is

* Developers, testers, other associates in the project
* Client
* Delivery Assurance Group, Process Engineering group

## Definition & Acronyms

|  |  |
| --- | --- |
| **Acronyms** | **Definitions** |
| SSN | Social Security Number |

## References

# Conventions and Standards Followed



# Assumptions, Dependencies and Risks

**Assumptions:**

* Once the application is built the system will manually load the current user list into the system.
* There will be no concurrent users for the application. The user profile will be maintained by the Admin.
* Any changes to the design mentioned in the BRD would be taken as a Change Request.
* A request for booking/cancellation of flight from any source to any destination, giving connected flights in case no direct flight between the specified Source-Destination pair exists.

**Dependencies:**

N/A

**Risks:**

* Application should have single login. Login should be the first interface that the user would encounter.
* Only registered users can book tickets.
* Same email id and SSN number cannot be used.
* Only registered customers can access the application.
* None of the application features can be accessed without login.
* There should be logout feature. Once logged out should come to the first interface of login again.
* At any point of time user should be having the option to navigate back to the home interface which is the welcome interface after login.
* Multiple user should be able to access the application at the same time & every user is allocated Quota time to access.
* Once logged out the user will not be able to back to any interface.
* On login user credentials should be validated against the registered users in the system.
* Log in page should have username, Password fields to receive username and password and should have login and reset button.
* Password must be encrypted.

# Use Case Realization

*<The background or the reason that led to the initiation of the Business Context study to be mentioned here. >*

## Flow of Events – Design

*<Description of how the use case is realized in terms of collaborating objects. Its main purpose is to summarize the diagrams connected to the use case and to explain how they are related through SEQUENCE DIAGRAMS and COLLABORATION DIAGRAMS >*

**Sequence Diagrams**

*<Interactions could be represented in the form of sequence diagrams. >*



*<<Insert sequence diagrams here. Please note that you could insert the diagrams here directly OR provide reference to the file, which contains this.*

*The actual insertion into the document can be deferred at the time of delivery of the document to the client.*

*But is necessary to specify the version number of every such document referenced here and every change in the referenced document should be tracked as part of the Change Log to the Design Document>>*

## Derived Requirements

*<Description that collects all requirements, such as non-functional requirements, on the use-case realizations that are not considered in the design model, but that need to be taken care of during implementation>*

### User Registration

### 



### Search Flight



### Booking Flight



### Cancellation



### Edit Profile

Login

User Option

Edit / Update

Get Email, Password and validation

Edit field

Data update

Set Email, Password

# Package & Sub-system Design

*<The purpose of this section is to explain the design over view of package or the sub system. The Design Model can be split into smaller units to make it easier to understand. By grouping Design Model elements into packages and subsystems, then showing how those groupings relate to one another, it is easier to understand the overall structure of the model>*

## Package Design

*<Packages are general-purpose hierarchical organizational units (units as in classes, relationships etc.). Such units are not contained in some other element. Elements such as attributes, operations, states, lifelines, and other messages are contained in other elements and do not appear as direct contents of packages.*

*A package may contain other packages. There is a root package that indirectly contains the entire model of a system. There are several ways to organize packages: - By view, BY functionality, or other relevant basis.*

* *If the packages are well chosen they can reflect the high-level architecture of the system.*
* *Identification of packages so that teams can work without interfering with each other.*
* *Delineating source code into logical partitions to enhance maintainability*
* *Implementing effective Configuration Management*
* *Access Control*

Constructing libraries etc.>

### Packages 1…N

*<Packages may be depicted here>*

|  |  |
| --- | --- |
| Attribute Name | Brief Description of Attribute |
| Name | The name of the package. |
| Brief Description | A brief description of the role and purpose, or the "theme" of the package. |
| Classes | The classes directly contained in the package. |
| Relationships | The relationships directly contained in the package. |
| Use-Case Realizations | The use-case realizations directly contained in the package. |
| Diagrams | The diagrams directly contained in the package. |
| Design Packages | The packages directly contained in the package. |
| Import Dependencies | The import dependencies from the package to other packages. |

## Sub-system Design

*<A subsystem is a physical unit of implementation with well-define interfaces that is intended to be used as replaceable part of the system. Each sub system embodies the implementation of certain classes. Well-defined subsystems do not depend directly on other sub systems but on interfaces that sub system support.*

*Note: If this already addressed as part of architecture, this may be skipped here and referenced here>*

|  |  |
| --- | --- |
| Property Name | Brief Description |
| Name | The name of the subsystem |
| Brief Description | The short description of the role and purpose, or the "theme" of the subsystem. |
| Interfaces | Associations to realized interfaces |
| Contents | Aggregation associations to contained model elements |
| Dependencies | Dependency associations to interfaces, packages, or subsystems on which the subsystem depends |
| Diagrams | Any diagrams local to the subsystem, such as class diagrams or statechart diagrams. |

# Generic Components Design

*<Generic design may include the following:*

* *Framework classes*
* *Audit Trial Classes*
* *Authority Checks*
* *Alarms/Error/Notification Requirements*
* *Operational Sequence Check*
* *Device Check*
* *Electronic Signatures*
* *Process Control Parameters*
* *Authorization*
* *Record Access*
* *Record retention & recovery requirements>*

*A component is a physical unit of implementation with well-defined interfaces that is intended to be used as replaceable part of the system. Each component embodies the implementation of certain classes. Well-defined components do not depend directly on other components but on interfaces that components support>*

# Object Model

## System Object Model

**Registration**



**searching**



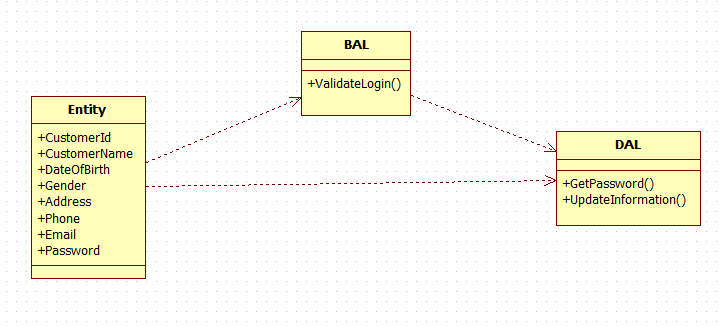
**Booking**



**Cancellation**



**Edit Profile**



## Object Descriptions

### Login

### Functional Requirement

Login form checks the authentication of the user and directs the user to the respective form. User will be provided with the credentials like email and password which checks for the email and password in the database to authenticate it.

### UI Controls

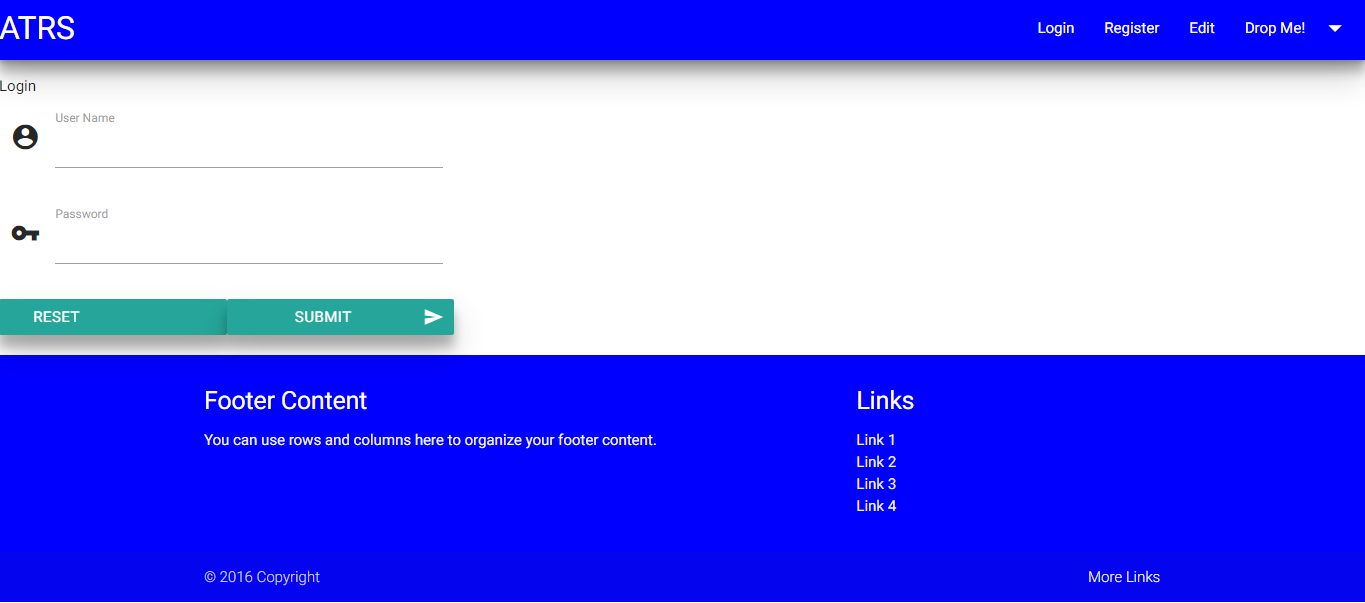
#### Login

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| Email | Textbox | No | Yes | No | Mandatory | Varchar | 30 |
| Password | Textbox | No | Yes | No | Mandatory | Varchar | 30 |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| btnLogin\_Click | loginButton | Allows the user to redirect to the Home screen |
| btnRegister\_Click | registerButton | Allows the user to redirect to the  Registration screen |
| btnReset\_Click | resetButton | All details entered by the user is reset |
| btnExit\_Click | exitButton | Used to exit the application |

### UI Design



### Registration

### Functional Requirement

Registration form should allow users to add name, email, password, dob, address, phone, gender, SSN type, SSN number. All fields should be entered by the user without fail. Email id and SSN number should be unique. Validation should be done for every field.

### UI Validations

* Application should have single login. Login should be the first interface that the user would encounter.
* Only registered customers can access the application.
* None of the application features can be accessed without login.
* On login user credentials should be validated against the registered users in the system.
* Log in page should have Email, Password fields to receive email and password and should have login and reset button.
* Password must be encrypted.

### UI Controls

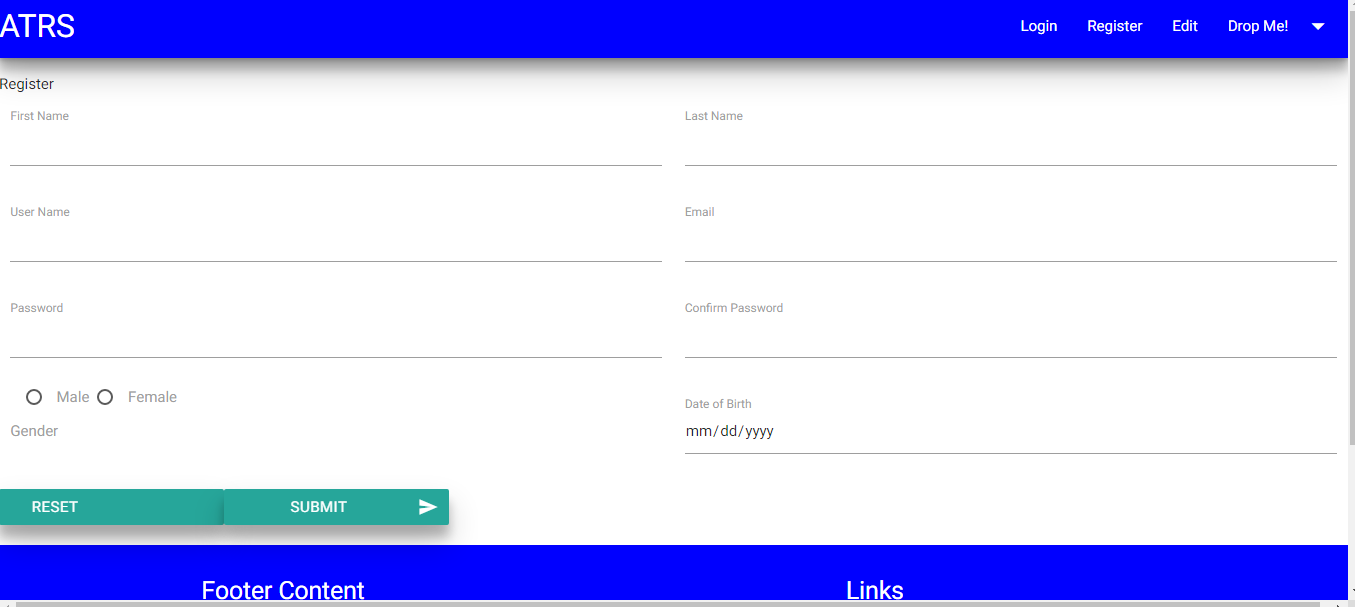
**Registration**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 30 |
| Email | Textbox | No | Yes | No | Mandatory | Varchar | 30 |
| Password | Textbox | No | Yes | No | Mandatory | Varchar | 30 |
| Dob | Date Picker | No | Yes | Date | Mandatory | - | - |
| Address | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 50 |
| Phone | Textbox | No | Yes | Only Numbers | Mandatory | Ulong | - |
| Gender | Radio button | No | Yes | Only Alphabets | Mandatory | Varchar | 6 |
| SsnType | Combo Box | No | Yes | Only Alphabets | Mandatory | Varchar | 30 |
| SsnNumber | Textbox | No | Yes | Alphanumeric | Mandatory | Varchar | 30 |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| btnSubmit\_Click | submitButton | Once all details are entered it redirects to the Login screen. |
| btnReset\_Click | resetButton | All details entered by the user are reset. |
| btnBack\_Click | backButton | Redirect to the login form. |

### UI Design



### UserFetchDetails

### Functional Requirement

This class is used to handle business rules regarding login and registration details provided by the user for air ticket reservation application. It acts as the medium between the presentation layer and data access layer.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| GetLoginDetails | UserDetails  userDetails | Gets the login details from presentation layer. |
| GetUserDetails | UserDetails  userDetails | Gets the registration details from the presentation layer. |
| StoreBoxDetails | None | Stores the SSN type details into the presentation layer. |
| FetchValidate | UserDetails  userDetails | Gets the details from the data access layer and validates. |
| HashPassword | Password | Used to hash the password. |

### UserStoreDetails

### Functional Requirement

This Class is used to store the login and registration details into the database fetched from the user.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| ConnectLoginDetails | UserDetails  userDetails | Checks the login details entered by the user with the details in the database. |
| ConnectUserDetails | UserDetails  userDetails | Stores the registration details into the database fetched from the user. |
| GetBoxDetails | None | Gets the SSN type details from the database. |
| ValidateEmail | UserDetails  userDetails | Gets the email from the database that equals to the user entered email. |
| ValidateSsnNumber | UserDetails  userDetails | Gets the SSN number from the database that equals to the user entered SSN number. |

### Home

### Functional Requirement

Home screen will allow the user to search for available flights, book tickets, edit profile and cancel tickets.

### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Welcome To Book My Flight | Label | welcomeLabel |  |
| 2 | Search Flights | Button | searchButton | Allows the user to redirect to the SearchFlight screen |
| 3 | Book Ticket | Button | bookButton | Allows the user to redirect to the SearchFlights screen |
| 4 | Edit Profile | Button | editButton | Allows the user to redirect to the Edit Profile screen |
| 5 | Ticket Cancellation | Button | cancelButton | Allows the user to redirect to the Cancellation screen |
| 6 | Logout | Button | logoutButton | Allows the user to redirect to the Login screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| Home\_Load | Form | Loads the Home screen |
| searchButton\_Click | searchButton | Allows the user to redirect to the SearchFlight screen |
| editButton\_Click | editButton | Allows the user to redirect to the  EditProfile screen |
| cancelButton\_Click | cancelButton | Allows the user to redirect to the  Cancellation screen |
| bookButton\_Click | bookButton | Allows the user to redirect to the SearchFlights screen |
| logoutButton\_Click | logoutButton | Allows the user to redirect to the Login screen |

### UI Design



### SearchFlight

### Functional Requirement

SearchFlight screen will allow the user to provide required search fields such as source location, destination location and date of journey.Based on the user information the search results will be displayed.

### UI Validations

On click of “Show Flights” button, check for the following criteria

* Check if Leaving From, Going To, Date Of Journey fields are entered by the user.If not, display appropriate message.
* Leaving From and Going To location cannot be same
* Date of Journey should be future date only.
* Date of Journey should be in the format dd-mm-yyyy

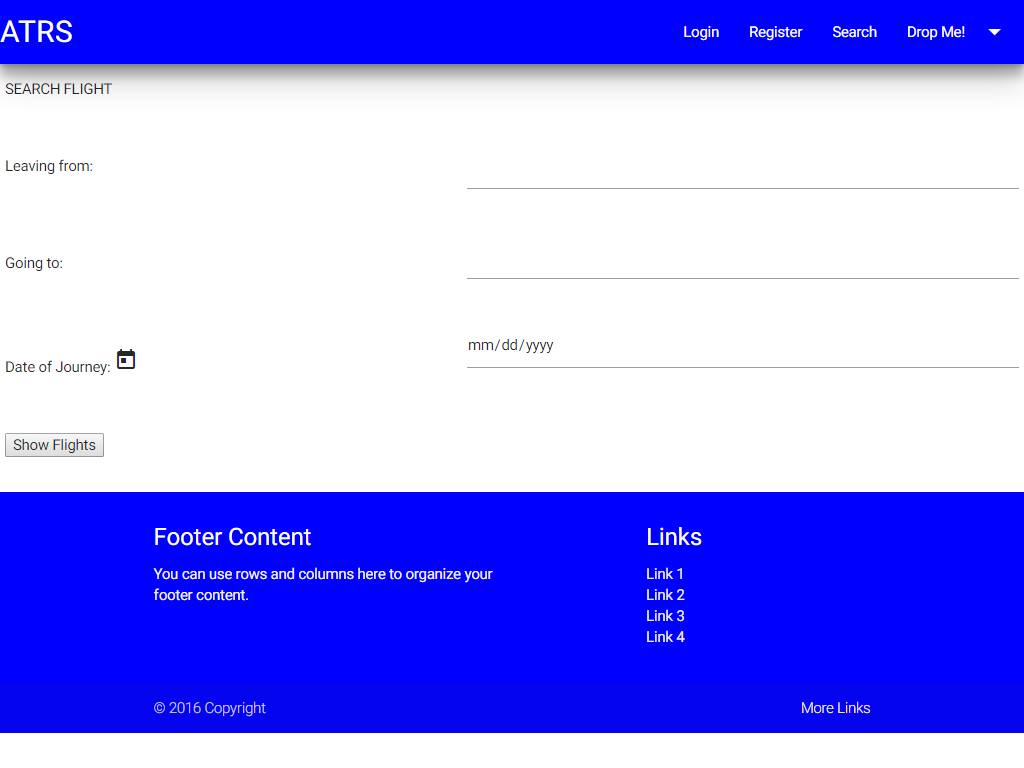
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Leaving From | Label | leavingFromLabel |  |
| 2 | Leaving From | Combobox | leavingFromCombobox | Gets the source location of the flight from the user |
| 3 | Going To | Label | goingToLabel |  |
| 4 | Going To | Combobox | goingToCombobox | Gets the destination location of the flight from the user |
| 5 | Date Of Journey | Label | dateOfJourneyLabel |  |
| 6 | Date Of Journey | Textbox | dateOfJourneyTextbox | Gets the date of journey from the user |
| 7 | Show Flights | Button | showFlightsButton | Allows the user to redirect to the ShowResult screen |
| 8 | Home | Button | homeButton | Allows the user to redirect to the home screen |

### UI EventsDefinition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| SearchFlight\_Load | Form | Loads the SearchFlight screen  Populates the source location from the database  Populates the destination location from the database |
| showFlightsButton\_Click | showFlightsButton | Allows the user to redirect to the ShowResult screen |

### UI Design



### ShowResult

### Functional Requirements

ShowResult screen will display the appropriate search results based on the user information.

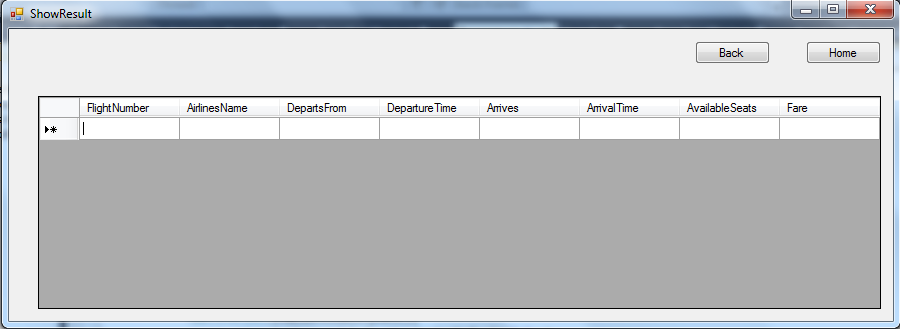
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Gridview | Gridview | searchResultGridview | Displays the search results |
| 2 | Back | Button | backButton | Allows the user to redirect to the SearchFlight screen |
| 3 | Home | Button | homeButton | Allows the user to redirect to the Home screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Event** | **Control/Component** | **Event Details** |
| ShowResult\_Load | form | Loads the search results based on the user information |
| backButton\_Click | backButton | Allows the user to redirect to the SearchFlight screen |
| homeButton\_Click | homeButton | Allows the user to redirect to the Home screen |

### UI Design



### FlightDetails

### Functional Requirement

This class is used to handle business rules regarding flight details provided by the user for air ticket reservation application. Also it passes the flight details to the database layer to perform search operation.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| GetFlightDetails | SearchDetails  searchDetails | Gets the search results from data access layer. |
| ValidateSearchDetails | SearchDetails  searchDetails | Validate the search details provided by the user based on the business rules |
| GetLocations | None | Gets the location names from the data access layer |

### FlightDetailsConnection

### Functional Requirement

This Class is used to fetch the search results from the database based on the user information.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| FetchFlightDetails | SearchDetails  searchDetails | Gets the search results from the database. |
| RetrieveLocations | None | Gets the location names from the database |

### Search Flights

### Functional Requirement

The search flights screen allow s the user to user to enter the search criteria for searching the flights. The search criteria include source and destination locations, date of journey and number of passengers. The user then searches for flights.

### UI Validations

On click of “Book” button, check for the following criteria

1. Check if leaving from, going to, date of journey and number of passengers fields are entered by the user. If not, display appropriate message.
2. Leaving from and going to locations cannot be same
3. Date of journey should be future date only
4. Date of journey should be in the format dd-mm-yyyy
5. Check if the number of passengers is in the range of 1-20. If not, display appropriate message.

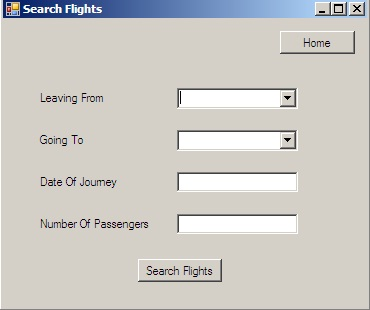
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Leaving From | Label | leavingFromLabel |  |
| 2 | Leaving From | Combobox | leavingFromCombobox | Gets the source location of the flight from the user |
| 3 | Going To | Label | goingToLabel |  |
| 4 | Going To | Combobox | goingToCombobox | Gets the destination location of the flight from the user |
| 5 | Date Of Journey | Label | dateOfJourneyLabel |  |
| 6 | Date Of Journey | Textbox | dateOfJourneyTextbox | Gets the date of journey from the user |
| 7 | Number Of Passengers | Label | numberOfPassengersLabel |  |
| 8 | Number Of Passengers | Textbox | numberOfPassengersTextBox | Gets the number of passengers from the user |
| 7 | Search Flights | Button | searchFlightsButton | Allows the user to go to the search results screen |
| 8 | Home | Button | homeButton | Allows the user to go to the home screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| SearchFlights\_Load | Form | Loads the search flights screen  Populates the source and destination locations |
| searchFlightsButton\_Click | searchFlightsButton | Allows the user to go to the search results screen |
| homeButton\_Click | homeButton | Allow the user to go to the home screen |

### UI Design



### Search Results

### Functional Requirement

The search results screen displays the search results to the user based on the search criteria. The user then selects the flight to be booked and proceeds to book.

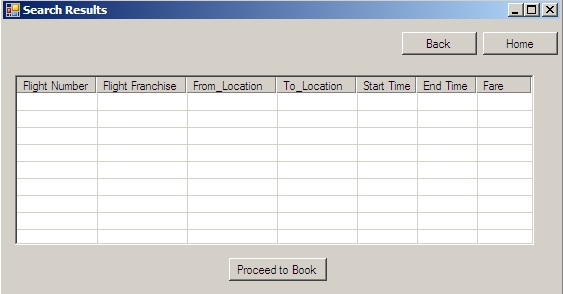
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | ListView | ListView | searchResultsListView | Displays the search results |
| 2 | Back | Button | backButton | Allows the user to go to the search flights screen |
| 3 | Home | Button | homeButton | Allows the user to go to the home screen |
| 4 | Proceed To Book | Button | proceedToBookButton | Allows the user to go to the pro forma invoice screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| SearchResults\_Load | Form | Loads the search flights screen  Populates the source and destination locations |
| proceedToBook\_Click | proceedToBookButton | Allows the user to go to the pro forma invoice screen |
| backButton\_Click | backButton | Allows the user to go to the search flights screen |
| homeButton\_Click | homeButton | Allows the user to go to the home screen |

### UI Design



### Pro-forma Invoice

### Functional Requirement

Pro-forma Invoice screen displays the flight details, customer name, total price and booking date. The user then books the tickets.

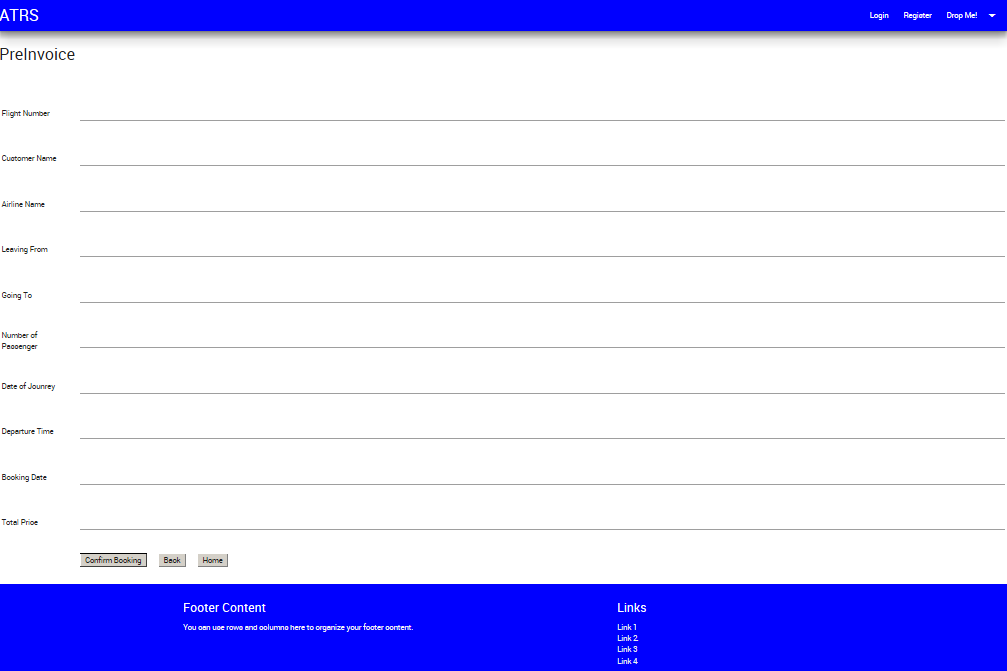
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Flight Number | Label | flightNumberLabel |  |
| 2 | Flight Number | Textbox | flightNumberTextBox | Displays the flight number to the user |
| 3 | Customer Name | Label | customerNameLabel |  |
| 4 | Customer Name | Textbox | customerNameTextBox | Displays the customer name to the user |
| 5 | Airlines Name | Label | airlinesNameLabel |  |
| 6 | Airlines Name | Textbox | airlinesNameTextBox | Displays the airlines name to the user |
| 7 | Leaving From | Label | leavingFromLabel |  |
| 8 | Leaving From | Textbox | leavingFromTextBox | Displays the source location to the user |
| 9 | Going To | Label | goingToLabel |  |
| 10 | Going To | Textbox | goingToTextBox | Displays the destination location to the user |
| 11 | Number Of Passengers | Label | numberOfPassengersLabel |  |
| 12 | Number Of Passengers | Textbox | numberOfPassengersTextBox | Displays the number of passengers to the user |
| 13 | Date Of Journey | Label | dateOfJourneyLabel |  |
| 14 | Date Of Journey | Textbox | dateOfJourneyTextBox | Displays the date of journey to the user |
| 15 | Departure Time | Label | departureTimeLabel |  |
| 16 | Departure Time | Textbox | departureTimeTextBox | Displays the departure time to the user |
| 17 | Booking Date | Label | bookingDateLabel |  |
| 18 | Booking Date | Textbox | bookingDateTextBox | Displays the booking date to the user |
| 19 | Total Price | Label | totalPriceLabel |  |
| 20 | Total Price | TextBox | totalPriceTextBox | Displays the total price to the user |
| 21 | Book | Button | bookButton | Allows the user to go to the final invoice screen |
| 22 | Back | Button | backButton | Allows the user to go back to the search results screen |
| 23 | Home | Button | homeButton | Allows the user to go to the home screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Event** | **Control/Component** | **Event Details** |
| proFormaInvoice\_Load | Form | Loads the pro forma invoice screen. Calculates the total price and displays it to the user |
| bookButton\_Click | bookButton | Allows the user to go to the final invoice e screen |
| backButton\_Click | backButton | Allows the user to go back to the search results invoice screen |
| homeButton\_Click | homeButton | Allows the user to go to the home screen |

### UI Design



### Final Invoice

### Functional Requirements

Final Invoice screen allows the user to confirm the flight ticket booking. It generates the booking Id. The user then confirms the booking.

### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Flight Number | Label | flightNumberLabel |  |
| 2 | Flight Number | ComboBox | FlightNumberComboBox | Displays the flight number to the user |
| 3 | Customer Name | Label | customerNameLabel |  |
| 4 | Customer Name | Textbox | customerNameTextBox | Displays the customer name to the user |
| 5 | Booking Id | Label | bookingIdLabel |  |
| 6 | Booking Id | Textbox | bookingIdTextBox | Displays the booking id to the user |
| 7 | Airlines Name | Label | airlinesNameLabel |  |
| 8 | Airlines Name | Textbox | airlinesNameTextBox | Displays the airlines name to the user |
| 9 | Leaving From | Label | leavingFromLabel |  |
| 10 | Leaving From | Textbox | leavingFromTextBox | Displays the source location to the user |
| 11 | Going To | Label | goingToLabel |  |
| 12 | Going To | Textbox | goingToTextBox | Displays the destination location to the user |
| 13 | Number Of Passengers | Label | numberOfPassengersLabel |  |
| 14 | Number Of Passengers | Textbox | numberOfPassengersTextBox | Displays the number of passengers to the user |
| 15 | Date Of Journey | Label | dateOfJourneyLabel |  |
| 16 | Date Of Journey | Textbox | dateOfJourneyTextBox | Displays the date of journey to the user |
| 17 | Departure Time | Label | departureTimeLabel |  |
| 18 | Departure Time | Textbox | departureTimeTextBox | Displays the departure time to the user |
| 19 | Booking Date | Label | bookingDateLabel |  |
| 20 | Booking Date | Textbox | bookingDateTextBox | Displays the booking date to the user |
| 21 | Total Price | Label | totalPriceLabel |  |
| 22 | Total Price | TextBox | totalPriceTextBox | Displays the total price to the user |
| 23 | Confirm Booking | Button | confirmBookingButton | Allows the user to confirm the booking |
| 24 | Back | Button | backButton | Allows the user to go back to the pro forma invoice screen |
| 25 | Home | Button | homeButton | Allows the user to go to the home screen |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Event** | **Control/Component** | **Event Details** |
| finalInvoice\_Load | Form | Loads the final invoice screen. Generates the booking id and displays it to the user. |
| confirmBooking\_Click | confirmBookingButton | Allows the user to confirm booking. Stores the booking details in the database |
| backButton\_Click | backButton | Allows the user to go back to the pro forma invoice screen |
| homeButton\_Click | homeButton | Allows the user to go to the home screen |

### UI Design



### BookingCalculations

### Functional Requirement

This class is used to handle business rules regarding ticket booking in air ticket management system. It performs the necessary calculations for the invoice. Also it passes the information to the data access layer to get locations, search results, customer details and store the booking details.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| GetLocations | None | Gets the location names from the data access layer |
| ValidateSearchValues | SearchValues searchValues | Validates the search values provided by the user based on the business rules |
| GetSearchResults | SearchValues searchValues | Gets the search results from the data access layer based on the search criteria |
| GetCustomerDetails | String email | Gets the customer id and customer name from the data access layer |
| CalculateTotalFare | int numberOfPassengers,  int fare | Calculates the total fare for the flight tickets |
| GenerateBookingId | int numberOfPassengers | Generates the booking Id |
| SetBookingDetails | BookingStorage bookingStorage | Sends the booking details to data access layer to perform database store operation. |

### BookingConnection

### Functional Requirement

This class is used to connect to the database to perform database operations. It retrieves the locations, search results, customer details and stores the booking details.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| RetrieveLocations | None | Gets the location names from the database |
| RetrieveSearchResults | SearchValues searchValues | Gets the search results from the database based on the search criteria |
| RetrieveCustomerDetails | string email | Gets the customer id and customer name from the database based on the email |
| StoreBookingDetails | BookingStorage bookingStorage | Stores the booking details in the database |
| RetrieveBookingIds | None | Gets the booking ids from the database |

### Cancellation Form

### Functional Requirement

Cancellation screen will allow the user to enter the booking id to cancel the tickets.

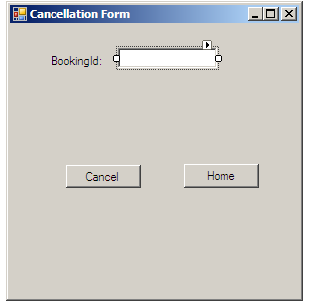
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | BookingID | TextBox | BookingId | Allows the user to enter the booking id. |
| 2 | cancel | Button | cancelButton | Allows the user to cancel the ticket. |
| 3 | Home | Button | HomeButton | Allows the user to return to Home screen. |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Control/Component** | **Details** |
| Home\_Load | Form | Loads the Home screen |
| cancelButton\_Click | cancelButton | Allows the user to cancel the ticket. |
| HomeButton\_Click | HomeButton | Allows the user to redirect to the  Home screen |

### 5.2.1.4 UI Design



### Submission Form

### Functional Requirement

Based on the booking id, the submission screen displays the following information.

* Booking ID
* Booking Date
* Journey Date
* From
* To
* Total Fare
* Cancellation charge
* Refundable Amount

### UI Validations

On click of “cancel” button, check for the following criteria

* Check the booking id generated by the system is in the proper format.
* Leaving From and Going To location cannot be same
* Date of Journey should be future date only.
* Date of Journey should be in the format dd-mm-yyyy

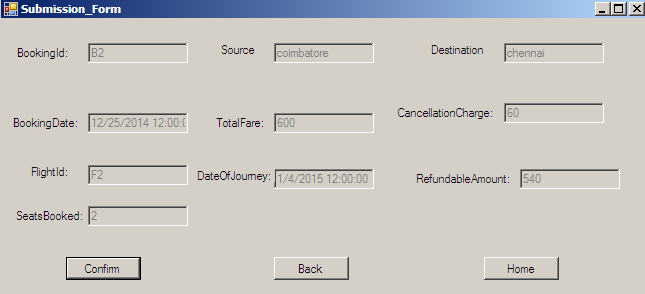
### UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **UI Component** | **Type** | **Name** | **Details** |
| 1 | BookingID | TextBox | BookingId | Displays the booking id to the user. | |
| 2 | BookingDate | TextBox | BookingDate | Displays the booking date to the user. | |
| 3 | DateofJourney | TextBox | JourneyDate | Displays the date of journey to the user. | |
| 4 | From | TextBox | From | Displays the source to the user | |
| 5 | To | TextBox | To | Displays the destination to the user. | |
| 6 | Total Fare | TextBox | Total Fare | Displays the total fare to the user. | |
| 7 | Cancellation Charge | TextBox | Cancellation Charge | Calculate the cancellation charge and display to the user. | |
| 8 | Refundable Amount | TextBox | Refundable Amount | Calculate the refundable amount and display to the user. | |
| 9 | SeatsBooked | TextBox | SeatsBooked | It displays the no of seats booked. | |
| 10 | FlightId | TextBox | FlightId | It displays the flight id. | |

### UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Event** | **Control/Component** | **Event Details** |
| On Load | form | Load all the UI Controls automatically when the user press the cancel button |
| ConfirmButton\_Click | Events | Update the seats in the database. |
| HomeButton\_Click | Form | Load the Home form |
| BackButton\_Click | Form | Load the cancellation Form |

### UI Design



### BookingCancellation

### Functional Requirement

This class is used to fetch the booking id from the user to cancel the ticket.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| CancelBooking() | BookingDetails  bookingDetails | Gets the data table from data access layer. |
| ValidateBookingId() | BookingID | Validates the booking id. |
| CalcuateAmount() | Totalfare,departure date | Calculate the refundable amount and cancellation charge |
| SetDetails() | BookingId | Update the seats availability and delete the booking id. |
| CheckBookingId() | BookingId | It checks the booking id whether it is present in the database or not. |

### BookingDetailsConnection

### Functional Requirement

This Class is used to fetch the booking details from the database based on the booking id.

### Method Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| ConnectBookingInformation() | BookingDetails  cancelticket | Gets the booking details from the database. |
| StoreDetails() | BookingId | It displays the updates seats. |
| GetBookingId | BookingId | It checks the booking id in the database |

### Edit Page

### Functional Requirement

Edit Page allows the customer/user to edit the individual information provided with specific fields.

* + - 1. UI Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **UI Component** | **Type** | **Name** | **Details** |
| 1 | Customer Name | Label | customerNameLabel |  |
| 2 | Customer Name | Textbox | customerNameTextBox | Get the customer name details of employee from admin |
| 3 | Customer Name Validator | Required Field Validator | customerNameRequiredFieldValidator | Checks if the customer name has been entered. |
| 4 | Gender | Label | Gender |  |
| 5 | Gender | Dropdownlist | Gender | Gets the Gender details of customer |
| 6 | Date of Birth | Label | dobLabel |  |
| 7 | Date of Birth | Textbox | dateOfBirthTextBox | Gets the date of birth of customer |
| 8 | Date of Birth Validator | Required field Validator | dobRequiredFieldValidator | Checks if Date of Birth is entered. |
| 9 | Email | Label | emailLabel |  |
| 10 | Email | Textbox | emailTextBox | Get the email of customer |
| 11 | Email Validator | RegularExpressionValidator | emailRegularExpressionValidator | Validates if email address is entered in correct format. |
| 12 | Phone No | Label | phoneLabel |  |
| 13 | Contact No | Textbox | phoneTextBox | Gets the phone no of customer. |
| 14 | Phone Validator | RegularExpressionValidator | phoneRegularExpressionValidator | Validates if phone no is entered in correct format. |
| 15 | Message Label | Label | messageLabel | Displays the status of database operation. |

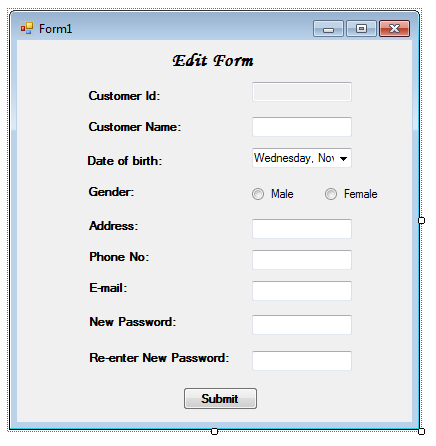
* + - 1. UI Methods Definition

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Parameter** | **Details** |
| ValidatePassword | Email  Password | Check if customer’s email and password is correct and displays appropriate message to the customer. |

* + - 1. UI Events Definition

|  |  |  |
| --- | --- | --- |
| **Event** | **Control/Component** | **Event Details** |
| On Load | Page | Load the customer details from database. |
| Edit/Update\_Click | UpdateButton | Edits customer details to the Database. |

* + - 1. UI Design



### Registration

* **Purpose:**
* **Constraints**: None
* **Persistent**: No (created at system initialization from other available data)

#### Attribute Descriptions

1. **Attribute:**

**Type:**

**Description:**

**Constraints:**

1. **Another attribute…**

#### Method Descriptions

**Method:**

Return Type: *Boolean*

Parameters:

Return value: *success or failure*

Pre-condition:

Post-condition:

***2.* Attributes read/used:**

**3. Methods called:**

**4. Processing logic:**

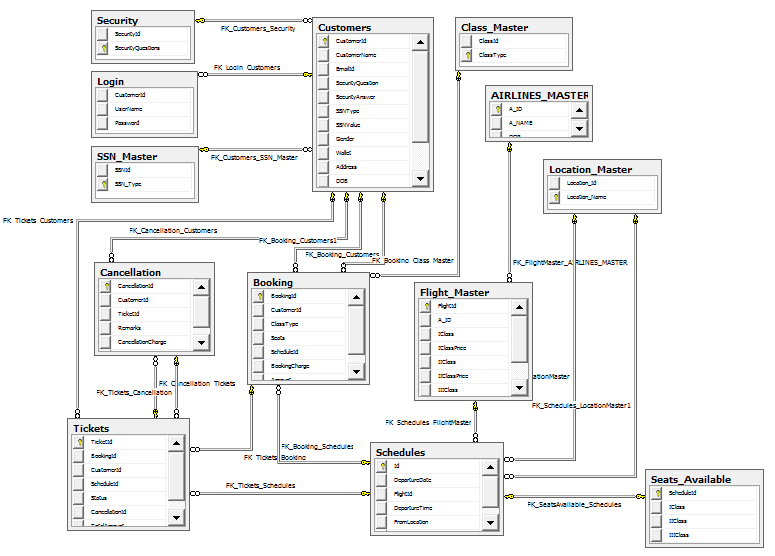
# Database Design

## Data Model

*<Map persistent design classes into the data model>*

### Identification of Tables

### 



### Table Details

UserDetails

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| CustomeId | Numeric | 10 | No |
| CustomerName | Varchar | 30 | No |
| Email | Varchar | 30 | No |
| Password | Varchar | 150 | No |
| Dob | Varchar | 30 | No |
| Address | Varchar | 50 | No |
| Phone | Numeric | 10 | No |
| Gender | Varchar | 6 | No |
| SsnType | Varchar | 30 | No |
| SsnNumber | Varchar | 30 | No |

**SSNMaster**

|  |
| --- |
| **Ssntype** |
| Pan card |
| Voter-id card |
| Driving licence |
| passport |

### 

|  |  |  |  |
| --- | --- | --- | --- |
| **AirlinesMaster** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| AirlinesId | varchar | 30 | N |
| AirlinesName | varchar | 30 | N |

|  |  |  |  |
| --- | --- | --- | --- |
| ***SeatsAvailable*** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| FlightId | varchar | 30 | N |
| AvailableSeats | Int |  | N |
| DepartureDate | Date |  | N |

|  |  |  |  |
| --- | --- | --- | --- |
| **FlightMaster** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| FlightId | varchar | 30 | N |
| AirlinesId | varchar | 30 | N |
| TotalSeats | int |  | N |
| Source | varchar | 30 | N |
| Destination | varchar | 30 | N |
| DepartureTime | time | 7 | N |
| ArrivalTime | time | 7 | N |
| Fare | int |  | N |

|  |  |  |  |
| --- | --- | --- | --- |
| **BookingDetails** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| BookingId | varchar | 30 | N |
| BookingDate | Date |  | N |
| FlightId | varchar | 30 | N |
| CustomerId | varchar | 30 | N |
| TotalPrice | int |  | N |
| SeatsBooked | int |  | N |
| DepartureDate | date |  | N |
| DepartureTime | time | 7 | N |

|  |  |  |  |
| --- | --- | --- | --- |
| **AirlinesMaster** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| AirlinesId | varchar | 30 | N |
| AirlinesName | varchar | 30 | N |

|  |  |  |  |
| --- | --- | --- | --- |
| **LocationMaster** | | | |
| **Column Name** | **Data Type** | **Length** | **Nulls** |
| LocationId | int |  | N |
| LocationName | varchar | 30 | N |

### Referential Integrity Constraints

*<Specify all Primary Key, Foreign key, Triggers and other such constraints that enforce data integrity as well as referential integrity constraints>*

### Computations/processing in Database

|  |  |  |
| --- | --- | --- |
| **Stored Procedure Names** | **Fields Used** | **Description** |
| GetLogin | * Email * password | Used to check whether the password and email id matches with the credentials present in the database. |
| StoreRegistration | * customerid * name * email * password * dob * address * phone * gender * ssntype * ssnnumber | Used to insert the data entered by the user during registration process into the database. |
| GetSsnTable | * ssntype | Used to fetch the SSN type details from the database to the presentation layer. |
| GetCustomerId | * customerid | Used to get the last customerid generated. |
| EmailValidate | * email | Used to get the email id from the database matching with the user entered email. |
| SsnNumberValidate | * ssnnumber | Used to get the ssn number from the database matching with the user entered ssn number. |

### 8.1.4 Object Model

|  |  |
| --- | --- |
| **Object Name** | **Object Description** |
| userDetails | Gets the user details from the user |
| userFetchDetails | Used to call the method in business logic layer |
| userStoreDetails | Used to call the method in data access layer |

|  |  |
| --- | --- |
| **Object Name** | **Object Description** |
| bookingDetails | Get the booking details from the user |
| bookingCancellation | Used to call the method in business logic layer |
| bookingDetailsConnection | Used to call the method in data access layer |

|  |  |
| --- | --- |
| **Object Name** | **Object Description** |
| GetPassword | Get the password from the database |
| ValidatePassword | Checks the given password with the database password. |
| UpdateCustomerDetails | Updates the customer details to the database |

|  |  |
| --- | --- |
| searchFlight | Gets the flight details from the user |
| showResult | Displays the search results to the user |
| flightDetails | Used to call the method in business logic layer |
| flightDetailsConnection | Used to call the method in data access layer |

|  |  |
| --- | --- |
| **Object Name** | **Object Description** |
| searchFlights | Gets the search criteria from the user |
| searchResults | Displays the search results to the user and allows the user to select a particular flight |
| proFormaInvoice | Displays the proforma invoice to the user with booking date and total price |
| FinalInvoice | Displays the final invoice to the user with the generated booking id |
| bookingCalculations | Used to call the methods in business logic layer |
| bookingConnection | Used to call the methods in data access layer |

### Access Control Details

*<Administrative details which are not specific to any program of the application, but connected to database/file system.*

* *Administration*
* *Security and access rights*
* *Record locking, etc>.*

## Storage Characteristics

*<Storage characteristics needs to be mentioned here and should include current as well as estimates for expansion in the future. This include some of the following:*

* *Disk page density*
* *Disk page location*
* *Disk space allocation etc*

## Database Performance Improvement Measures

* *Usage of Indexes to improve query performance (Keep in mind that indexes have a hidden cost: indexes cost time to update and occupy disk space. Be sure you get value from using them)*
* *Right choice of indexing strategy*
* *Query optimization*
* *Use of Prepared Statements and parameter binding*

*<INSERT YOUR DATA MODEL HERE>*

# References

# Change Log

*Please note that this table needs to be maintained even if a Configuration Management tool is used.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Changes made | | | |
| V1.0 | *<First version>* | | | |
| V1.1 | *<If the change details are not explicitly documented in the table below, reference should be provided here>* | | | |
| Page no | Changed by | Effective date | Changes effected |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| V1.2 | *<If the change details are not explicitly documented in the table below, reference should be provided here>* | | | |
| Page no | Changed by | Effective date | Changes effected |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |