**Integrated Assignment**

**Developing a client side application using Angular 8**

**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 **IntegratedAssignment\_toTrainee** in to your system
* **FBWebService** folder contains the **server side code**
* **FlightBookingUI** 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:**

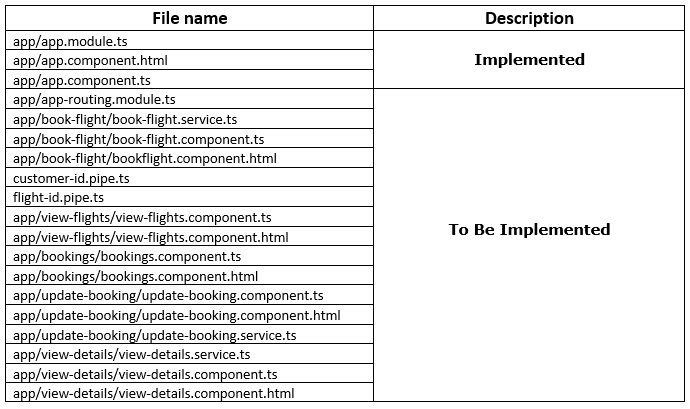
* The express application runs in port 105**0**
* Execute **npm install** command in both the **FBWebService** and **FlightBookingUI** folders
* Start the web service using appropriatecommand

**Problem Description:**

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

* Book a Flight
* View All Booking Details
* View Bookings based on Custom filters
* Cancel/Delete a particular Booking details

**Implementation details of FlightBookingUI** **folder:**

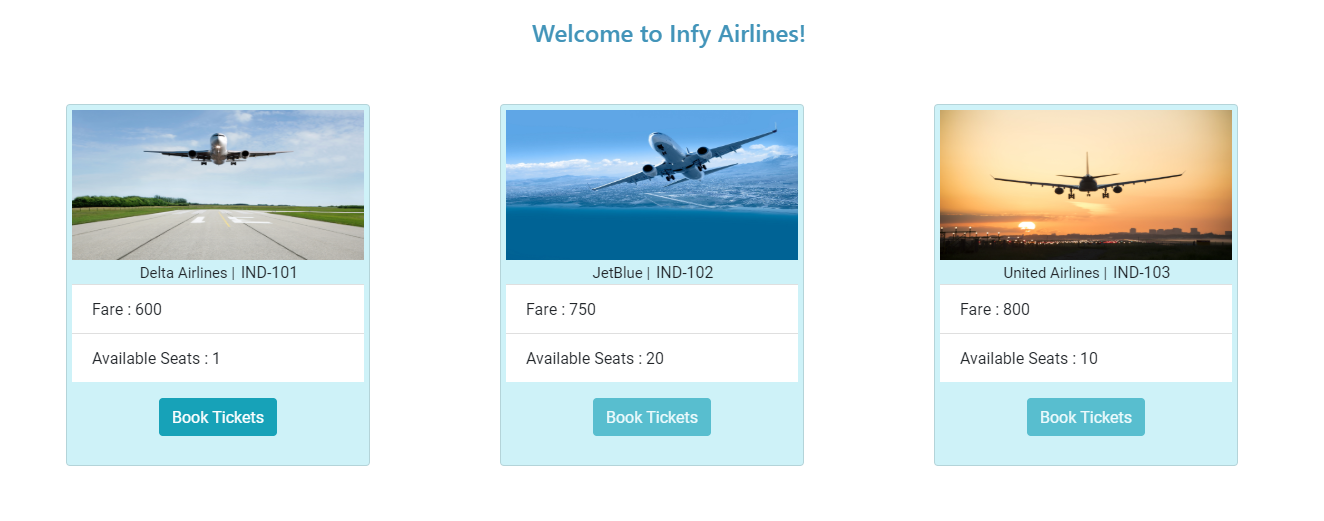


**view-flights/view-flights.component.ts**: – **(To be implemented)**

* **ViewFlightsComponent** class contains the following properties and methods:
  + flightDetails: FlightBooking[]
  + imageUrl: []
  + flightToEmit: EventEmitter
  + errorMessage : String
  + sendFlightData()
  + getAllFlights()
* Instance of **ViewDetailsService** should be injected into the constructor
* **getAllFlights(): (To Be Implemented)**
  + It should invoke **view()** of **ViewDetailsService**, which in turn returns an observable
  + It should populate **flightDetails** and **errorMessage** using the observable object
  + This method should be invoked on load of the component
* **sendFlightData(flight)**
  + It should take the selected flight object as the parameter.
  + It should use the flightToEmit EventEmitter instance to emit the same selected flight object
  + This method should be invoked on click of **Book Tickets** button

**view-flights/view-flights.component.html :- (To Be Implemented)**

When Book Flight link of navigation bar is clicked, the view should be displayed as shown below:



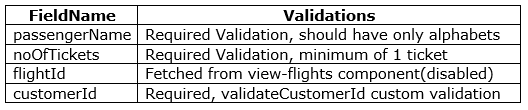
**Note: - Book Tickets** button should get **disabled** when the **availableSeats** becomes **0** for a particular flight or the status of the flight is **‘Cancelled’**

**app/book-flight/book-flight.service.ts: (To be implemented)**

* **bookFlight(formData):**
  + This method consumes the web service exposed at the URI.
  + It should send a POST request to the URI **http://localhost:1050/bookFlight** by passing **formData** data through body of request
  + It should return the same response as received from the webservice

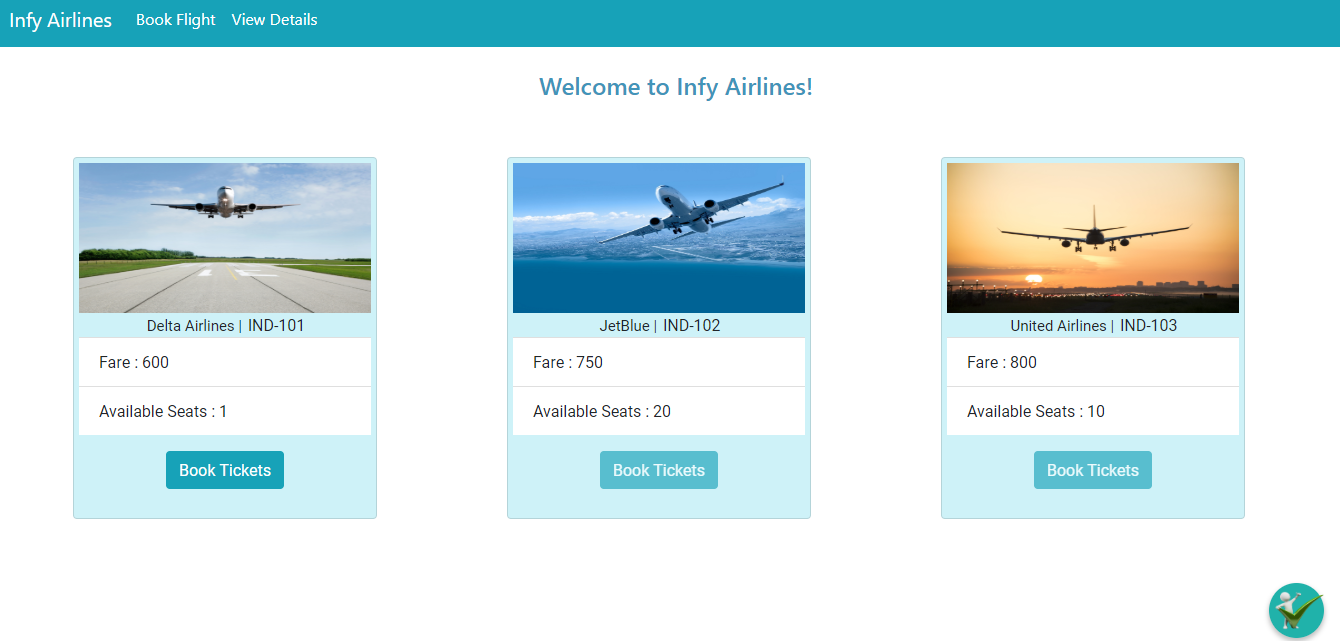
**app/book-flight/book-flight.component.ts: (To be implemented)**

* **BookFlightComponent** class contains the following properties and methods:
* errorMessage: string;
* successMessage: string;
* flightDetails
* bookingForm: FormGroup;
* book()
* showForm()
* This file also contains a function **validateCustomerId()** to validate the **customerId**
* Instances of **FormBuilder** and **BookFlightService** should be injected into the constructor
* **validateCustomerId(): (To Be Implemented)**
  + It should take customerId form control as a parameter and validate it.
  + If should validate if the **customerId** starts with an upper case alphabet followed by a 4-digit number (not starting with 0)
  + If the validation fails, it should return an object with **customerIdErr** property with value as an object containing **message** property as **“Enter a valid Customer Id (E.g., R1001)”**
* **book()** :- **(To be implemented)**
  + It should reset the **errorMessage** and **successMessage** to **null**
  + It should invoke **bookFlight()** of **BookFlightService** by passing the value of **bookingForm** object as a parameter, which in turn returns an observable
  + It should populate **successMessage** and **errorMessage** using the observable object
* **showForm(flight)** :- **(To be implemented)**
  + It should take selected flight object as a parameter and populate the same into **flightDetails** property
  + A FormGroup object named **bookingForm** should be created using the **FormBuilder** instance, having **passengerName**, **noOfTickets**, **flightId** and **customerId** as form controls (**flightId** should be assigned with the value from **flightDetails** object)
  + Add the validators for each form controls as mentioned below:



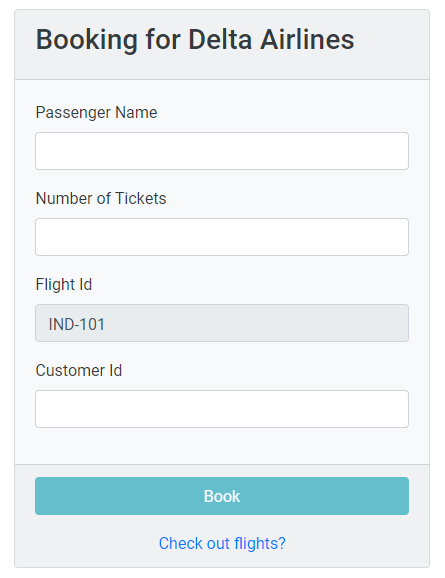
**app/app.component.html (To be modified)**

* Add the tag required to load the view of routed component
* When **“http://localhost:4200”** is entered in the browser, it should navigate to **/book** and the **BookFlightComponent** should be displayed as shown below:



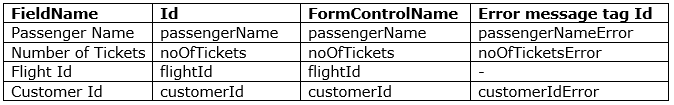
**app/book-flight/book-flight.component.html: - (To be implemented)**

* When the **BookFlightComponent** loads in browser, it should load the view of **ViewFlightsComponent** showing all the flights for booking as shown in the above screenshot
* When **Book Tickets** Button for any of the flight is clicked, it should display the **bookingForm** as shown below for the selected Flight as shown below: (**The below form is for flightId IND-101**)



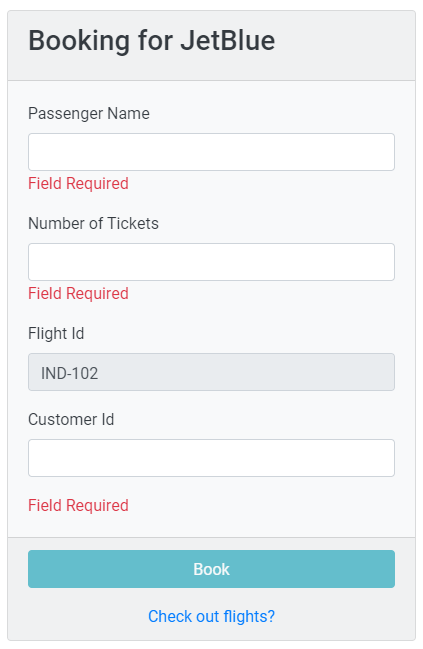
Note :

* The flightId field should be have the default value as the flightId of the selected flight for booking and should be **disabled**
* On click of the link **“Check out flights?”** should render all available flights from view-flights component and form should disappear.
* The following ids and form control names have to be added for each field:

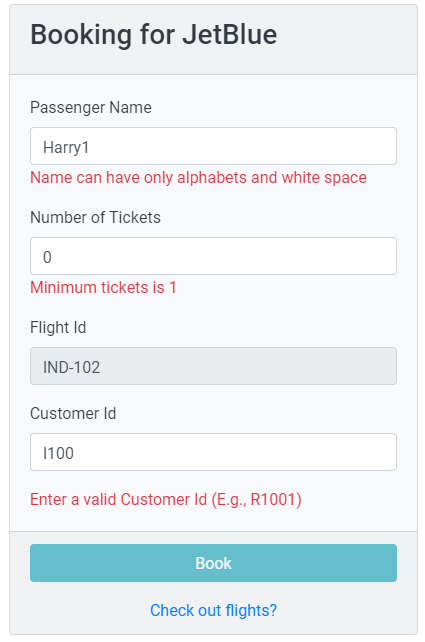


* Validation messages should be rendered only when the field becomes **dirty** and **validation fails** for a form control as shown below:

**Required Validations:**



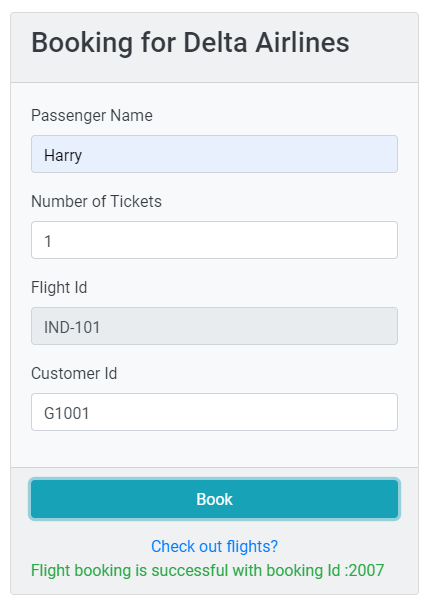
**Other Validations: -**



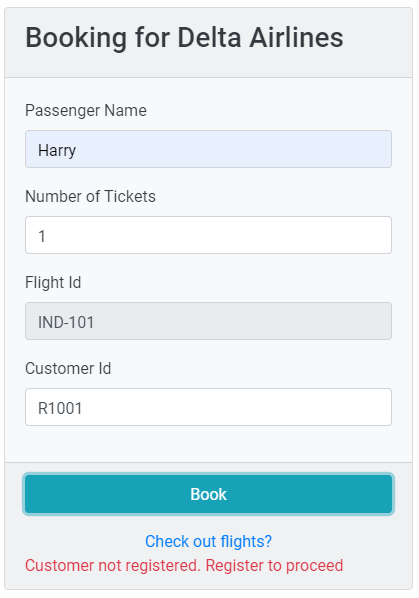
* All Fields (except **flightId** field) should have **Field Required** error message

**Note:**  If there is more than one validation message for a particular field, write the validation message in separate **span** tags and enclose them in common **div** tag and give **id** for that **div** tag

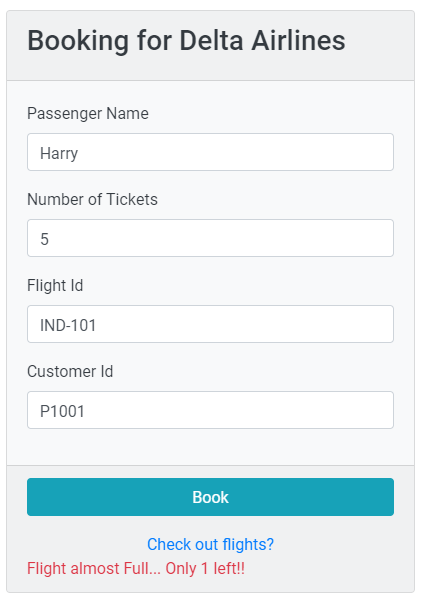
* **Book** Button should be **disabled** until the form becomes **valid**.
* **Id** for the Book Button is **book**
* When the form is submitted, **book()** method of **BookFlightComponent** should be invoked
* **Ids** for the error message and success message are **errorMessage** and **successMessage** respectively
* On successful booking, the success message should be displayed as shown below:



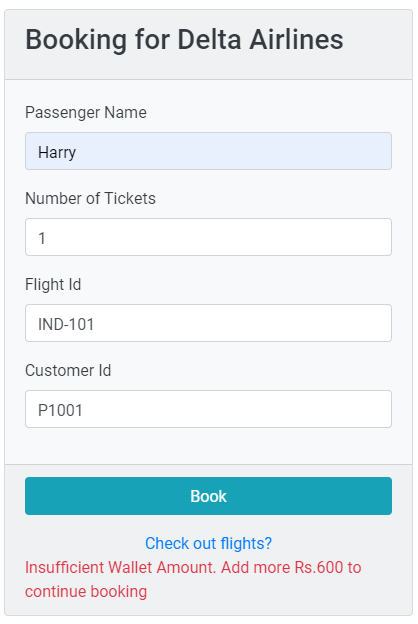
* In case if you have entered an invalid **customerId**, the error message should be displayed as shown below:



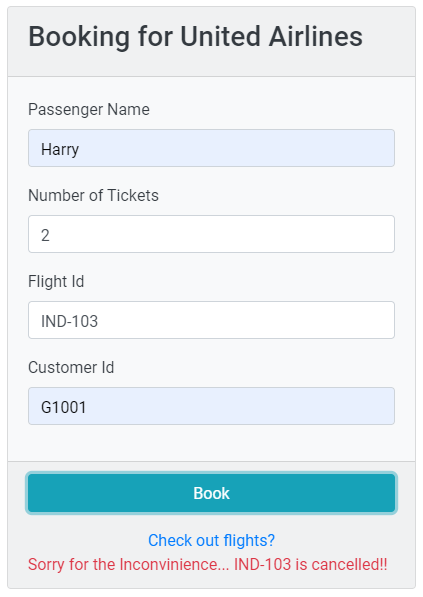
* In case if there are no enough number of seats available, the error message should be displayed as shown below:



* In case if there is no enough wallet amount available for a particular customer, the error message should be displayed as shown below:



* In case if the particular flight is cancelled, the error message should be displayed as shown below:



**app/view-details/view-details.service.ts (To be implemented)**

* This file contains two methods **view()** and **delete(bookingId).** The implementation details are as given below:
* **view()**
* It should send a **GET** request to the URI **http://localhost:1050/getAllBookings** which in turn returns an observable of type **FlightBooking** Array as a response
* Finally, the same response should be returned
* **delete(bookingId)**
* It should send a **DELETE** request to the URI **http://localhost:1050/delete/:bookingId** which in turn returns an Observable as a response
* Finally, the same response should be returned

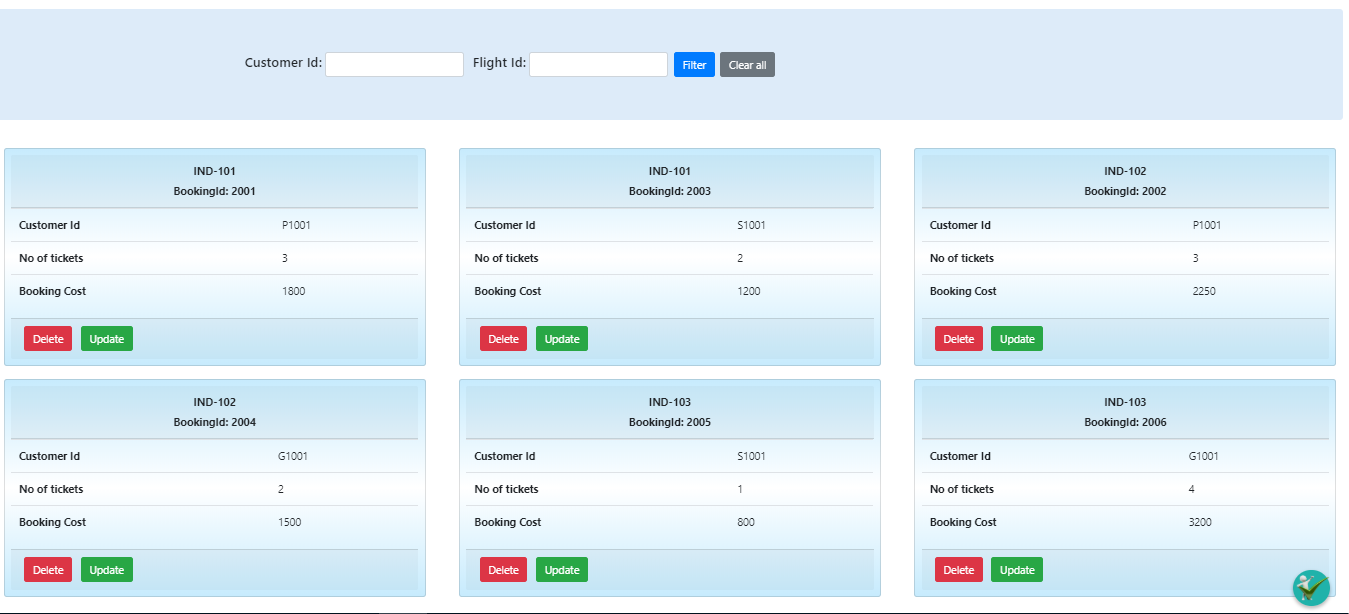
**view-details/view-details.component.ts (To be implemented)**

**ViewDetailsComponent** class contains the following properties and methods:

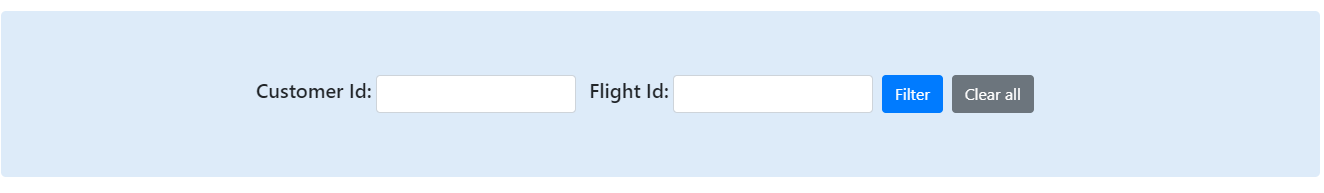
* bookings : Flights[]
* flightDetails: **FlightBooking[]**
* flights: **Flights[]**
* custId: string
* flightId: string
* errorMessage: string
* viewAllBookings()
* **viewAllBookings() :** (To be implemented)
* This method should not take any parameter and display all the bookings when the component loads in browser
* It should invoke **view()** method of **ViewDetailsService** which in turn returns an observable of type **FlightDetails** Array
* It should populate the **flightDetails** and **errorMessage** property appropriately with the returned observable in success and error cases respectively
* It should then iterate over the **bookings** array of each object in **flightDetails** array and populate corresponding **flightId** into each **booking** of **bookings** array
* This method should be invoked on load of the component

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

When View Details is clicked from the navigation bar, then the initial view of **view-details.component.html** view should be as shown below:



**Note:** For the input fields created as shown below, **D0 NOT** **use Forms**. Use only Two Way Binding should be used

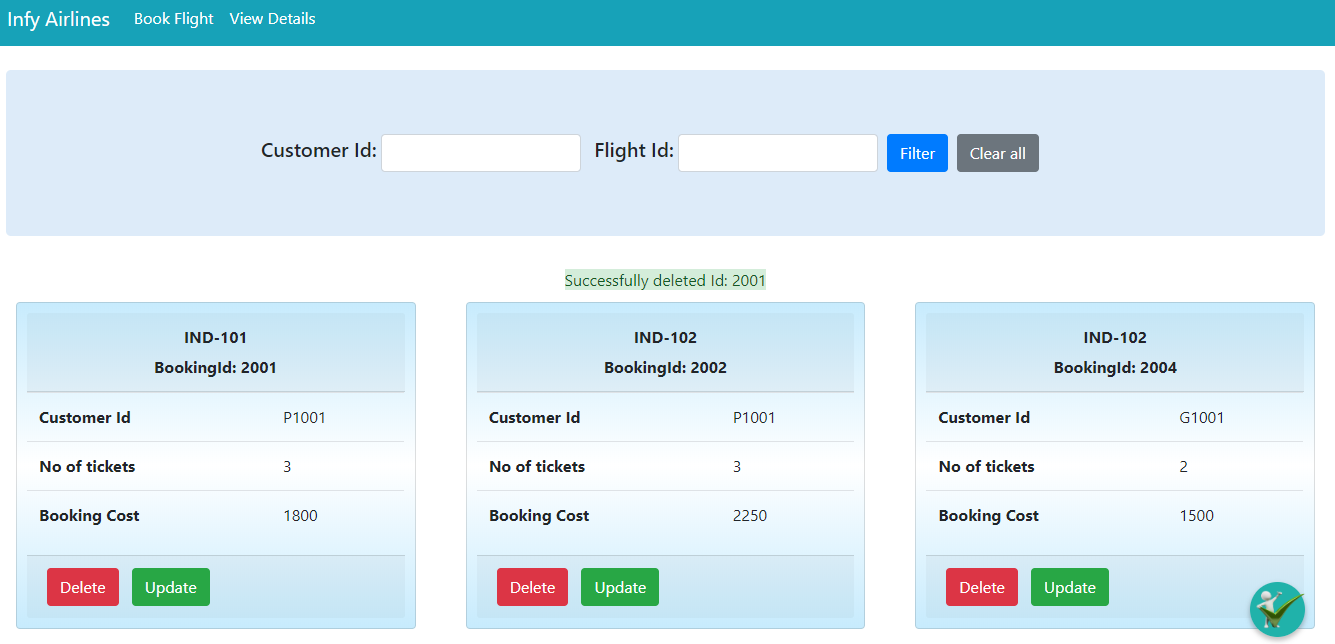


* Id for the Customer Id field is **custId** and it is a **text** input field
* Id for the Flight Id field is **fId** and it is a **text** input field
* When the user enters **customerId** or **FlightId** or **both**, and clicks on **Filter** button, booking details of that particular filter should be displayed.
* When the user clicks on **Clear all** button, the applied filter should be reverted back and should display all the bookings by nullifying **custId** and **flightId**.
* The **bookings** array should be passed to **BookingsComponent** (using parent to child component data transfer method) and stored with the property **bookingDetails** of **BookingsComponent** class.
* The custom pipes **customerId** and **flightId** should also be applied on the same bookings array (filtered bookings array should be transferred in case any filter is applied)

**app/bookings/bookings.component.ts: (To be implemented)**

* BookingsComponent class contains the following properties and methods:
* bookingDetails
* successMessage: string
* errorMessage: string
* delete()
* updateBooking()
* **delete(bookingId): (To Be Implemented)**
  + This method should take **bookingId** as a parameter and delete the booking corresponding to the given **bookingId**
* It should invoke the **delete()** method of **ViewDetailsService** by passing **bookingId** as a parameter which in turn returns an observable
* If successfully deleted, it should populate the **successMessage** property with the response message and remove the **deleted** **booking** object from the bookings array of **bookingDetails**
* If any error occurred, it should populate the **errorMessage** property with the error message of the response
* **updateBooking(booking): (To Be Implemented)**
  + It should take booking object as the parameter
  + It should navigate to **UpdateBookingComponent** by passing **bookingId** and **flightId** of the **booking** object through route parameter

**bookings/bookings.component.html : (To be implemented)**



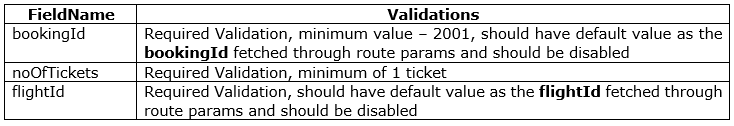
* The Delete button for each of the bookings should only be enabled if the cost of booking is less than **3000**. Else it should be disabled

**app/update-booking/update-booking.service.ts: To be implemented**

* This file contains one method **updateBooking()** for which the implementation details are as given below:
* **updateBooking(data)**
* It should send a PUT request to the URI **http://localhost:1050/updateBooking/:bookingId**, by passing **data** as parameter which in turn returns an observable as a response
* Finally, it should return the same Observable

**app/update-booking/update-booking.component.ts: To be implemented**

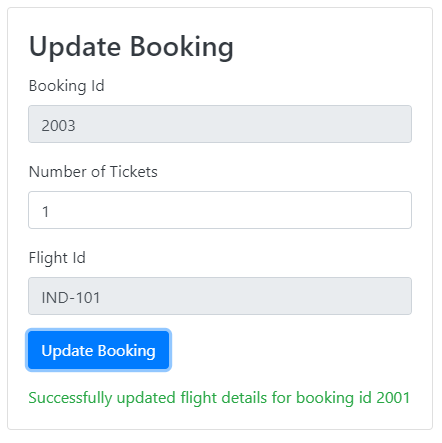
* **UpdateBookingComponent** class contains the following properties and methods:
* bookingId: number
* flightId: string
* updateBookingForm: Form Group
* successMessage: string
* errorMessage: string
* updateBooking()
* Instances of **FormBuilder**, **ActivatedRoute** and **UpdateBookingService** should 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:



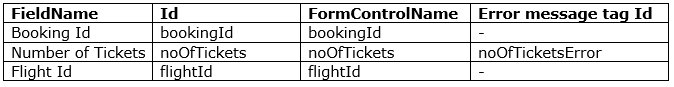
* **updateBooking()**
  + This method should not take any parameter and update the **bookingDetails** for the selected booking
  + It should reset the values of **successMessage** and **errorMessage** to **null**
  + It should invoke the **updateBooking()** method of **UpdateBookingService** by passing 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

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

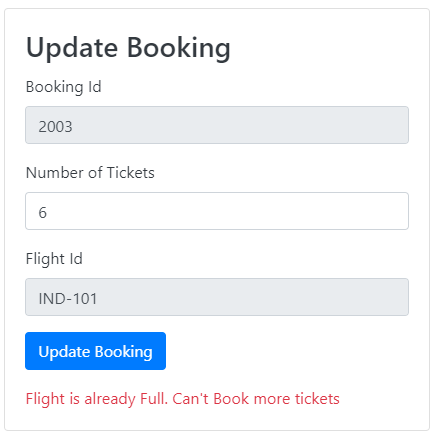
The view of **UpdateBookingComponent** is as shown below:



* The following id and form control name has to be added for each fields:



If the flight has no more seats available for booking, it should display the **errorMessage** as shown below:



**customer-id.pipe.ts : (To be implemented)**

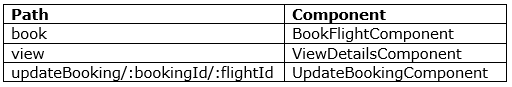
* This is a custom pipe that should be used to filter the bookings based on **customerId**
* It should take the **bookings** array and return the array containing **bookings** corresponding to that **bookingId**

**flight-id.pipe.ts : (To be implemented)**

* This is a custom pipe that should be used to filter the bookings based on **flightId**
* It should take the **bookings** array and return the array containing **bookings** corresponding to that **bookingId**

**app/app-routing.module.ts (To be implemented)**

* Configure the routes for the below information:



* In case an Invalid URL is provided, the user must be redirected to the FlightBooking **(/book)** page

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