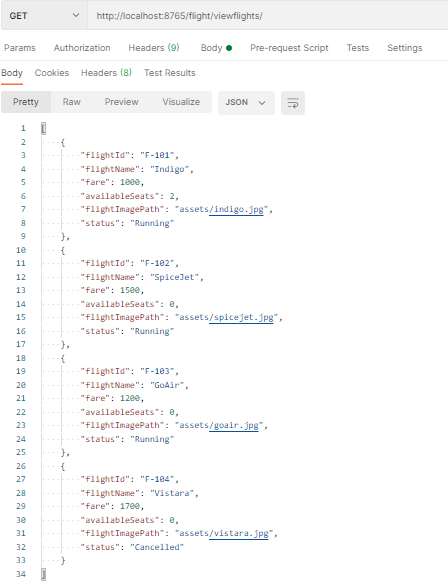
**Integrated Assignment**

**Developing a Single Page Application using React**

**Duration: 3Hrs Marks:50**

**Instructions to use the executable jar file provided:**

* Download the **“To\_Trainee”** folder and save it in your local system
* Inside **FlightBooking\_BackEnd** folder **“StartBackEnd.bat**”batch file is provided, double click the same to run the back-end web service.
* To ensure that the backend is running successfully, open Postman and verify the following Get URL:http://localhost:8765/flight/viewflights/
* Check below screenshot for the same
* 

**Instructions to use the FrontEnd folder provided:**

* Open **FlightBooking\_FrontEnd in** Visual studio code.
* Execute below **npm** command into **FlightBooking \_FrontEnd** project folder.
* **npm config set registry https://infyartifactory.ad.infosys.com/artifactory/api/npm/npm/**
* **npm install**

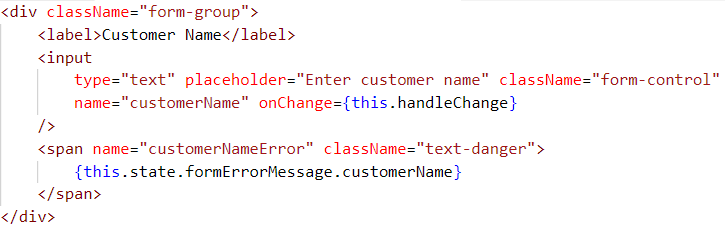
**Backup, Verification, and Submission guidelines:**

* Inside the **FlightBooking\_FrontEnd** folder “**StartVerification.bat”** batch file is provided, double click the same to verify your project.
* This will execute the jar file and opens **localhost:1729** in a web browser.
* Select the “**I Agree**” checkbox and then click on “**Verify”** button to Verify the projectand **“Copy Scripts to Desktop”** button **for back up**
* Enter your **employee id** correctly and click on “**Copy Scripts to Desktop**” button.
* This will create a folder on your desktop with the name as your **[employee id]** andcopies only **src**. Use this employee id folder to take **backup** and **Submission** from hands-on client.

**FrontEnd Coding Guidelines:**

* **DO NOT** alter the function name or the argument list of the function that is provided to you
* **DO NOT** add any new function.
* **DO NOT** write codes that result in infinite loops.

For Example:



Note: There should not be any space in the workspace folder name.

**Problem description**

**FlightBooking:** Infy Airlines wants to automate the process of flight ticket booking and management. The following functionalities are required to be implemented:

* View flights
* Book a flight

The **client side** needs to be developed, for interacting with the server side, which has been already implemented.

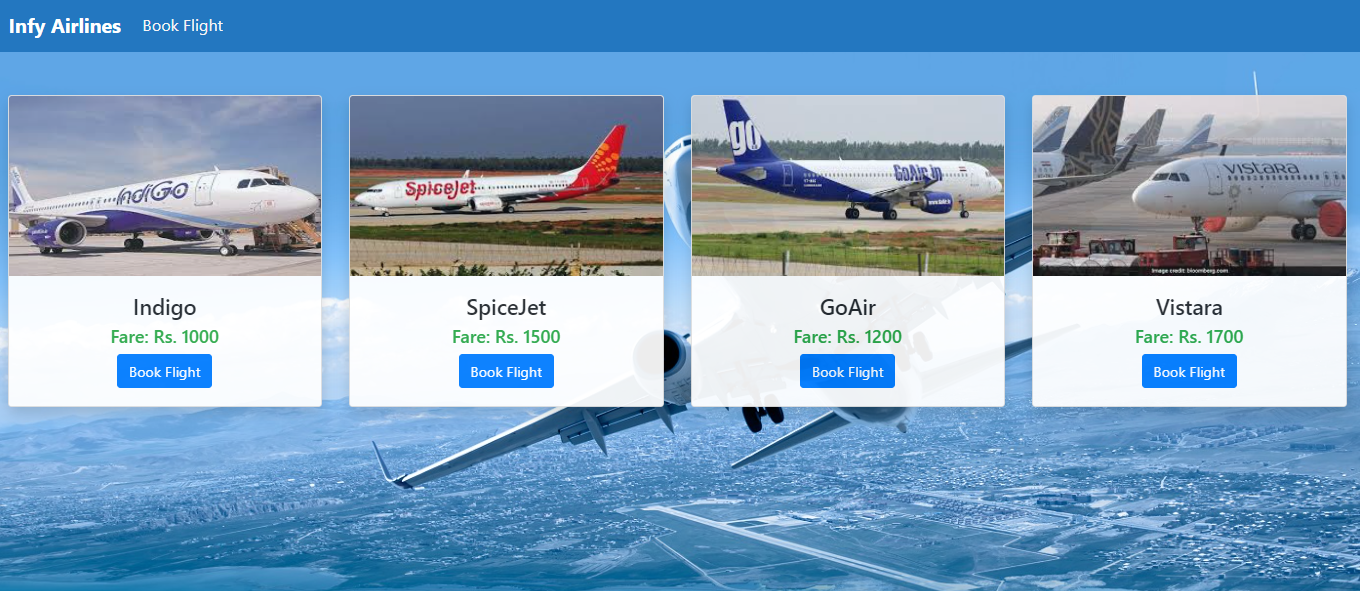


**src/App.js : (to be implemented)**

* Import all the necessary modules for implementing the routing
* Configure the following routes for your Infy Airlines Application:



When the application loads, the following view should be displayed:



**src/components/ViewFlights.js (to be implemented)**

The states of the component are defined as given below:

* **flightsArray :** It should store the flights data retrieved from the backend
* **selectedFlightName :** It should store the selected flightName
* **errorMessage:** It should store the error message
* **redirectStatus:** It should be set to false initially

The **methods** in **ViewFlights** component are given below:

* **fetchFlights() : (to be implemented)**
  + This method should send an **axios** **GET** request to the constant **url** in ViewFlights component
  + If the request is successful, it should set the **flightsArray** state with the response data
  + If the request fails, it should check if the error response is defined,
    - If the error response is defined, it should set **errorMessage** with the message of the error response data.
    - Else, it should set the **errorMessage** state as **“Please run the backend”.**

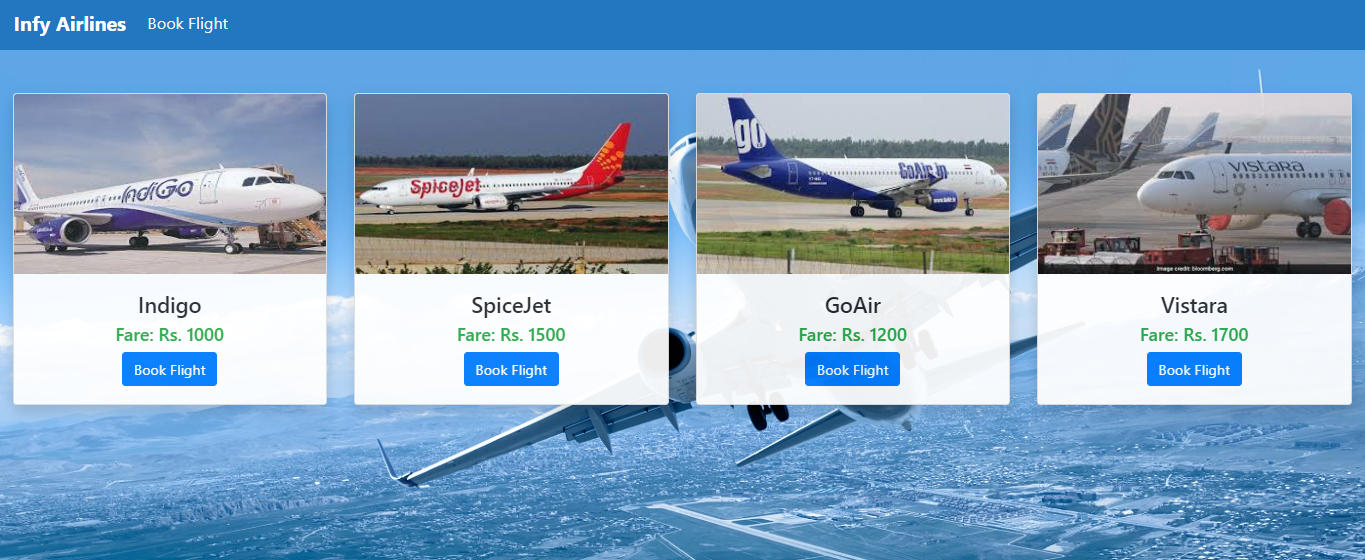
**Note: If the request is successful, the errorMessage (if any) should disappear and VICE VERSA**

**Note: Do not return any value from this method.**

* **render() : (to be implemented)**
  + if flightsArray has flight objects, it should invoke the displayFlights() method
  + else it should render the errorMessage

**JSX of ViewFlights component:**

* The JSX of the component should be coded to render the below view on load of the component:



* If the server is not running when the component gets loaded, it should display an error message as given below:



**src/components/CreateBooking.js : (to be implemented)**

All the **states** for the **CreateBooking** component are defined as given below:

* **formValue:** It should store the data entered by the user in the flight booking form
* **formErrorMessage:** It should store error message for each input field when validation fails for that input field
* **formValid:** It should store the validity of each input field
* **flightIds:** It should store the array of flightId’s
* **errorMessage:** It should store the error message
* **successMessage:** It should store the success message

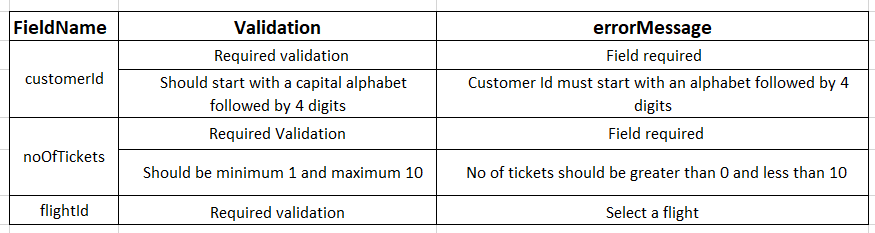
**Note:- Do not add any new states or alter the states given**

The **methods** in **CreateBooking** component are as given below:

* **handleChange() : (to be implemented)**
  + This method takes the event as a parameter and sets the value entered by the user to the corresponding state of **formValue** object
  + After setting the state value, it should invoke **validateField()** to check whether the entered value is valid according to the validations mentioned or not

**Note: Do not return any value from this method**

* **validateField() : (to be implemented)**
  + This method should take the name of the field and the corresponding value entered to validate it based on the given requirements
  + If the value entered is invalid, it should set an appropriate error message to the state of the corresponding field in **formErrorMessage** object and corresponding state in **formValid** object should be set as **false**
  + If valid input is entered, the corresponding state in **formValid** object should be set to **true** and the **errorMessage** (if any) for that field should disappear
  + Fields, validation, and their validation messages are as given below:



* + Note: The **Book Flight** button should be disabled until all the fields have valid input. The **buttonActive** state of formValid object should be changed as per requirements to either disable or enable the **Book Flight** button
  + While setting formErrorMessage and formValid objects to state, the previous states of successMessage and errorMessage should be cleared

**Note: Do not return any value from this method.**

* **handleSubmit() : (to be implemented)**
  + This method should take the event as a parameter
  + It should prevent the page reload on form submission and then invoke **submitBooking()**

**Note: Do not return any value from this method**

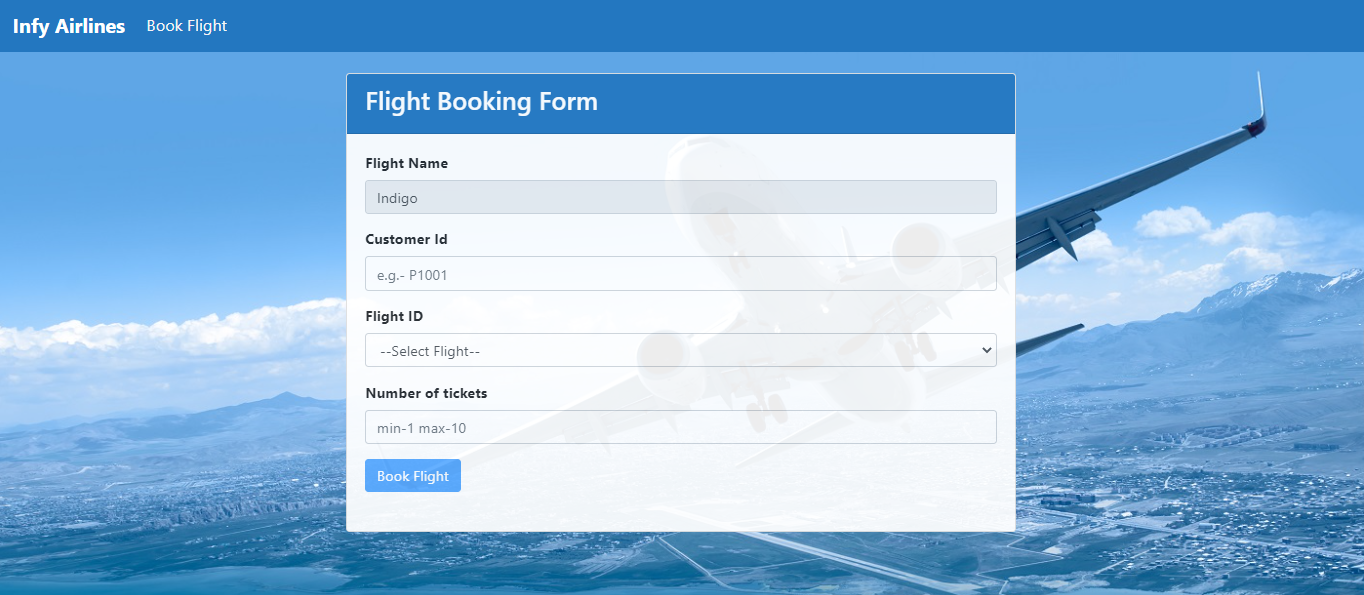
* **submitBooking() : (to be implemented)**
  + It should reset the **errorMessage** and **successMessage** states with an empty string.
  + It should send an **axios** **post** request to the constant **url** givenat top of the component bypassing **formValue** through the body of the request, which in turn returns a message on successful booking.
  + If the request is successful, it should set the **successMessage** state with the response data.
  + If the request fails, it should check if the error response is defined,
    - If the error response is defined, it should set **errorMessage** with the error response data.
    - Else, it should set the **errorMessage** state as **“Please run the backend”.**

**Note: If the request is successful, the errorMessage (if any) should disappear and VICE VERSA**

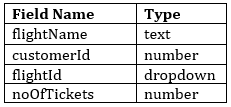
**Note: Do not return any value from this method.**

**JSX of CreateBooking component:**

The JSX of the component should be coded to render the below view



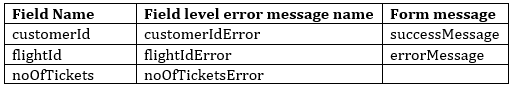
* **Placeholder** for the input fields should be provided as shown in the screenshot
* Form Fields and their type



**Note:**

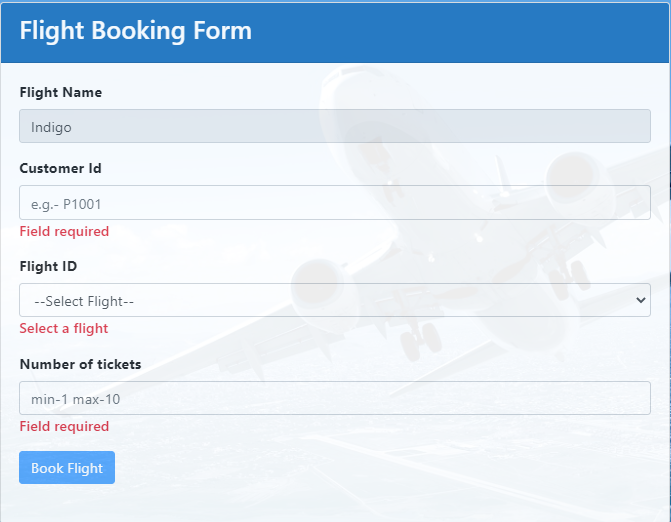
* The flightName field should be disabled by default
* The dropdown should be populated from the array of Flight id’s stored with the **flightIds** state
* **Book Flight** button should be disabled till valid inputs are given in all the input fields
* Once valid data is entered into the input fields, the error message of the field should disappear
* **handleChange()** method must be invoked every time the value changes in any of the input fields
* **USE APPROPRIATE BOOTSTRAP CLASSES WHEREVER NECESSARY**
* Use only **span** tag for displaying error messages wherever necessary
* Any error Message should not be displayed on the load of the component

**Do not forget to specify the value for name attributes in respective fields as given below : -**

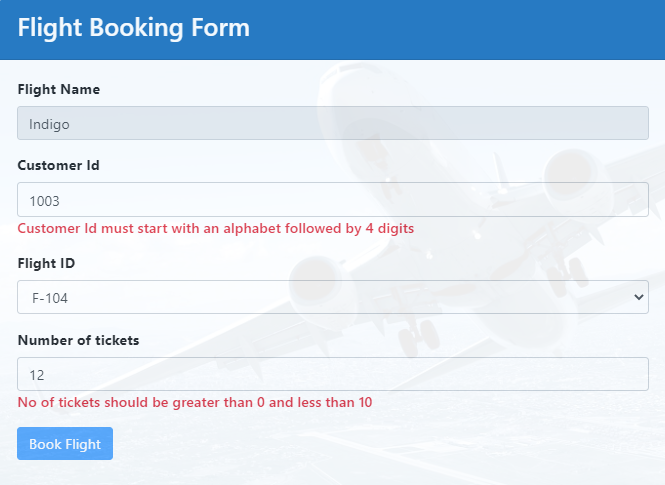


The error messages for the input fields should be displayed as shown below:

**Required Validation Error Messages:**



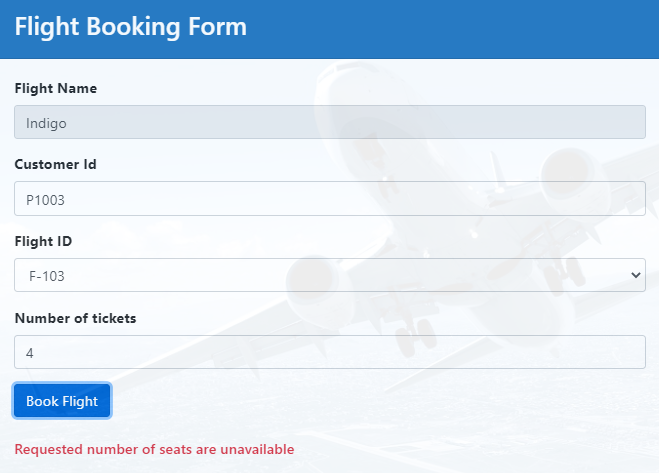
**Other Validation Error Messages:**



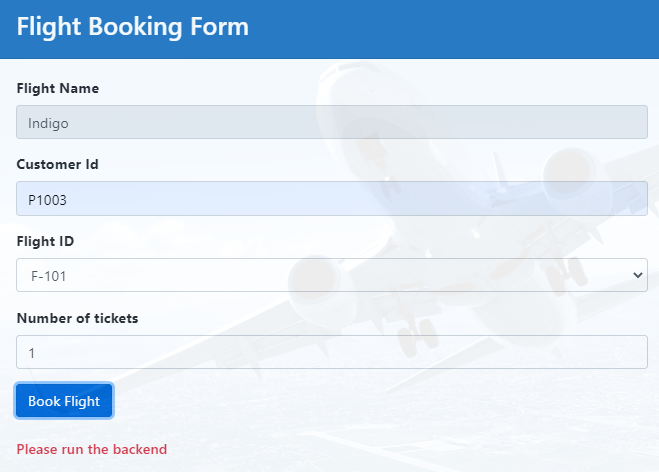
When valid inputs are given, the success message should be displayed as shown:



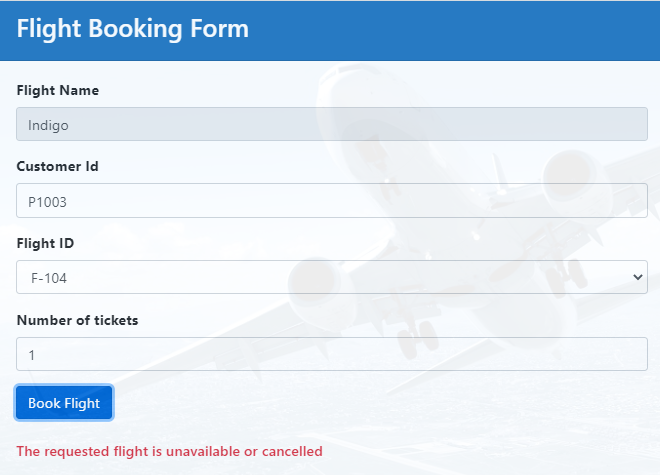
If the seats are not available to the selected flight, an error message should be displayed as shown below:



If the server is not running while the request is sent, the error message should be displayed as shown:



If the customer is trying to book for a flight which is **cancelled**, it should display the error message as shown below:



**~~~~~~~~~~~~~~~~~~~All The Best~~~~~~~~~~~~~~~~~~**