**Integrated Assignment**

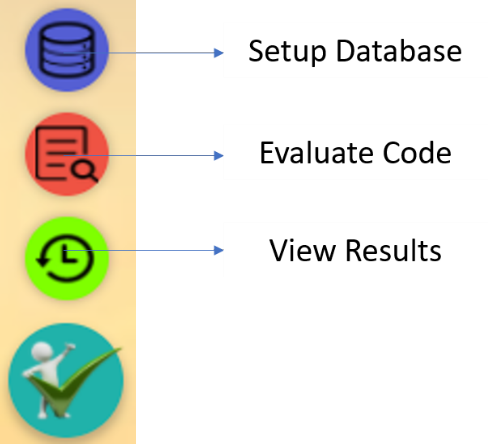
**Developing a Single Page Application using Angular**

**Instructions to use the project file provided:**

* You need to code the assessment in the folder containing partially coded project
* Read the problem statement, examples and the other details provided carefully and implement the solution
* Download the project **InfyAirlines-toTrainee** to your system
* **InfyAirlinesWS** folder contains the **server-side code**
* **InfyAirlinesUI** folder must contain the **client-side code**
* **DO NOT** alter the function name or the argument list of the function that is provided to you
* **DO NOT** alter the **meta data** provided in the components
* **DO NOT** add any new functions apart from the one given in the file where you write the solution
* Use the **bootstrap classes** wherever necessary
* Do not modify any **meta data** provided in the files

**Note:**

* The express application runs in port **1050**
* Execute **npm install** command in both the **InfyAirlinesWS** and **InfyAirlinesUI** folders
* Execute **ng serve** command to run the angular app and **node app** command to run the express web service.
* Once the webservice is running, hit the URL **http://localhost:1050/setupdb** to initialize the database
* Use these options provided on the bottom right corner of the application for different purposes as mentioned



**Problem Description:**

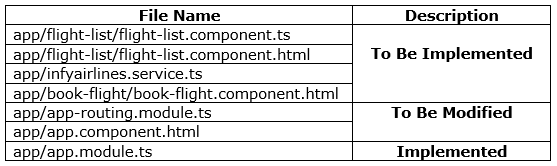
**InfyAirlines** wants to automate the process of booking flight tickets to their customers. The following functionalities are required to be implemented.

* Display the list of flights available for Booking
* Book the required Number of seats in the selected flight
* Display all the bookings done
* Provide Ability to filter the bookings based on either of, or both customerId and flightId
* Update the number of tickets booked in a flight
* Ability to cancel a ticket which is already booked
* Navigate to PageNotFound Component when a page is requested which does not exist

You need to implement these requirements one by one to complete the application for Infy Airlines

***Requirement 1:- Display All the flights available for Booking***

**Implementation Details for Requirement 1**



**app/infyairlines.service.ts**

* This file contains a method **getAllFlights()** which will help in fetching all the flights Available for booking.
* **getAllFlights() : (To Be Implemented)**
  + It should send GET request to the URL [**http://localhost:1050/flights/**](http://localhost:1050/flights/) which in turn returns an observable of type Flight[]
  + The same observable object should be returned from the function

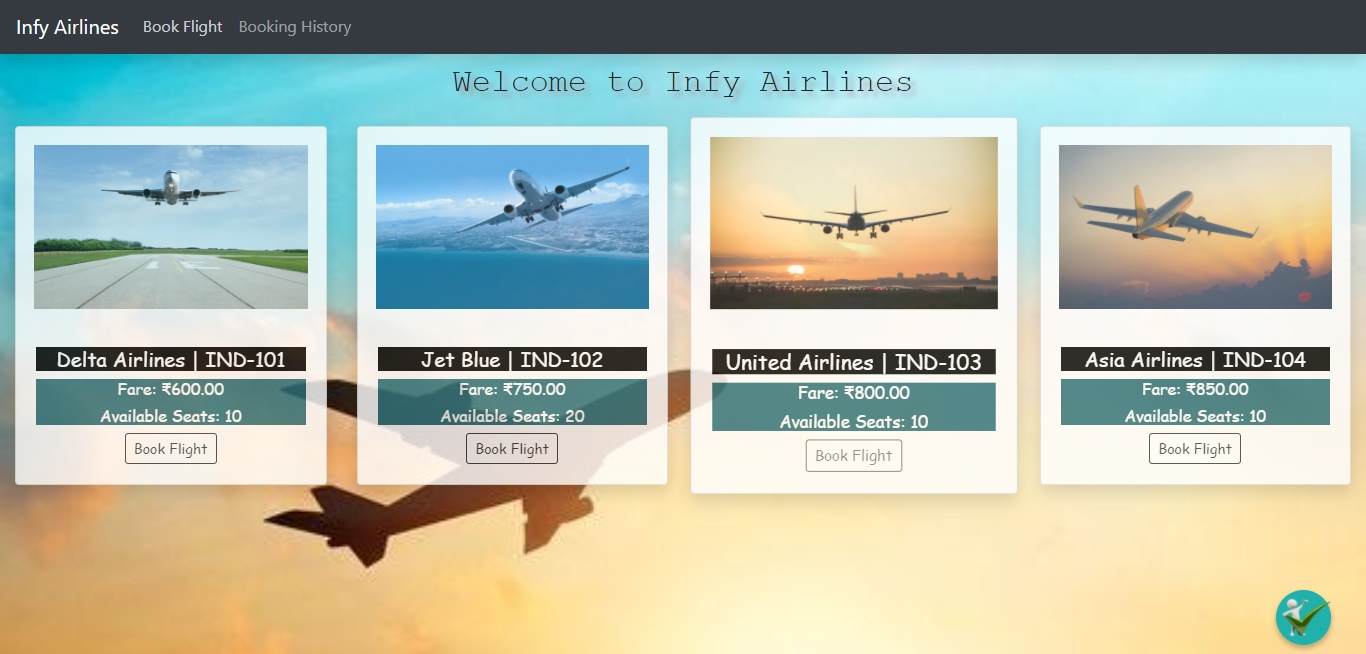
**app/flight-list/flight-list.component.ts**

This file contains the following properties and methods to be used for implementing this functionality:

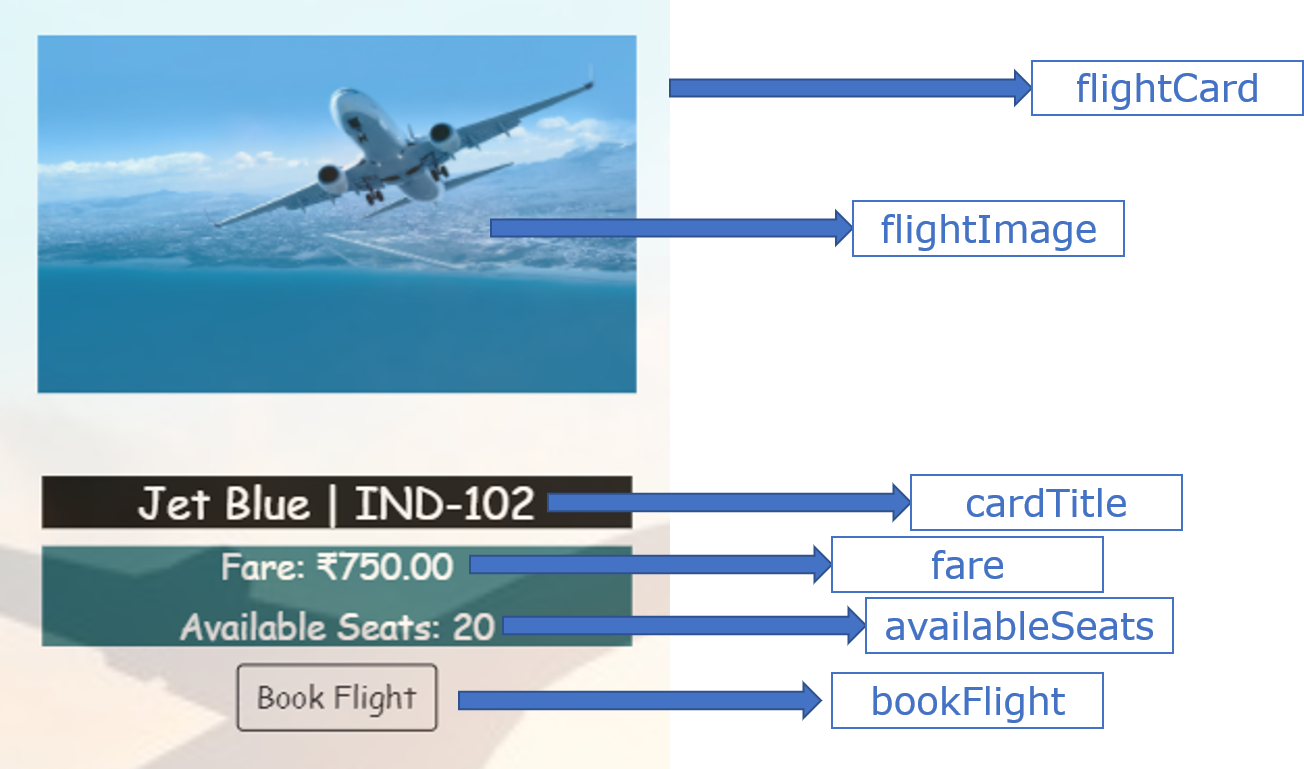
* flightsArray = [];
* errorMessage = “”;
* getAllFlights()
* **getAllFlights() : (To Be Implemented)**
  + This method should be used to fetch all the available flights for booking
  + It should first reset **flightsArray** to empty array and **errorMessage** to empty string
  + It should then invoke **getAllFlights()** method of InfyairlinesService class which in turn returns an observable
  + In **success** case, the response should be populated into **flightsArray** property
  + In **error** case, the error message should be populated in **errorMessage** property
  + This method should be invoked on load of the component

**app/flight-list/flight-list.component.html**

The template should be coded in such a way that when the component loads in browser it displays the flights available for booking as cards, as shown in the below image:



* Ensure the cards are having ids as mentioned below:



* Book Flight button should remain disabled if the **status** is **cancelled** or available seats is 0 or less.

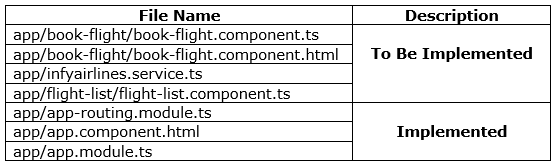
* To render the cards in browser, do the following modifications in the files as mentioned:
  + Load the flight-list component in book-flight.component.html as a child component.
  + Add a route as mentioned below in **app-routing.module.ts:**



* + Add appropriate code in **app.component.html** to display the routed component

***Requirement 2:- Select a flight and book the required number of tickets in the selected flight***

**Implementation Details for Requirement 2**



**app/flight-list/flight-list.component.ts**

This file contains the following properties and methods to be used for implementing this functionality:

* flightEmitter 🡺 It should be made as an instance of EventEmitter to pass the selected flight Data to the **Book Flight(parent) component**
* sendFlightData()
* **sendFlightData(flight) : (To Be Implemented)**
  + It should use the flightEmitter property to emit the selected flight object passed as a parameter to the function.
  + This method should be invoked on click of Book Flight button

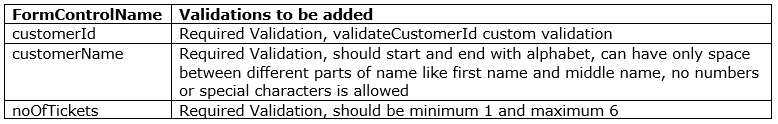
**app/infyairlines.service.ts: (To be implemented)**

* **bookFlight(flightId, bookingForm):**
  + This method consumes the web service exposed at the URI.
  + It should send a **PUT** request to the URI **http://localhost:1050/book/:flightId** by passing flightId as a route parameter and **bookingForm** data through body of request
  + It should return the same response as received from the webservice

**app/book-flight/book-flight.component.ts**

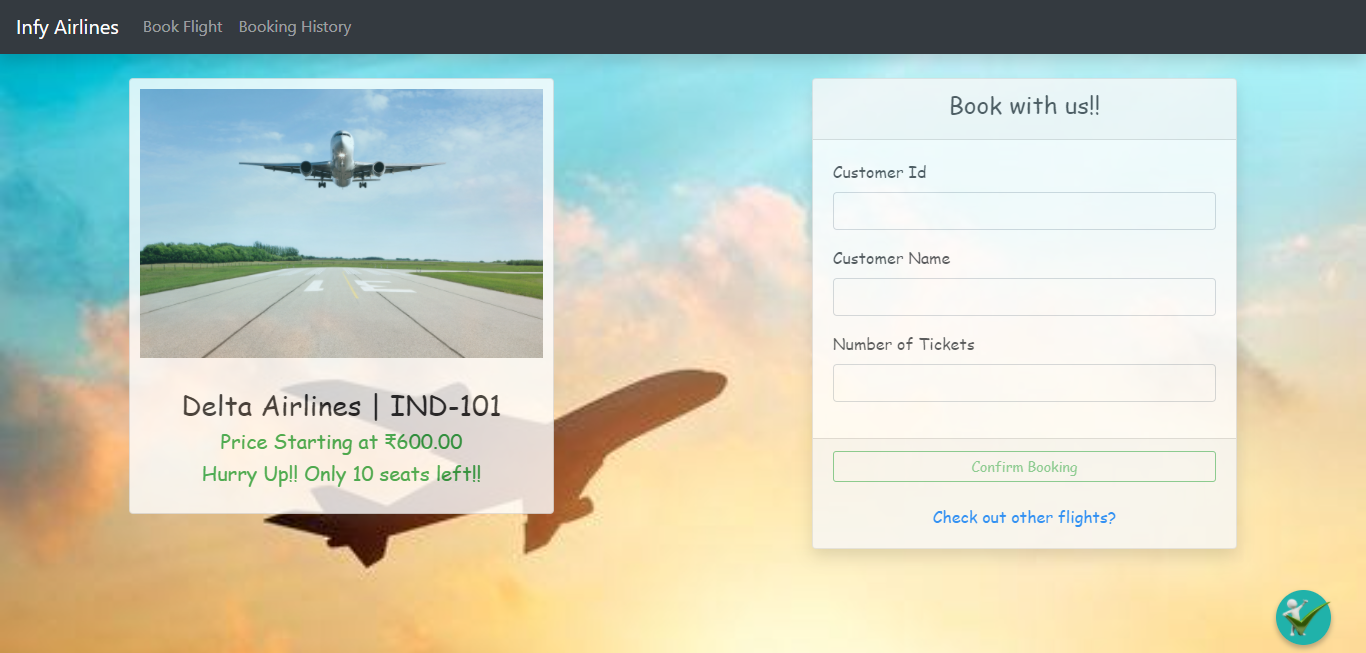
This file contains a class **BookFlightComponent** with following properties and methods to be used for implementing this functionality:

* successMessage = null;
* errorMessage = null;
* flightDetails;
* bookingForm;
* recieveFlightData();
* bookFlight()
* validateCustomerId() 🡺 custom validator for **customerId** field
* **recieveFlightData(flight) : (To Be Implemented)**
  + It should take the **flight** object received as a function parameter and assign it to **flightDetails** property.
  + It should reset the **bookingForm** values by invoking the **reset()** method.
  + It should also reset **successMessage** and **errorMessage** to **null**
* **validateCustomerId(formControl): (To be Implemented)**
  + It should take customerId form control as a parameter and validate It
  + If should validate if the **customerId** starts with an uppercase alphabet followed by a 4-digit number (not starting with 0)
  + If the validation fails, it should return an object with **customerIdError** property with value as an object containing **message** property as **“Enter a valid Customer Id (E.g., R1001)”**
  + For a valid customerId, it should return **null**
* **bookFlight()** :- **(To be implemented)**
  + It should reset the **errorMessage** and **successMessage** to **null**
  + It should invoke **bookFlight()** of **InfyairlinesService** by passing **flightId** of the selected flight and value of **bookingForm** as a parameter, which in turn returns an observable
  + It should populate **successMessage** and **errorMessage** appropriately using the returned observable object
  + This method should be invoked on click of **Confirm Booking** button.
* A FormGroup object named **bookingForm** should be created using the **FormBuilder** instance, having **customerName**, **noOfTickets** and **customerId** as form controls
* Add the validators for each form controls as mentioned below:

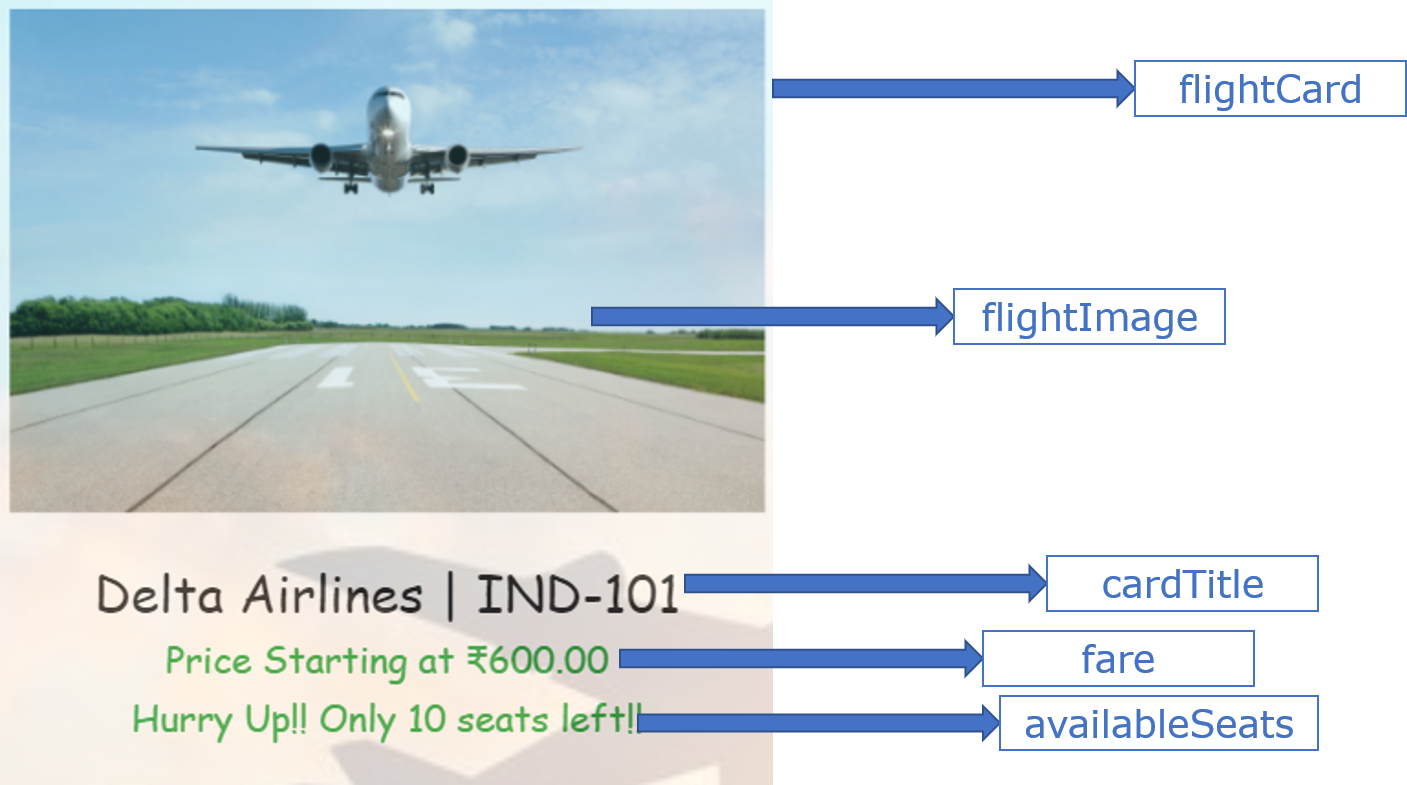


**app/book-flight/book-flight.component.html**

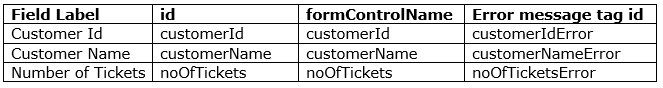
The template should be coded in such a way that when the component loads in browser it displays the flights available for booking as cards, as shown in the below image:



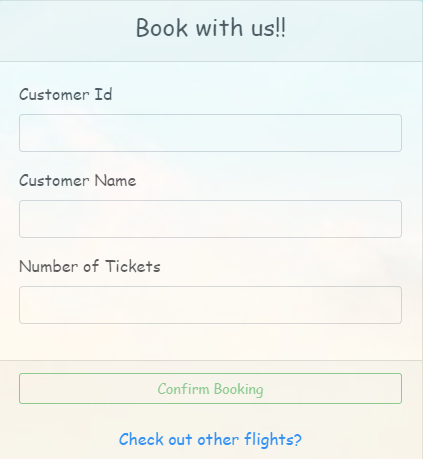
* Modify the code written in ***book-flight.component.html*** to receive the flight data sent using the **flightEmitter** instance from flight-list component.
* The selected flight card should be rendered with the ids as shown below:



* Create the form template with the details as mentioned below:

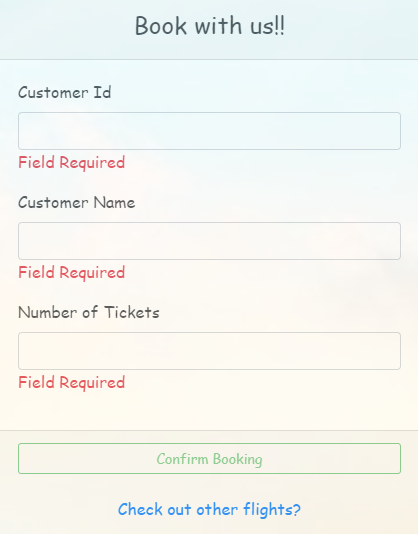
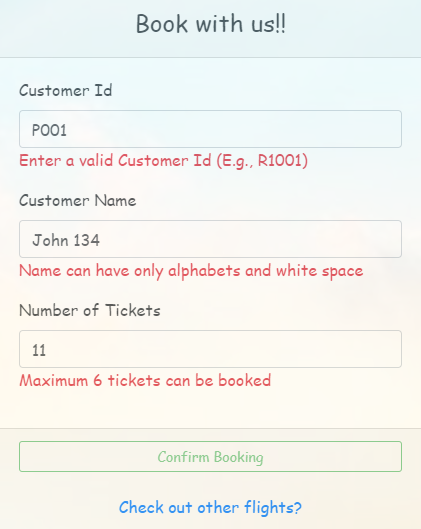


This will create a form as shown below:

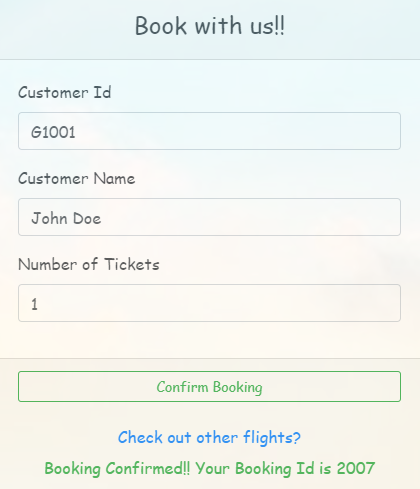


* The error messages should be displayed only when the it is **invalid** and **dirty**.
* If a field has multiple error messages for different validations, they should be coded in different span tags and then grouped together into a single div tag and the id should be added to the div tag

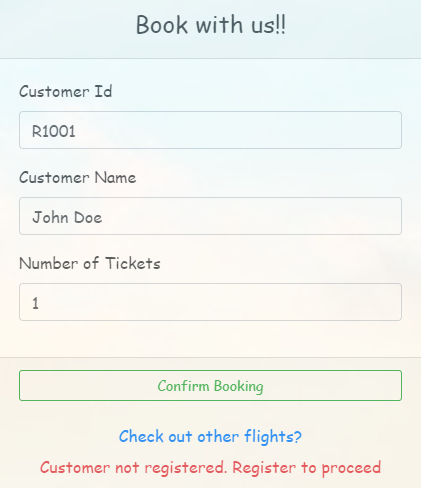
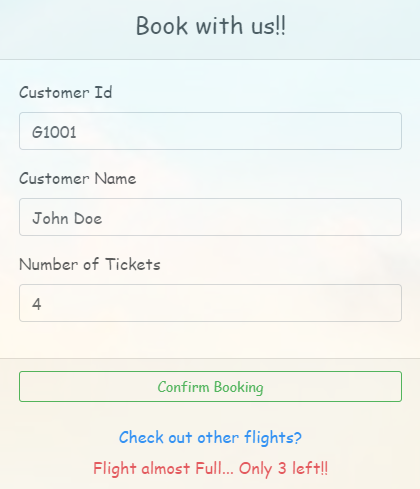
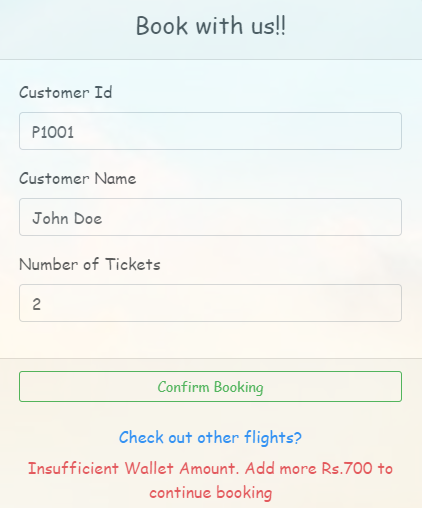
**Required Validation Other Validations**

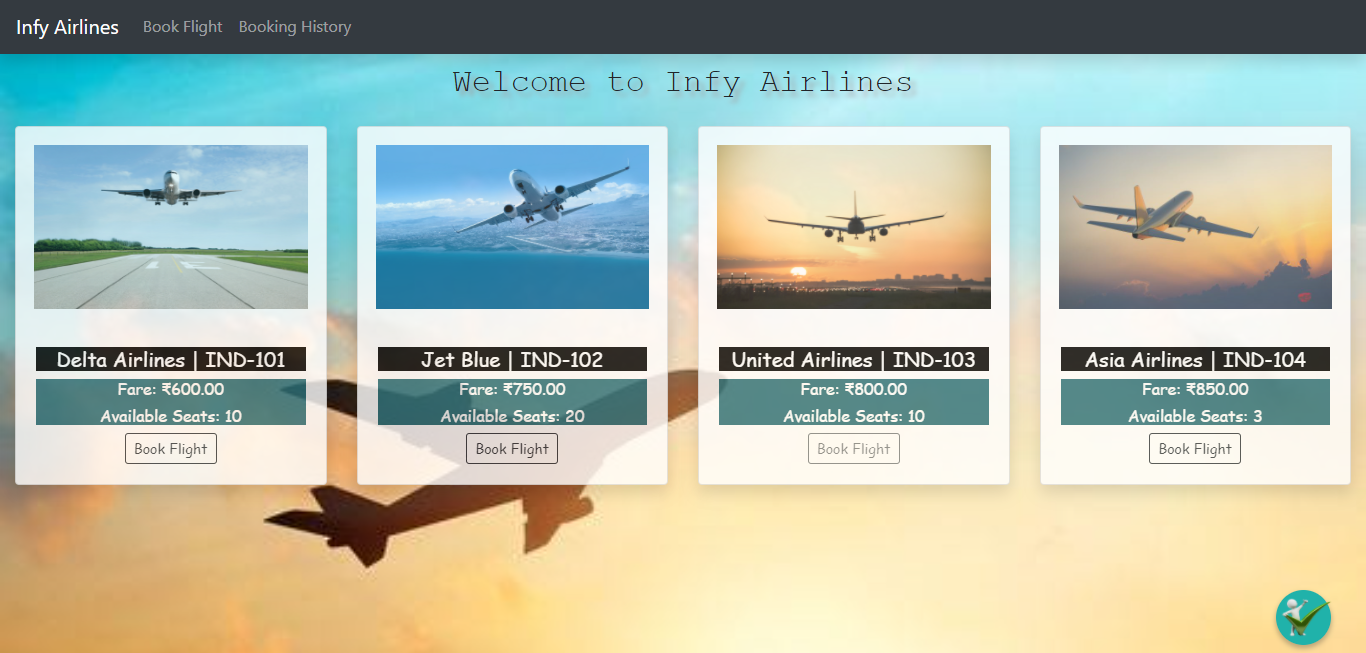
* **Confirm Booking** button should be disabled until all the fields have valid input. It should have an id as **“confirmBooking”**
* **Check out other flights ?** button should have id as **checkOutFlights**
* Form Level success and error message tags should have **successMessage** and **errorMessage** as id’s respectively
* On successful Booking, the success message should be displayed as shown below:



* In case if you have entered in appropriate input to different fields for which booking cannot be done. It should render the error Messages for different cases as shown below:

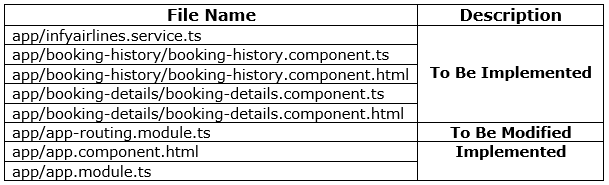
  

* On Click of Check out other flights button, it should display the list of flights again. Use appropriate property of BookFlight component to achieve this



***Requirement 3:- Display All the bookings available in InfyAirlines Database***

**Implementation Details for Requirement 3**



**app/infyairlines.service.ts**

* This file contains a method **getAllBookings()** which will help in fetching all the flights Available for booking.
* **getAllBookings() : (To Be Implemented)**
  + It should send **GET** request to the URL [**http://localhost:1050/bookings/**](http://localhost:1050/bookings/) which in turn returns an observable of type FlightBooking[]
  + The same observable object should be returned from the function

**app/booking-history/booking-history.component.ts**

This file contains a class **BookingHistoryComponent** class with following properties and methods to be used for implementing this functionality:

* bookingsArray = [];
* errorMessage = null;
* getAllBookings()
* **getAllBookings() : (To Be Implemented)**
  + This method should not take any parameter and fetch all the bookings available.
  + It should invoke **getAllBookings()** method of **InfyairlinesService** which in turn returns an observable of type **FlightBooking** Array
  + In success case, the returned response is an array with data in following structure at every index:

**{**

**flightId: "IND-101",**

**bookings: [**

**{**

**bookingCost: 2250**

**bookingId: 2002**

**customerId: "P1001"**

**customerName: "Nicholas"**

**flightId: "IND-102"**

**noOfTickets: 3**

**},**

**{**

**bookingCost: 1500**

**bookingId: 2004**

**customerId: "G1001"**

**customerName: "Joe Richardson"**

**flightId: "IND-102"**

**noOfTickets: 2**

**}**

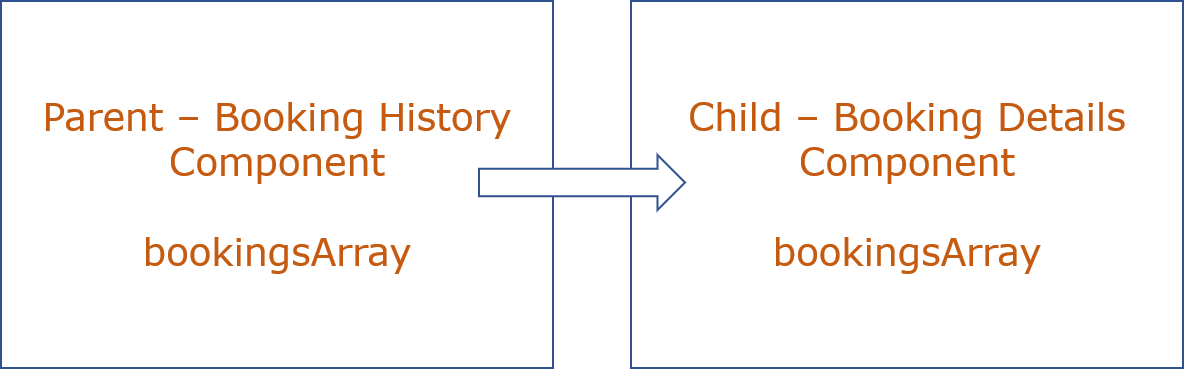
**]**

**}**

* + It should then iterate over the **bookings** array of each object in the response array and populate corresponding flightId into each booking of bookings array
  + Once the flightId is added to the booking object, it should be pushed into the **bookingsArray**
  + Once the iteration is completed for all the bookings, the **bookingsArray** should be sorted ascending order of **bookingId**
  + It should populate **errorMessage** property appropriately with the returned observable in error case
  + This method should be invoked on load of the component

**app/booking-history/booking-history.component.html**

* The **bookingsArray** should be passed as an **Input** to the **BookingDetails** component. Use nesting of components for achieving this, considering **BookingHistoryComponent** as Parent and **BookingDetailsComponent** as child component
* The **bookingsArray** will then be used in the **BookingDetailsComponent** to display the **bookings** in the form of a table

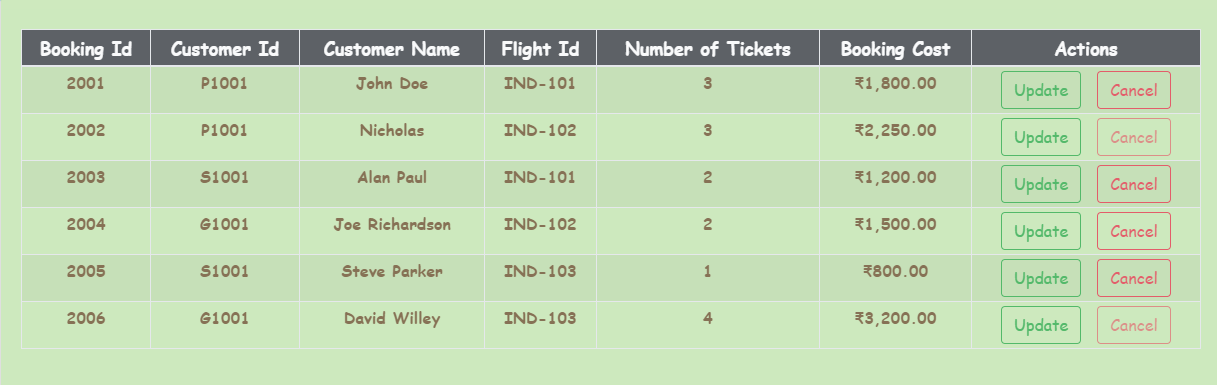


**app/booking-details/booking-details.component.ts**

* **bookingsArray** should be used as an Input property to receive the booking data sent from **BookingHistoryComponent**

**app/booking-details/booking-details.component.html**

* Code the **BookingDetailsComponent** html template to render the booking table as shown below:



* **Cancel** button should be disabled if the bookingCost is greater than **2000**
* **Update** and **Cancel** buttons should have id as **update** and **cancel** respectively
* The table should be displayed only when the **bookingsArray** has the booking data available. Else, it should display error message as shown:

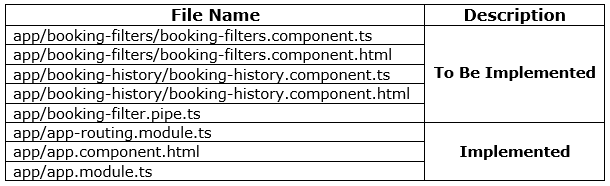


* Add the following route in **app-routing.module.ts** to render the output on browser:



***Requirement 4:- Apply Filters for displaying the selected bookings based on flightId and customerId***

**Implementation Details for Requirement 4**



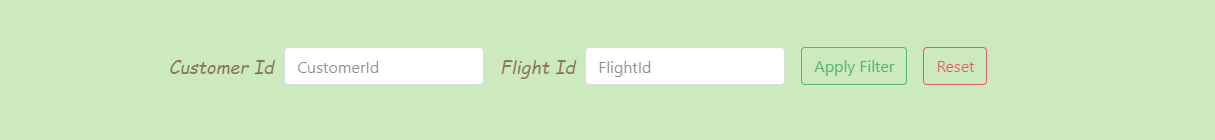
**app/booking-filters/booking-filters.component.ts**

This file contains a class **BookingFiltersComponent** with following properties and methods to be used for implementing this functionality:

* flightId
* customerId
* filtersEmitter
* sendFilterData()
* resetFilters()
* **sendFilterData() : (To Be Implemented)**
  + It should use the **filtersEmitter** instance to emit an object containing **flightId** and **customerId** entered by the user
  + This method should be invoked on click of **Apply Filter** Button
* **resetFilters() : (To Be Implemented)**
  + It should use the **filtersEmitter** instance to emit an object containing **flightId** and **customerId** both as empty string
  + This method should be invoked on click of **Reset** Button

**app/booking-filters/booking-filters.component.html**

* Code the **BookingFiltersComponent** html template to render the booking filter fields as shown below:



* **customerId, flightId** fields should have id’s as **customerId** and **flightId** respectively
* **Apply Filter** and **Reset** buttons should have id’s as **applyFilter** and **reset** respectively

**app/booking-history/booking-history.component.ts**

This file contains a class **BookingHistoryComponent** with following properties and methods to be used for implementing this functionality:

* flightId
* customerId
* recieveFiltersData()
* **receiveFiltersData(filtersObj) : (To Be Implemented)**
  + It should set **customerId** and **flightId** properties with using **filtersObj** received as function parameter

**app/booking-filter.pipe.ts**

* The pipes transform should take the **bookingsArray** as the value and **customerId** and **flightId** as arguments and filter the bookingsArray based on the specified values
* If only **flightId** is provided it should return all **bookings** for that flight
* If only **customerId** is provided it should return all **bookings** from different flights for the given customerId
* If both **flightId** and **customerId** is provided it should filter the **bookings** made by the given **customer** in the given **flight**
* In case none of the values are provided, it should return entire **bookingsArray** as a value

**app/booking-history/booking-history.component.html**

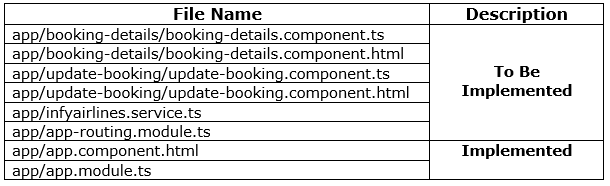
* The **filtersEmitter** instance should be used to invoke **receiveFiltersData()** method while loading **BookingFiltersComponent** as a child:



* Modify the code where **bookingsArray** is being passed to the **BookingDetailsComponent** by adding **bookingFilter** as a pipe and passing the **flightId** and **customerId** received from **BookingFiltersComponent** as arguments

***Requirement 5:- Update the no of seats booked in a already made booking***

**Implementation Details for Requirement 5**



**app/app-routing.module.ts**

* Add the following route to load the UpdateBooking Component for the given path:



**app/infyairlines.service.ts**

* This file contains **InfyairlinesService** class with **updateBooking()** method to be used for this requirement:
* **updateBooking(formData) : (To be Implemented)**
  + It should send **PUT** request to the URL [**http://localhost:1050/update/**](http://localhost:1050/update/) by passing **bookingId** from formData as the route parameter and **formData** as the request body which in turn returns an observable of type any
  + The same observable object should be returned from the function

**app/booking-details/booking-details.component.ts**

* This file contains a method **openUpdateForm()** to implement this given requirement:
* **openUpdateForm() : (To Be Implemented)**
  + It takes the selected Booking for which the **noOfTickets** needs to be updated as a parameter
  + It should invoke appropriate router method to load the Update Booking component by passing **bookingId** and **flightId** of the selected booking.

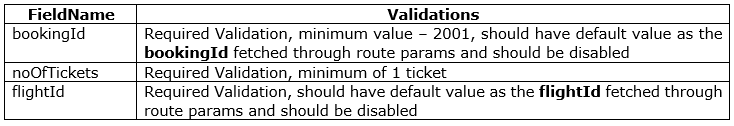
**app/booking-details/booking-details.component.html**

* The **openUpdateForm()** method should be invoked on click of **Update** button for any of the bookings displayed.



**app/update-booking/update-booking.component.ts**

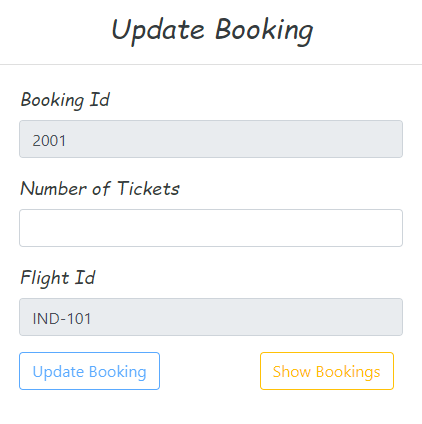
* **UpdateBookingComponent** class contains the following properties and methods:
* bookingId: number
* flightId: string
* updateBookingForm: FormGroup
* successMessage: string
* errorMessage: string
* updateBooking()
* Instances of **FormBuilder**, **ActivatedRoute** and **InfyairlinesService** should be injected into the constructor
* The values of **bookingId** and **flightId** should be fetched from the route params and assigned to **bookingId** and **flightId** properties of UpdateBookingComponent respectively
* A FormGroup object named **updateBookingForm** should be created using the **FormBuilder** instance, having **bookingId**, **noOfTickets** and **flightId** as form controls
* Add the validators for each form controls as mentioned below:



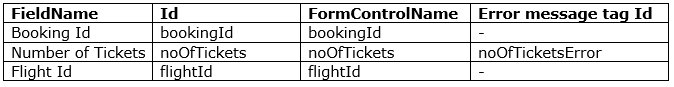
* **bookingId** and **flightId** fields should be populated with the value received and should be **disabled**
* **updateBooking()**
  + This method should not take any parameter and update the **noOfTickets** for the selected booking
  + It should reset the values of **successMessage** and **errorMessage** to **“”**
  + It should invoke the **updateBooking()** method of **InfyairlinesService** by passing an object containing **bookingId** and **noOfTickets** which in turn returns an observable
  + It should populate the **successMessage** and **errorMessage** properties appropriately with the observable returned

**app/update-booking/update-booking.component.html : (To be implemented)**

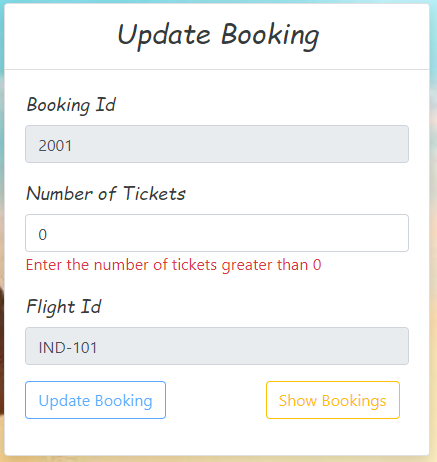
The view of **UpdateBookingComponent** is as shown below:



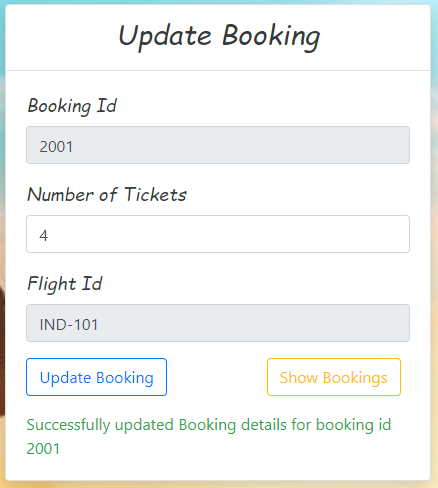
* The following id and form control name should be added for each field:



* The error message for the field should be displayed when it is **invalid** and **dirty**
* If a field has multiple error messages for different validations, they should be coded in different span tags and then grouped together into a single div tag and the id should be added to the div tag

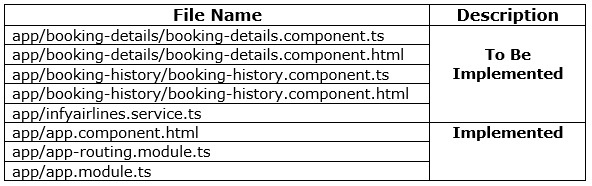


* If appropriate no of seats is specified for which wallet amount is sufficient and seats are also available, the success message should be displayed as shown:



***Requirement 6:- Cancel the selected Booking***

**Implementation Details for Requirement 6**



**app/booking-details/booking-details.component.ts**

* This file contains the following properties and methods which will help in implementing this requirement.
  + bookingEmitter
  + sendCancellationId()
* **sendCancellationId() : (To Be Implemented)**
  + It should use the **bookingEmitter** instance to emit the **bookingId** of the selected booking to cancel

**app/booking-details/booking-details.component.html**

* **sendCancellationId()** method should be invoked on click of **Cancel** button by passing the **bookingId** of the selected booking to cancel as a parameter

**app/infyairlines.service.ts**

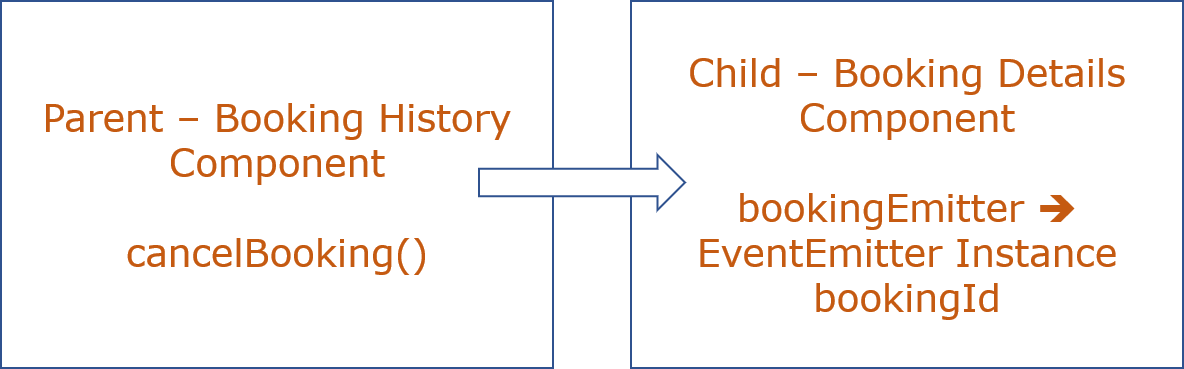
* This file contains a method **cancelBooking()** which will help in implementing this requirement
* **cancelBooking(bookingId) : (To Be Implemented)**
  + It should take bookingId as a parameter and send an API call to delete the booking
  + It should send a **DELETE** request to the URL **http://localhost:1050/cancel/:bookingId** which in turn returns an Observable as a response.
  + The same observable object should be returned from the method

**app/booking-history/booking-history.component.ts**

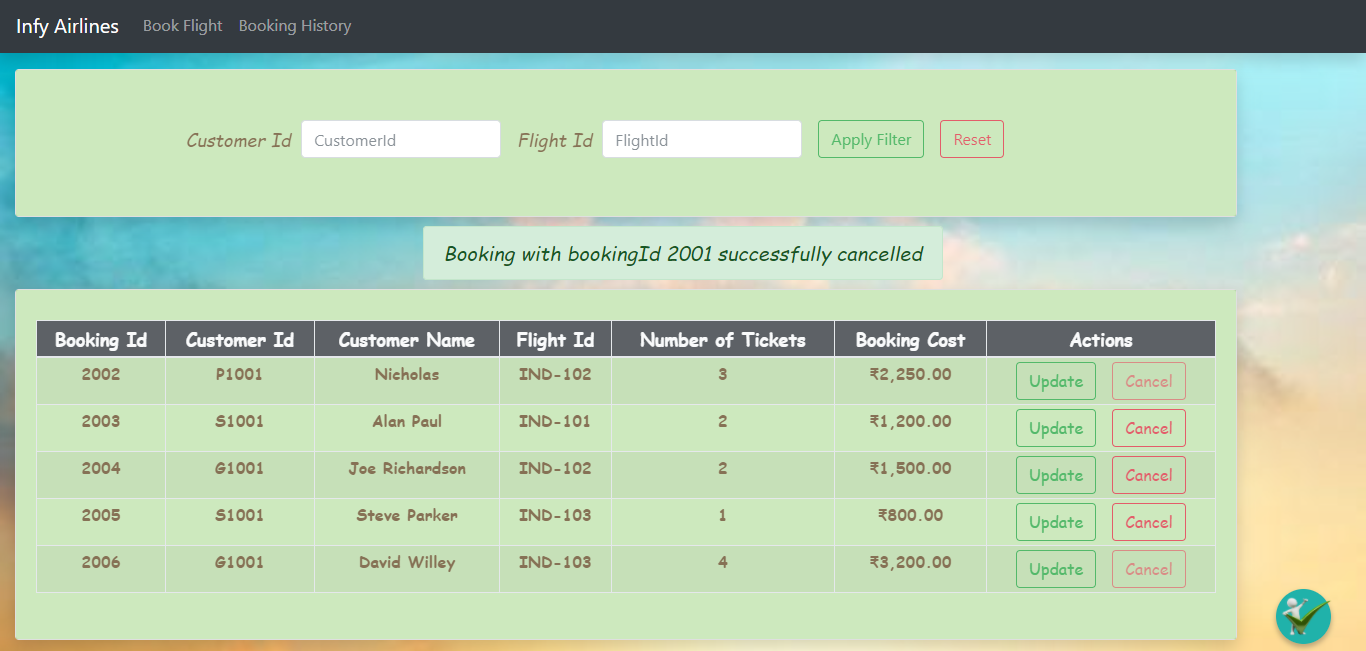
* This file has a class **BookingHistoryComponent** with the following properties and methods that will help in implementing the given requirement:
  + successMessage = “”;
  + errorMessage = “”;
  + bookingsArray;
  + cancelBooking()
* **cancelBooking(bookingId) : (To Be Implemented)**
  + It should take the **bookingId** as a parameter and delete the booking corresponding to it
  + It should first reset **successMessage** and **errorMessage** properties to **null**
  + It should then invoke cancelBooking() method of InfyAirlines service by passing the booking Id as a parameter, which in turn returns an observable
  + In success case,
    - Set the response message to **successMessage** property
    - Remove the booking object corresponding to the given **bookingId**
    - Reset **successMessage** property to **null** after 4 seconds
  + In error case,
    - Set the error response to **errorMessage** property
    - Reset **errorMessage** property to **null** after 4 seconds

**app/booking-history/booking-history.component.html**

* Modify the code by adding **bookingEmitter** instance to invoke **cancelBooking()** method while loading **BookingDetailsComponent** as a child:

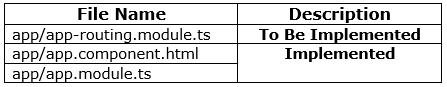


* On deletion **successMessage** or **errorMessage** should be displayed as shown:



***Requirement 7:- Handle the routes for which components are not defined***

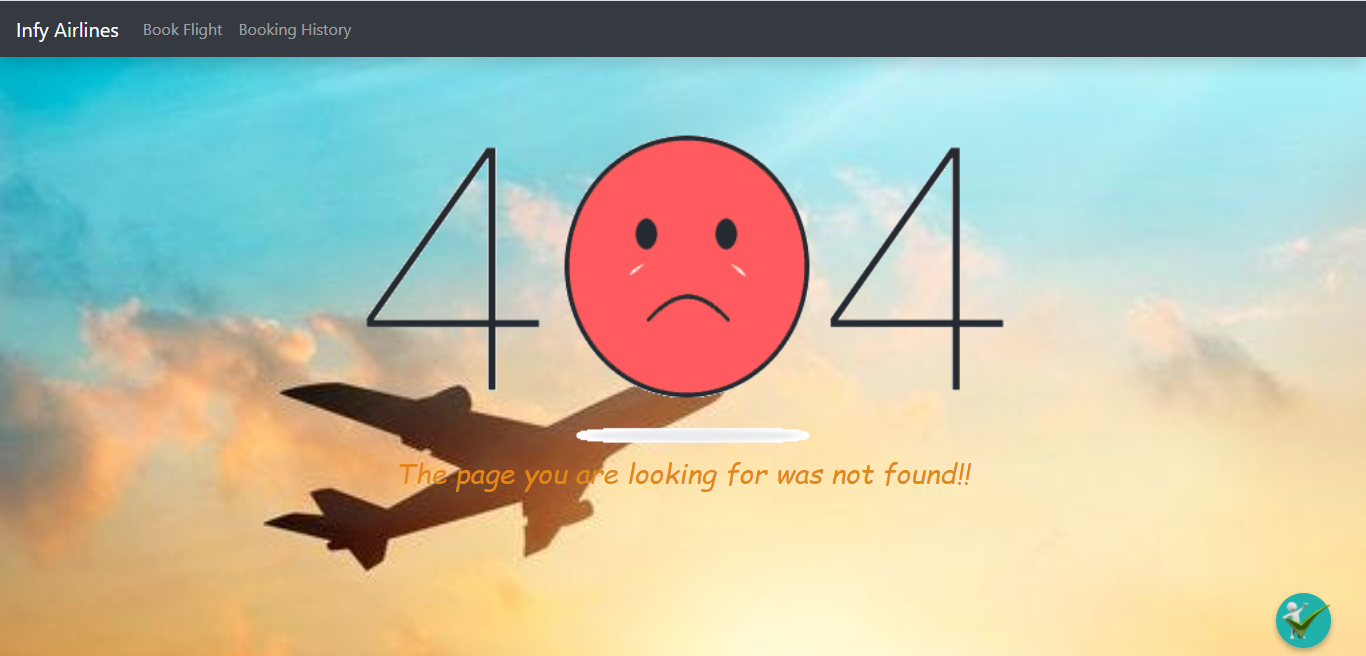
**Implementation Details for Requirement 7**



* Add the route in **app-routing.module.ts** as mentioned below:



* All the requests for which routes are not defined, should be redirected to **‘/pagenotfound’**
* The view of **PageNotFoundComponent** should be as shown below:



**~~~~~~~~~~~~~~ All the Best ~~~~~~~~~~~~**