**Integrated Assignment**

**Developing a Single Page Application using React**

**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 **FlightBooking-toTrainee** in to your system and unzip it
* The unzipped folder contains two sub folders i.e. **flight-booking** and **FBWebService**
* **flight-booking** folder should contain the **client side code**
* **FBWebservice** foldercontains the **server side code**
* **DO NOT** alter the function name or the argument list of the function that is provided to you
* **DO NOT** add any new functions apart from the one given in the file where you write the solution
* **DO NOT** write codes that result in infinite loops/infinite recursive calls, because it just won’t work!
* Execute **npm install** in both **flight-booking** and **FBWebService** folders
* Execute **node app** in **FBWebService/src** folder to start your **webservice**

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

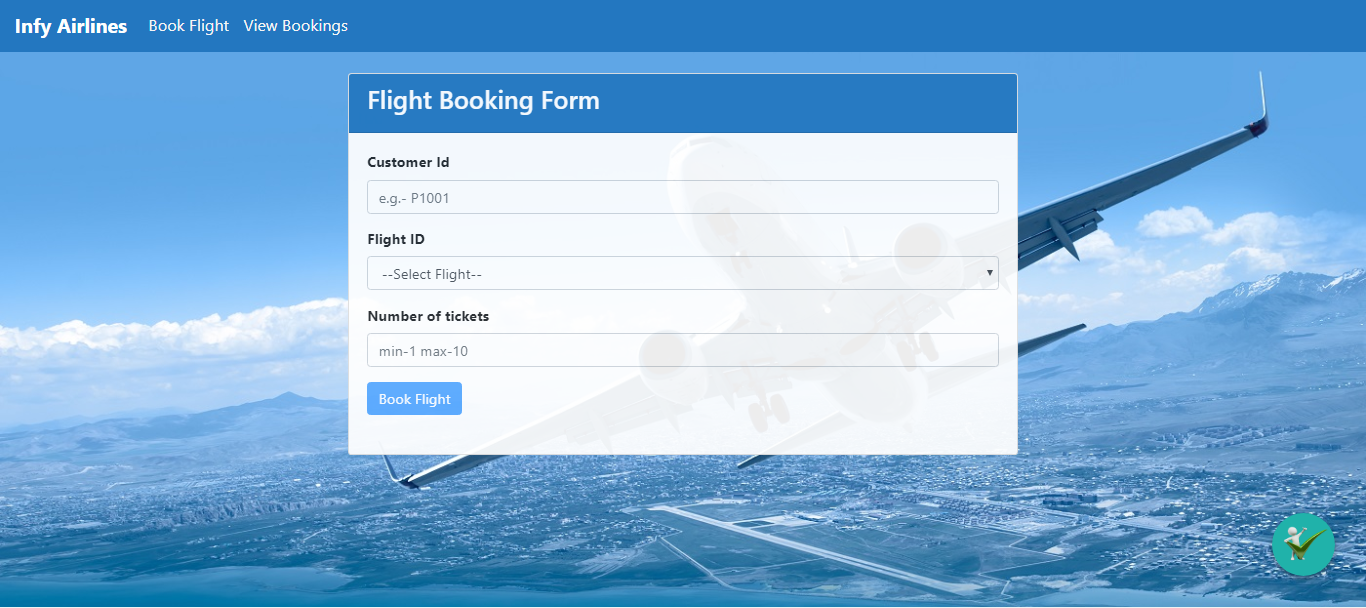
* Book a flight
* Getting data of all the bookings in all the flights
* Update booking for a particular booking based on bookingId
* Delete a particular booking based on bookingId

**Implementation details of flight-booking folder:**

|  |  |
| --- | --- |
| **File Name** | **Description** |
| src/components/CreateBooking.js | **Partially Implemented** |
| src/components/GetBookings.js |
| src/components/updateBooking.js |
| src/App.js |
| src/index.js | **Fully Implemented** |
| Public/index.html |

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

* Import all the necessary modules for implementing the routing
* Configure the following routes for your Infy Airlines Application:
  + **/bookFlight** that loads **CreateBooking** Component
  + **/viewBookings** that loads **GetBooking** component
  + **/updateBooking/:bookingId** that loads **updateBooking** component
* Configure another route to redirect to **CreateBooking** component in case of incorrect route path
* When the application loads we must the following view:

****

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

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

* **form:** It should store the data entered by the user in flight booking form
* **formErrorMessage:** It should store error message for each input field when validations fails
* **formValid:** It should store the validity of each input field
* **flightIds:** It should store the array of flightId’s obtained from the database
* **errorMessage:** It should store any error message which comes after axios request
* **successMessage:** It should store any success message which comes after axios request

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

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

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

* **ValidateField() : (to be implemented)**
  + This method should take name of the field and the corresponding value entered to validate it based on the given requirements.
  + If value entered is invalid, it should set appropriate errorMessage to the state of corresponding field in **formErrorMessage object**
  + Also, if the value entered in the field is invalid, the 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 given as below:

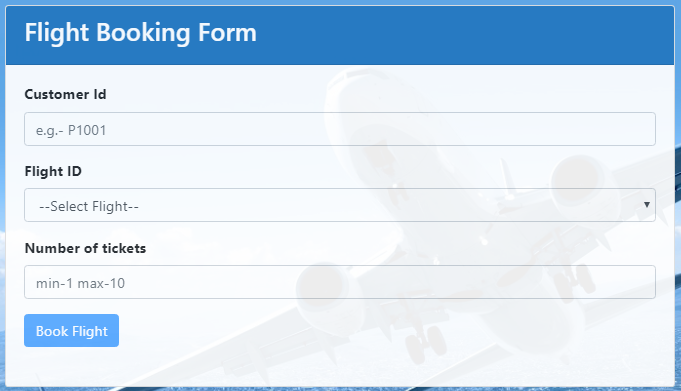
|  |  |  |
| --- | --- | --- |
| **FieldName** | **Validation** | **errorMessage** |
| customerId | 1. Required validation 2. Should start with a capital alphabet followed by 4 digits | 1. field required 2. Customer Id must start with an alphabet followed by 4 digits |
| noOfTickets | 1. Required Validation 2. Should be minimum 1 and maximum 10 | 1. Field required 2. No of tickets should be greater than 0 and less than 10 |
| flightId | 1. Required validation | 1. Select a Flight |

* + 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 disable and enable the **Book Flight** button
* **fetchFlightIds() : (to be implemented)**
  + It should reset the **flightIds** state with appropriate value
  + It should send an **axios** **GET** request to the URL **http://localhost:1050/getFlightIds**
  + If the request is successful, it should set the **flightIds** and **errorMessage** state appropriately
  + If the request fails, check for the error status,
    - If error status is **404**, it should set the **errorMessage** state with the message from error response data
    - Else, it should set the **errorMessage** state as **“Please start your Express server”**
  + **This method should be invoked on load of the component.**
* **submitBooking() : (to be implemented)**
  + It should send an **axios** **POST** request to the **URL http://localhost:1050/bookFlight/** with value of form state
  + If the request is successful, it should set the **successMessage** state from the response object
  + If the request fails, it should set the **errorMessage** from the error response object
  + **Note: If the request is successful, the errorMessage (if any) should disappear**
* **handleSubmit() : (to be implemented)**
  + This method should take event as a parameter
  + It should prevent the page reload on form submission and then invoke **submitBooking()** method

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

**JSX of CreateBooking component:**

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

****

* Form Fields and their type

|  |  |
| --- | --- |
| **Field Name** | **Type** |
| customerId | text |
| flightId | dropdown |
| noOfTickets | number |

**Note:**

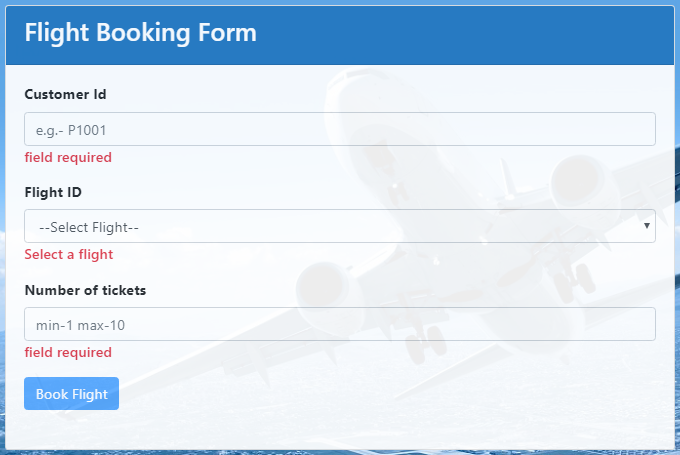
* The dropdown should be populated from the array of Flight id’s stored with the **FlightIds** state on load of the component
* **Placeholder** for the input fields should be as shown in the image
* **Book Flight** button should be disabled till valid inputs are given in all the input fields
* Once valid 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 WHERE EVER NECESSARY**
* Use only **span** tag for displaying error messages where ever necessary
* Any error Message should not be displayed on load of the component

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

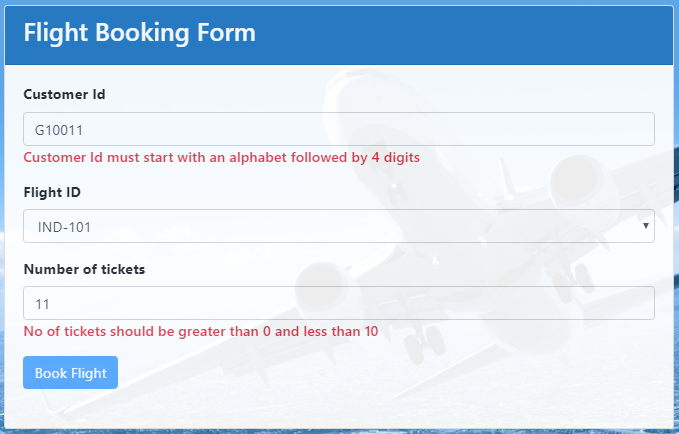
|  |  |  |
| --- | --- | --- |
| **fieldName** | **ErrorMessage Tag Name** | **formmessages** |
| customerId | customerIdError | successMessage |
| noOfTickets | noOfTicketsError | errorMessage |
| flightId | flightIdError |  |

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

**Required Validation Error Messages:**



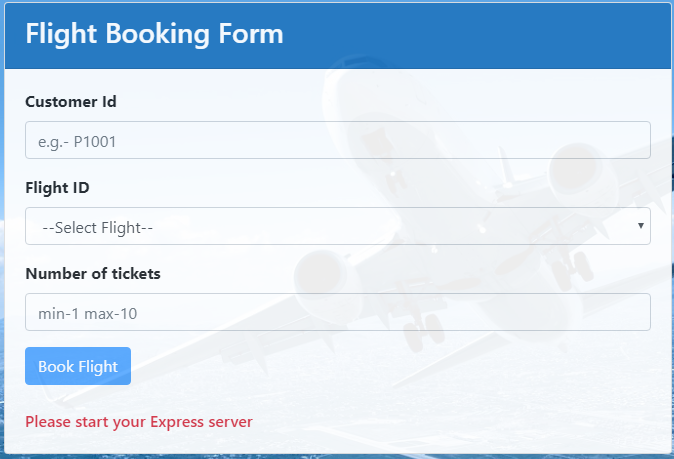
**Other Validation Error Messages:**

****

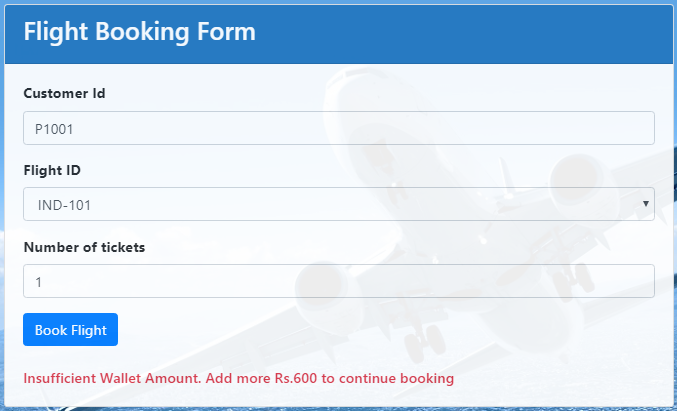
When valid inputs are given: The success message should be displayed as shown:



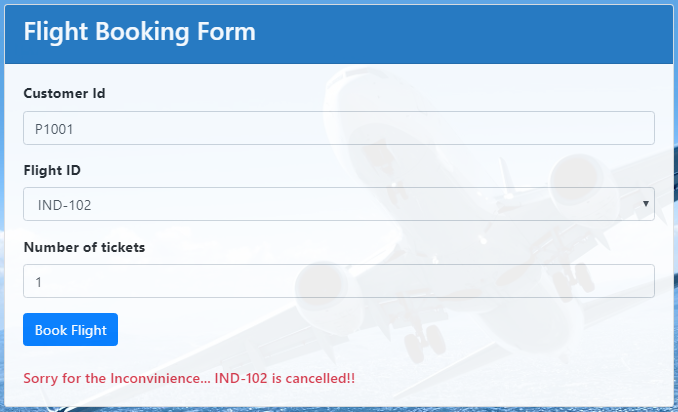
If the server is not running while request is sent, the error message should be displayed as shown:

****

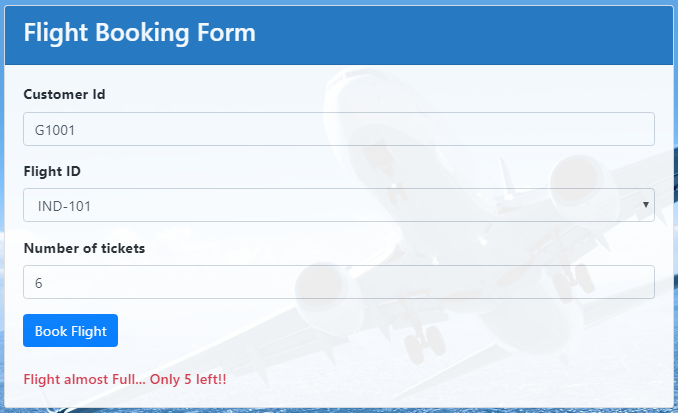
If the customer does not have sufficient wallet amount, it should display the error message as shown below:



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



If the customer tries to book number of tickets less than the seats available for booking, it should display the error message as shown below:



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

The states of the component are defined as given below:

* **bookingId:** It should store the bookingId of the booking selected by the user to update the booking
* **bookingData:** It should store the booking data retrieved for all the flights
* **updateStatus:** Its value should be updated to **true** on click of the Update button.
* **errorMessage:** It should store the error message if the request for booking detail fails
* **successMessage:** It should store the success message for any successful cancelation of booking

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

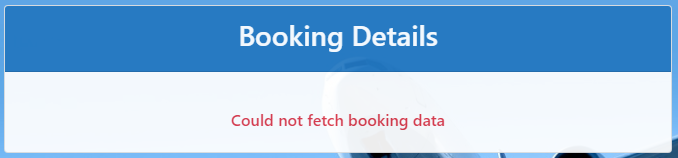
* **fetchBooking() : (to be implemented)**
  + This method should send an **axios** **GET** request to the URL **http://localhost:1050/getAllBookings**
  + If the request is successful, it should set the **bookingData** state with the response data
  + If the request fails, it should check for the error status,
    - If error status is **404**, it should set the **errorMessage** state with the message from error response data
    - Else, it should set the **errorMessage** state as **“Could not fetch booking data”**
  + This method should be invoked on load of the component
  + **Note: If the request is successful, the errorMessage (if any) should disappear and viceVersa**
* **updateBooking() : (to be implemented)**
  + This method takes **bookingId** as a parameter
  + This method should set the bookingId state with the bookingId of the selected booking for update.
  + Also it should set the **updateStatus** state to **true**
  + This method should be invoked on click of **Update** button
* **deleteBooking() : (to be implemented)**
  + This method should send an **axios DELETE** request to **http://localhost:1050/deleteBooking/:bookingId** to delete the details of the corresponding **bookingId**
  + If the request is successful, it should set the **successMessage** state with the response data.
  + It should then invoke **fetchBooking()** method to reload the available bookings
  + If the request fails, it should set the **errorMessage** state with the error response data
  + This method should be invoked on click of **Cancel** button as shown in the image

**JSX of GetBookings component:**

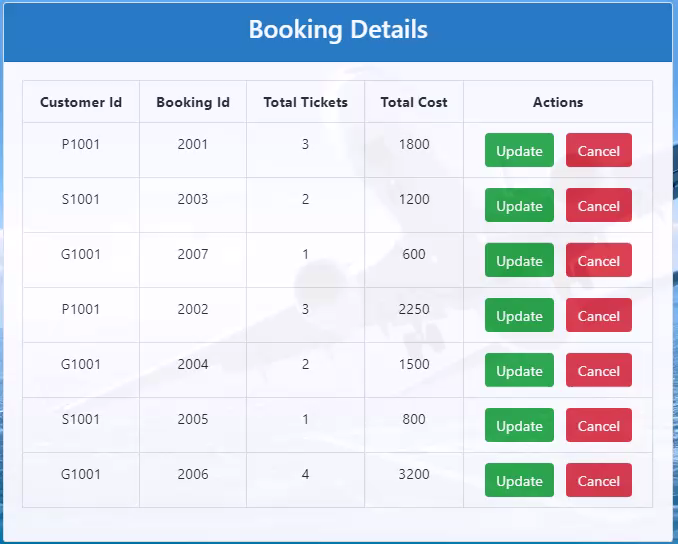
* The JSX of the component should be coded to render the below view on load of the component:

****

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



* On click of the **Cancel** button it should delete the selected record and reload the view as given below: **(for example: Cancel button for bookingId 2001 was clicked to get the below view)**

****

* On click of the **Update** button for a particular booking, it should redirect to the **updateBooking** component with the following view:

**Hint:-** Use the **updateStatus** state to render the view on click of **Update** Button



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

All the **states** for the **updateBooking** component are defined as given:

* **form:** It should store the data entered by the user in update form
* **formErrorMessage:** It should store error message for each input field when validations fails
* **formValid:** It should store the validity of each input field
* **errorMessage:** It should store any error message which comes after axios request
* **successMessage:** It should store any success message which comes after axios request
* **id:** It should store the bookingId obtained from the props for the selected booking for update

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

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

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

* **ValidateField() : (to be implemented)**
  + This method should take name of the field and the corresponding value entered to validate it based on the given requirements.
  + If value entered is invalid, it should set appropriate errorMessage to the state of corresponding field in **formErrorMessage object**
  + Also, if the value entered in the field is invalid, the 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 given as below:

|  |  |  |
| --- | --- | --- |
| **FieldName** | **Validation** | **errorMessage** |
| noOfTickets | 1. Required Validation 2. Should be minimum 1 and maximum 10 | 1. Field required 2. No of tickets should be greater than 0 and less than 10 |

* **updateBooking() : (to be implemented)**
  + It should reset the values of **successMessage** and **errorMessage** state appropriately
  + This method should send an **axios** **PUT** request to the URL [**http://localhost:1050/updatebooking/:bookingId**](http://localhost:1050/updatebooking/:bookingId)by passing the **bookingId** through route parameter along with noOfTickets as data with the request body
  + If the request is successful, it should set the **successMessage** state with the message from response data
  + If the request fails, it should set the **errorMessage** state with the message from error response data
  + **Note: If the request is successful, the errorMessage (if any) should disappear and viceVersa**
* **handleSubmit() : (to be implemented)**
  + This method should take event as a parameter
  + It should prevent the page reload on form submission and then invoke **updateBooking()** method

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

**JSX of updateBooking component:**

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

****

**Form Fields and their types:**

|  |  |
| --- | --- |
| **Field Name** | **type** |
| bookingId | number |
| noOfTickets | number |

**Note:**

* + The **bookingId** field should be disabled with the bookingId value of the booking for which update button was clicked
  + The **bookingId** state of form object should be populated with the value of **id** state on load of the component. Use appropriate life cycle method to set the state
* **Placeholder** for the input fields should be as shown in the image
* **Update Booking** button should be disabled till valid input is provided in noOfTickets field
* Once valid data is entered into the **noOfTickets** fields, the error message of the field should disappear
* **handleChange()** method must be invoked every time the value changes in the noOfTickets field
* **USE APPROPRIATE BOOTSTRAP CLASSES WHERE EVER NECESSARY**
* Use only **span** tag for displaying error messages where ever necessary

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

|  |  |  |
| --- | --- | --- |
| **fieldName** | **ErrorMessage Tag Name** | **Form messages** |
| bookingId | bookingIdError | SuccessMessage |
| noOfTickets | noOfTicketsError | ErrorMessage |

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

**Required Validation Error Messages:**



**Other Validation Error Messages:**



If the customer who has done the booking, does not sufficient wallet amount to update the booking for the required number of tickets, it should display the error message as shown below:



If the flight in which the customer has done booking has been cancelled, it should display the error message as shown below:



On successful update of the booking, the success response should be displayed as given below:



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