

Integrated Assignment

Developing a Single Page Application using React with Redux

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** folder contains 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 **web service**

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/index.js	Partially Implemented
src/utils/api.js	
src/actions/Bookings.js	
src/reducers/Bookings.js	
src/App.js	
src/components/CreateBooking.js	
src/components/GetBookings.js	
src/components/updateBooking.js	
src/reducers/index.js	Fully Implemented

public/index.html	
-------------------	--

src/index.js: (to be implemented)

- All the necessary modules should be imported.
- A redux store should be created using root reducer (created in index.js of reducers)
- ReactDOM.render() method should be modified appropriately to enable the use of redux store across components

src/utils/api.js: (to be implemented)

This file contains the following functions:

- **fetchFlightIdsAPI() : (to be implemented)**
 - This function should not take any parameter and send a request to fetch all the flightId's
 - It should send an **axios GET** request to the URL **http://localhost:1050/getFlightIds**
 - If the request is successful, it should return the flight Ids from data of response object
 - If the request fails, it should check for the error response,
 - If error response is defined, it should throw an error with the message from error response data
 - Else, it should throw an error with the message **"Please start your Express server"**
- **submitBookingAPI(formData) : (to be implemented)**
 - This function should take **formData** as a parameter and send a request for booking specified number of seats.
 - It should send an **axios POST** request to the URL **http://localhost:1050/bookFlight/** with value of **formData**
 - If the request is successful, it should return the message from the response data message
 - If the request fails, it should check for the error response,
 - If error response is defined, it should throw an error with the message from error response data
 - Else, it should throw an error with the message **"Please start your Express server"**
- **getAllBookingsAPI() : (to be implemented)**
 - This function should not take any parameter and send a request to fetch all the bookings across flights

- It should send an **axios GET** request to the URL **http://localhost:1050/getAllBookings**
 - If the request is successful, it should return the **booking data** from the response data
 - If the request fails, check for the error status,
 - If error status is **404**, it should throw an error with the error message as the message from error response data
 - Else, it should throw an error with the error message as **"Please start your Express server"**
- **deleteBookingAPI(bookingId) : (to be implemented)**
 - This function should take **bookingId** as a parameter and send a request to remove the booking corresponding to **bookingId** from the database
 - This function should send an **axios DELETE** request to **http://localhost:1050/deleteBooking/: bookingId** by passing the **bookingId** through route parameter, to delete the details of the corresponding **bookingId**
 - If the request is successful, it should return the message from the response data message
 - If the request fails, check for the error status,
 - If error status is **404**, it should throw an error with the error message as the message from error response data
 - Else, it should throw an error with the error message as **"Please start your Express server"**
- **updateBookingAPI(bookingId, formData) : (to be implemented)**
 - This function should take **bookingId** and **formData** as parameters and update the booking for the mentioned **bookingId**
 - This function should send an **axios PUT** request to the URL **http://localhost:1050/updatebooking/:bookingId** by passing the **bookingId** through route parameter along with **formData** as data with the request body
 - If the request is successful, it should return the message from the response data message
 - If the request fails, check for the error status,
 - If error status is **404**, it should throw an error with the error message as the message from error response data
 - Else, it should throw an error with the error message as **"Please start your Express server"**

src/reducers/Bookings.js: (to be implemented)

The **functions** in **Bookings.js** file are as given below:

- **FlightIds(): (to be implemented)**
 - It is a reducer which updates the state of **FlightIds** with the array of flightId's obtained from the database
 - This function takes **state = []** and **action** object as parameters
 - If the action type is **GET_ALL_FLIGHTIDS**, it should return **flightIds** array from **action** object
 - Else, it should return **state**
- **AllBookings(): (to be implemented)**
 - It is a reducer which updates the state of **AllBookings** with the array of booking data obtained from the database
 - This function takes **state = null** and **action** object as parameters
 - If the action type is **GET_ALL_BOOKINGS** then it should return value in **flightIds** of **action** object
 - Else, it should return **state**
- **Messages(): (to be implemented)**
 - It is a reducer which updates the state of **Messages** with the success and error messages received from the server after making API requests
 - This function takes **state = {successMessage: "", errorMessage: ""}** and **action** object as parameters
 - If the action type is **SET_MESSAGE**, it should return object as **{successMessage:<<successMessage>>, errorMessage:<<errorMessage>>}**. successMessage and errorMessage can be obtained from the **action** object
 - Else, it should return the **state**

src/reducers/index.js: (Implemented)

- This function imports all the reducers from the **src/reducers/Booking.js** file and export **combineReducers** method passing all imported the reducers to the method

src/actions/Bookings.js: (to be implemented)

This file contains the functions (action creators, actions, action dispatchers) as given below:

- **setMessageAction(): (to be implemented)**
 - This function is an action creator that takes **successMessage** and **errorMessage** as parameters and returns an object (Action) with the **type** as **SET_MESSAGE** and the **payload** as **successMessage** and **errorMessage**
 - This action creator is used to set the state of **Messages**

- **getBookingsAction(): (to be implemented)**
 - This function is an action creator takes **bookings** array as parameters and returns an object (Action) with the **type** as **GET_ALL_BOOKINGS** and the **payload** as **bookings** array
 - This action creator is used to set the state of **AllBookings**
- **fetchFlightIdsAction(): (to be implemented)**
 - This function is an action creator takes **flightIds** array as parameters and returns an object (Action) with the **type** as **GET_ALL_FLIGHTIDS** and the **payload** as **flightIds** array
 - This action creator is used to set the state of **FlightIds**
- **handleGetBookings() : (to be implemented)**
 - This function is an action dispatcher which takes **dispatch** and **bookingId** as parameter
 - It should invoke **getAllBookingsAPI()** function of **utils/api.js** file which in turn returns **bookings** array
 - If the request is successful, it should dispatch actions to set the state of **AllBookings** to **bookings** array and success & error messages of **Messages** to empty strings using appropriate action creators
 - If the request fails, it should dispatch an action to set the state of errorMessages in **Messages** of store to error message and state of **AllBookings** to **[]** using appropriate action creators
- **handleDeleteBooking() : (to be implemented)**
 - This function is an action dispatcher which takes **dispatch** and **bookingId** as parameter
 - It should invoke **deleteBookingAPI()** function of **utils/api.js** file by passing **bookingId** as a parameter, which in turn returns the **successMessage**
 - If the request is successful,
 - it should dispatch an action to set the state of successMessage in **Messages** of store to the response received using appropriate action creators
 - It should also invoke **handleGetBookings** by passing **dispatch** as a parameter
 - If the request fails, it should dispatch an action to set the state of errorMessage in **Messages** of store to error message using appropriate action creators
- **handleUpdateBooking() : (to be implemented)**
 - This function is an action dispatcher which takes **dispatch**, **bookingId** and **formData** as parameters

- It should invoke **updateBookingAPI()** function of **utils/api.js** file by passing **bookingId** and **formData** as a parameter, which in turn returns the **successMessage**
- If the request is successful, it should dispatch an action to set the state of **successMessage** in **Messages** of store to the response received using appropriate action creators
- If the request fails, it should dispatch an action to set the state of **errorMessages** in **Messages** of store to error message using appropriate action creators
- **handleFetchFlightIds() : (to be implemented)**
 - This function is an action dispatcher which takes **dispatch** as parameter
 - It should invoke **fetchFlightIdsAPI()** function of **utils/api.js** file by passing **bookingId** and **formData** as a parameter, which in turn returns array of **flightIds**
 - If the request is successful, it should dispatch an action to set the state of **FlightIds** to **flightIds** array and **success** and **error** messages of **Messages** to empty strings using appropriate action creators
 - If the request fails, it should dispatch an action to set the state of **errorMessage** in **Messages** of store to error message and state of **FlightIds** to **[]** using appropriate action creators
- **handleSubmitBooking() : (to be implemented)**
 - This function is an action dispatcher which takes **dispatch** and **formData** as parameters
 - It should invoke **submitBookingAPI()** function of **utils/api.js** file by passing **formData** as a parameter, which in turn returns **successMessage**
 - If the request is successful, it should dispatch an action to set the state of **successMessage** in **Messages** of store to the response received using appropriate action creators
 - If the request fails it should dispatch an action to set the state of **errorMessages** in **Messages** of store to error message using appropriate action creators

Note: - Both success and error message should not be displayed at the same time

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

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 invoke the **handleFetchFlightIds()** of **action/Booking.js** file by passing the **dispatch** function of props as parameter
- **This method should be invoked on load of the component**

- **submitBooking() : (to be implemented)**

- It should invoke the **handleSubmitBooking()** of **action/Booking.js** file passing the **dispatch** function of props and the form object in state as parameters
- This method should be invoked on form submission

- **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:

A screenshot of a web form titled "Flight Booking Form" with a blue header. The form contains three input fields: "Customer Id" with a placeholder "e.g.- P1001", "Flight ID" with a dropdown menu showing "--Select Flight--", and "Number of tickets" with a placeholder "min-1 max-10". A blue "Book Flight" button is located at the bottom left of the form. The background of the form has a faint image of an airplane.

- Form Fields and their type

Field Name	Type
customerId	text
flightId	Dropdown
noOfTickets	Number

Note:

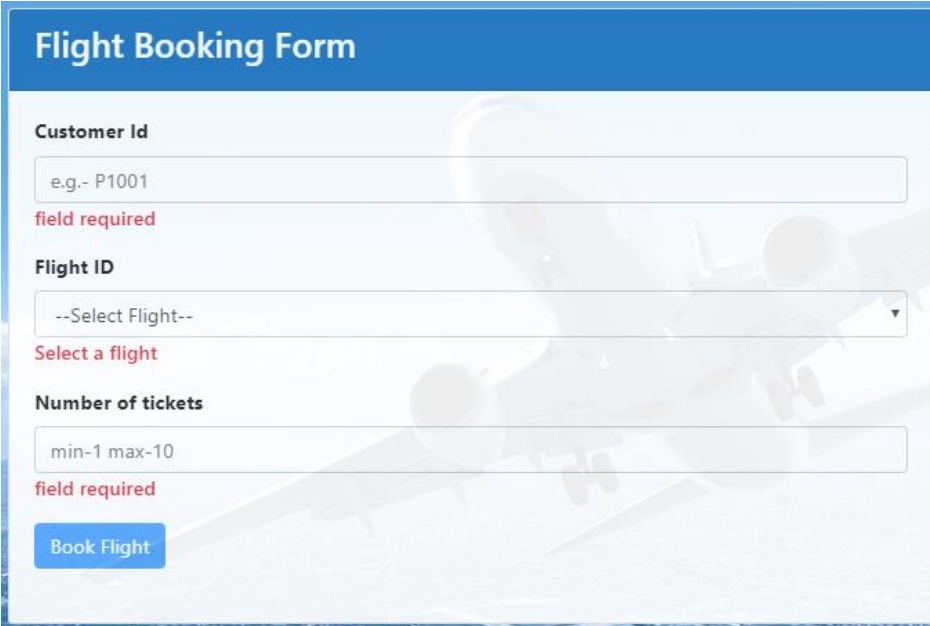
- The dropdown should be populated from the array of Flight id's stored with the **FlightIds** state returned from redux store on load of the component
- Placeholder** for the input fields should be as shown in the image
- Book Flight** button should be disabled till valid inputs are given in all the 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 WHEREVER NECESSARY**
- Use only **span** tag for displaying error messages wherever necessary
- Any error Message should not be displayed on load of the component
- On unmounting the component set the **successMessage** and **errorMessage** of **Messages** store to **empty string("")** by dispatching an action in this file
- Don't create any additional keys in state object
- Use **Messages** store to save and display error and success messages of API requests

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	

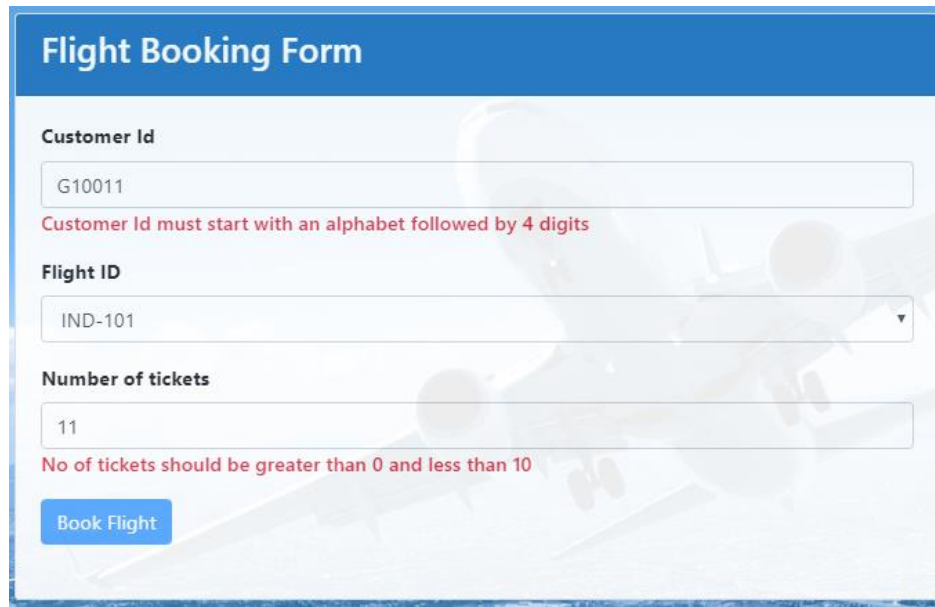
The error messages for the input fields should be displayed as shown below:

Required Validation Error Messages:



The screenshot shows a 'Flight Booking Form' with a blue header. It contains three input fields: 'Customer Id' with a text input containing 'e.g.- P1001' and a red 'field required' error message below it; 'Flight ID' with a dropdown menu showing '--Select Flight--' and a red 'Select a flight' error message below it; and 'Number of tickets' with a text input containing 'min-1 max-10' and a red 'field required' error message below it. A blue 'Book Flight' button is at the bottom left. The background of the form has a faint image of an airplane.

Other Validation Error Messages:



Flight Booking Form

Customer Id

Customer Id must start with an alphabet followed by 4 digits

Flight ID

Number of tickets

No of tickets should be greater than 0 and less than 10

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



Flight Booking Form

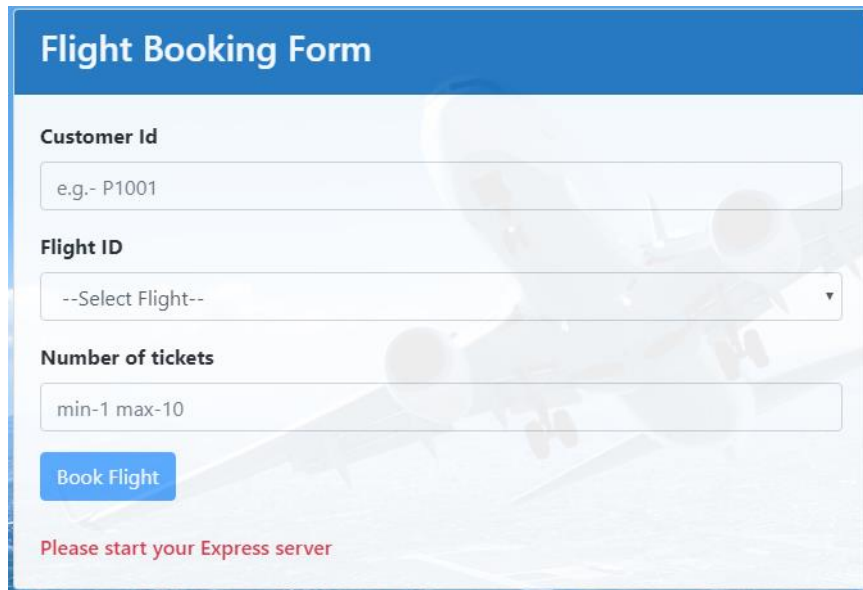
Customer Id

Flight ID

Number of tickets

Flight booking is successful with booking Id :2007

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



The image shows a web form titled "Flight Booking Form" with a blue header. The form has three input fields: "Customer Id" with the placeholder text "e.g.- P1001", "Flight ID" with a dropdown menu showing "--Select Flight--", and "Number of tickets" with the range "min-1 max-10". A blue "Book Flight" button is located below the inputs. At the bottom, a red error message reads "Please start your Express server". The background of the form features a faint image of an airplane.

Flight Booking Form

Customer Id

e.g.- P1001

Flight ID

--Select Flight--

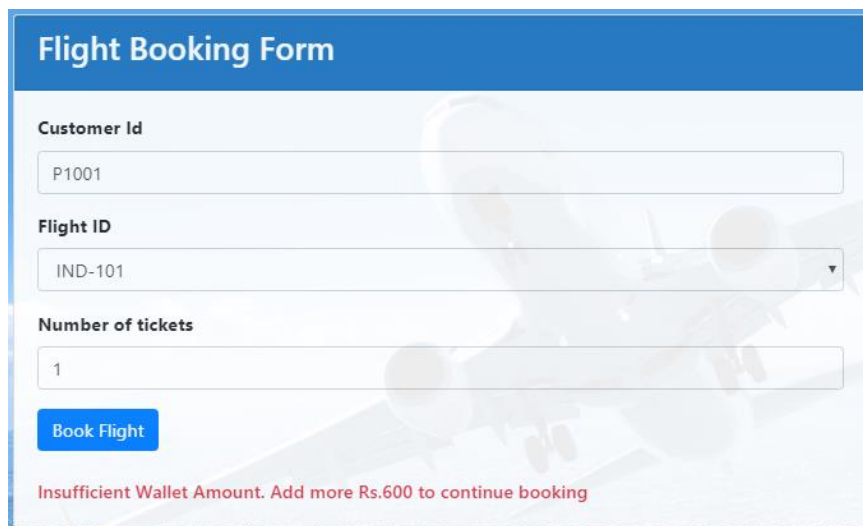
Number of tickets

min-1 max-10

Book Flight

Please start your Express server

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



This image shows the same "Flight Booking Form" as above, but with different input values and an error message. The "Customer Id" field contains "P1001", the "Flight ID" dropdown shows "IND-101", and the "Number of tickets" field contains "1". The "Book Flight" button is still present. A red error message at the bottom states "Insufficient Wallet Amount. Add more Rs.600 to continue booking". The background image of an airplane remains the same.

Flight Booking Form

Customer Id

P1001

Flight ID

IND-101

Number of tickets

1

Book Flight

Insufficient Wallet Amount. Add more Rs.600 to continue booking

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



Flight Booking Form

Customer Id

Flight ID

Number of tickets

Book Flight

Sorry for the Inconvinience... IND-102 is cancelled!!

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:



Flight Booking Form

Customer Id

Flight ID

Number of tickets

Book Flight

Flight almost Full... Only 5 left!!

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

- **updateStatus:** Its value should be updated to **true** on click of the Update button.

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

- **fetchBooking() : (to be implemented)**
 - It should invoke the **handleGetBookings()** of **action/Booking.js** file passing the **dispatch** function of props as parameter
 - **This method should be invoked on load of the component**

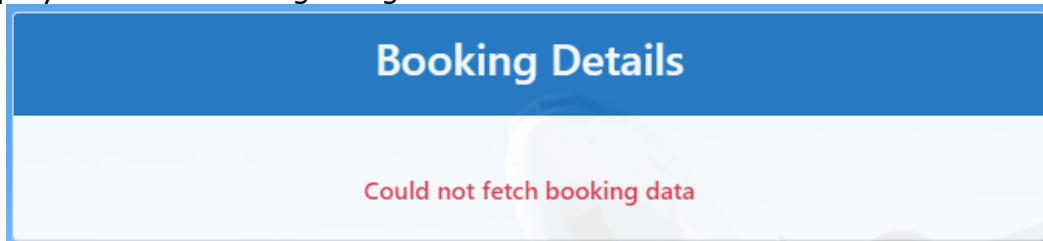
Note: If the request is successful, the errorMessage (if any) should disappear and viceVersa
- **updateBooking() : (to be implemented)**
 - This method takes **bookingId** as a parameter
 - This method should set the **bookingId** state with the bookingId of the selected booking for update.
 - Also it should set the **updateStatus** state to **true**
 - This method should be invoked on click of **Update** button
- **deleteBooking() : (to be implemented)**
 - This function should take **dispatch** and **id** as parameter
 - It should invoke the **handleDeleteBooking()** of **action/Booking.js** file by passing the **dispatch** function of props and **id** as parameters
 - 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:

Booking Details				
Customer Id	Booking Id	Total Tickets	Total Cost	Actions
P1001	2001	3	1800	<button>Update</button> <button>Cancel</button>
S1001	2003	2	1200	<button>Update</button> <button>Cancel</button>
G1001	2007	2	1200	<button>Update</button> <button>Cancel</button>
P1001	2002	3	2250	<button>Update</button> <button>Cancel</button>
G1001	2004	2	1500	<button>Update</button> <button>Cancel</button>
S1001	2005	1	800	<button>Update</button> <button>Cancel</button>
G1001	2006	4	3200	<button>Update</button> <button>Cancel</button>

- 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)**

Booking Details				
Customer Id	Booking Id	Total Tickets	Total Cost	Actions
P1001	2001	3	1800	<div>Update</div> <div>Cancel</div>
S1001	2003	2	1200	<div>Update</div> <div>Cancel</div>
G1001	2007	1	600	<div>Update</div> <div>Cancel</div>
P1001	2002	3	2250	<div>Update</div> <div>Cancel</div>
G1001	2004	2	1500	<div>Update</div> <div>Cancel</div>
S1001	2005	1	800	<div>Update</div> <div>Cancel</div>
G1001	2006	4	3200	<div>Update</div> <div>Cancel</div>

- 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

Update Flight Booking for id: 2001

bookingId

Number of Tickets

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
- **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 invoke the **handleUpdateBooking()** of **action/Booking.js** file by passing the **dispatch** function of props, **bookingId** and **form** object of state as parameters
 - **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

- On unmounting the component set the **successMessage** and **errorMessage** of **Messages** store to **empty string("")** by dispatching an action in this file in both **UpdateBooking** and **GetBookings** components
- Don't create any additional keys in state object
- Use **Messages** store to save and display error and success messages of API requests in both **UpdateBooking** and **GetBookings** components

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:



The screenshot shows a form titled "Update Flight Booking for id: 2001". It has two input fields: "bookingId" with the value "2001" and "Number of Tickets" with the value "min-1 max-10". Below the "Number of Tickets" field, there is a red error message "field required". At the bottom, there is a blue "Update Booking" button.

Other Validation Error Messages:



The screenshot shows the same form titled "Update Flight Booking for id: 2001". The "bookingId" field has the value "2001" and the "Number of Tickets" field has the value "11". Below the "Number of Tickets" field, there is a red error message "No of tickets should be greater than 0 and less than 10". At the bottom, there is a blue "Update Booking" button.

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:



The screenshot shows a web form titled "Update Flight Booking for id: 2001". It contains two input fields: "bookingId" with the value "2001" and "Number of Tickets" with the value "1". Below these fields is a blue "Update Booking" button. At the bottom of the form, a red error message reads: "Insufficient Wallet Amount. Add more Rs.600 to continue booking".

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



The screenshot shows a web form titled "Update Flight Booking for id: 2002". It contains two input fields: "bookingId" with the value "2002" and "Number of Tickets" with the value "1". Below these fields is a blue "Update Booking" button. At the bottom of the form, a red error message reads: "Sorry for the Inconvinience... IND-102 has been cancelled!!".

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

Update Flight Booking for id: 2005

bookingId

2005

Number of Tickets

1

Update Booking

Booking successfully updated in IND-103

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