**TekGain E-Learning Service Automation**

**UI as React Application**

TekGain is one of the popular training institutes that have branches all over India. Due to covid restrictions, TekGain planned to initiate E-Learning activities that help candidates all over the world to gain benefit. As an initiate they need developing an application that supports the below business activities.

* Course Management Service
* Associate Management Service
* Admission Management Service

**User Interface Business Requirements and Design Specification.**

UI is created using React Library. As part of the business requirements, effective and very interactive pages need to be implemented. To achieve this certain functionalities are done on the client-side application using ReactJS.

**Problem Specification**

Using this react application, have to create the UI for the below modules

Create a UI for the below modules using ReactJS.

1. Course Management
2. Associate Management
3. Admission Management

**React Component Specification**

Following react components need to be created.

* Routing Component
* App Component
* Course Component
* Admission Component
* Associate Component

**Routing Component**

**File: Routing.js**

Design the Routing component with the given routing details.

**Functional Requirement:**

* Implement **nested routing** for the Course, Admission and Associate components with its respective child components as given below.
* Routing for main components

|  |  |
| --- | --- |
| Router path | Route To |
| course | Course Component |
| admission | Admission Component |
| associate | Associate Component |

* Course component should have routing for the following sub components.

|  |  |
| --- | --- |
| Router path | Route To |
| addcourse | AddCourse Component |
| updatecourse | UpdateCourse Component |
| viewcourse | ViewCourse Component |
| viewfeedbackrating | CourseRating Component |
| deactivate | CourseDeactivate Component |

* Associate component should have routing for the following sub components.

|  |  |
| --- | --- |
| Router path | Route To |
| addassociate | AddAssociate Component |
| updateassociate | UpdateAssociate Component |
| viewassociate | ViewAssociate Component |

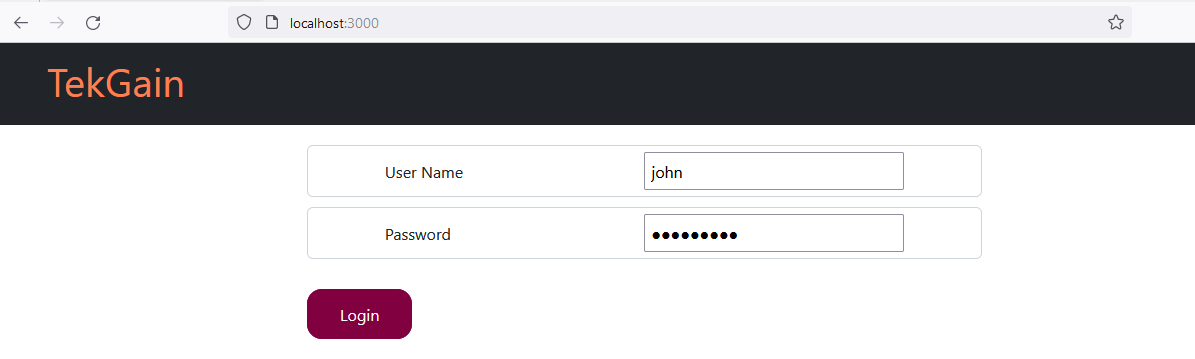
* Admission component should have routing for the following sub components.

|  |  |
| --- | --- |
| Router path | Route To |
| registration | AssociateRegistration Component |
| totalFee | TotalFees Component |
| addfeedback | AddFeedback Component |
| highestFee | HighestFeeCalculation Component |
| viewfeedback | ViewFeedback Component |
| makepayment | MakePayment Component |

**App Component**

**File: App.js**

Design the App component with the below screenshot 1.



**(Screenshot 1)**

**Functional Requirement:**

1. The title **TekGain** in the form must be created within the heading tag <h1>.
2. Imported the **logStatus** variable from the LoginService for getting the log status.
3. If logStatus is false render the login page.(**Refer to the screenshot 1**)
4. Design the form for the following specifications

|  |  |  |
| --- | --- | --- |
| **Element Name** | **UI Control type** | **Element id** |
| User Name | Text Box | userName |
| Password | Text Box | password |
| Login | Button | submit |

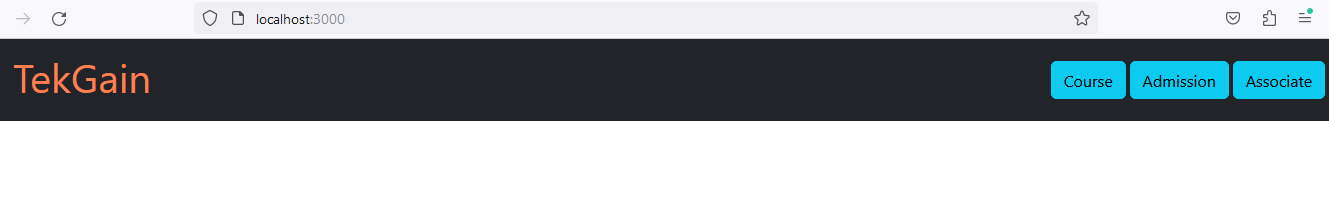
1. The form elements should be designed by using bootstrap form-control className and should handle the input changes for the respective elements.
2. On clicking the Login button, **handleSubmit** event handler method should invoke the method validateUser(userName,password) of LoginService to validate the credentials.
3. If the userName/password is incorrect then display a message **“Invalid username/Password”**
4. If **logStatus** is true, render the **Initial** page as given below**.**
5. By default **App** component should render the login page, On **successful login**, it should display the following menus **(Refer to screenshot 2)**

* Course
* Admission
* Associate

**Routing Link Configuration**

|  |  |  |
| --- | --- | --- |
| **Link** | **Router path** | **Route To** |
| Course | course | Course Component |
| Admission | admission | Admission Component |
| Associate | associate | Associate Component |

1. The form in the UI must be created using bootstrap <Navbar> and form-control className to design title and menus.
2. The menu should be designed by using bootstrap <Nav>, <NavLink> and <Button> tags with appropriate bootstrap className for the below routerlink.
3. Render the <Routing/> component to display the initial page.

****

**(Screenshot 2)**

**Service Specification**

**Login Service**

**File: LoginService.js**

Export the **logStatus** variable to know the login status.

|  |  |
| --- | --- |
| **Service** | **Method** |
| LoginService | validateUser(userName,pass) |

1. The LoginService should have the following requirements

|  |  |  |
| --- | --- | --- |
| **Requirements** | **Method** | **Description** |
| Requirement 1 | validateUser(userName,pass) | This service should invoke the microservice and validate the user credentials.  The micro service which in turn sends back the authorization token for the valid user. Get this token and store in browser local storage memory.  For valid credentials, set the logStatus variable to true.  Else, false.  Note: Use Http.post method to invoke the micro service |

**Note: This authorization token should be added to all the subsequent requests.**

**Course Component**

**File:Course.js.**

**Functional Requirements**

1. Display the title as “**COURSE INFORMATION**” in an <h2> tag.
2. Course component should be designed with the following menus.**(Refer to screenshot 3)**

* Add course
* Update course
* View Course
* View Feedback Rating
* Course Deactivate

Router Link path should be created for the above given components in the Course module as given in the following table.

**Routing Link Configuration**

|  |  |  |
| --- | --- | --- |
| Link | Router path | Route To |
| Add Course | addcourse | AddCourse Component |
| Update Course | updatecourse | UpdateCourse Component |
| View Course | viewcourse | ViewCourse Component |
| View Feedback Rating | viewfeedbackrating | CourseRating Component |
| Course Deactivate | deactivate | CourseDeactivate Component |

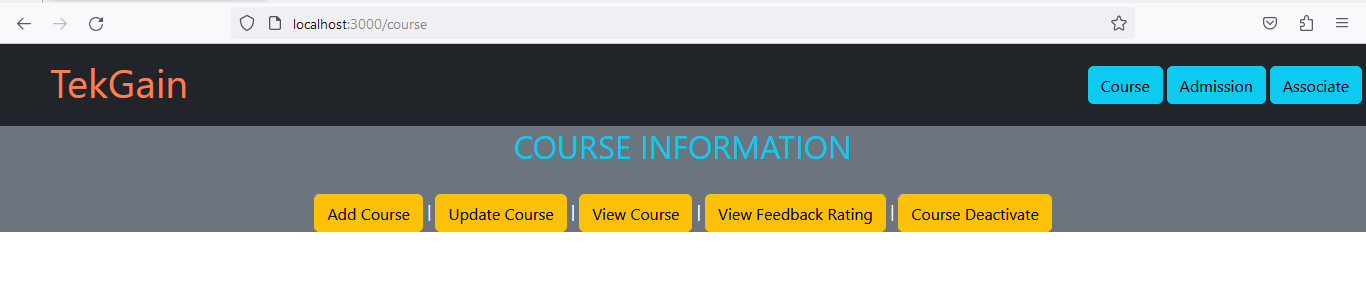
On clicking the Add Course link, the page should route to the AddCourse Component

On clicking the Update Course link, the page should route to UpdateCourse Component.

On clicking the View Course link, the page should route to ViewCourse Component.

On clicking the View Feedback Rating link, the page should route to CourseRating Component.

On clicking the Course Deactivate link, the page should route to CourseDeactivate Component.

****

**(Screenshot 3)**

**Requirements in Course Component**

* All the above components must be created separately.
* All the above created components must be imported in the **Routing** component (given in the code template) and create routes for successful routing.
* Render the <Routing/> component, after the Link creation of the above components.
* Render the <Outlet/> component after the <Routing/> component.
* Each child component should be rendered using the <Outlet /> tag upon clicking the respective buttons (Routing Links).

**AddCourse Component**

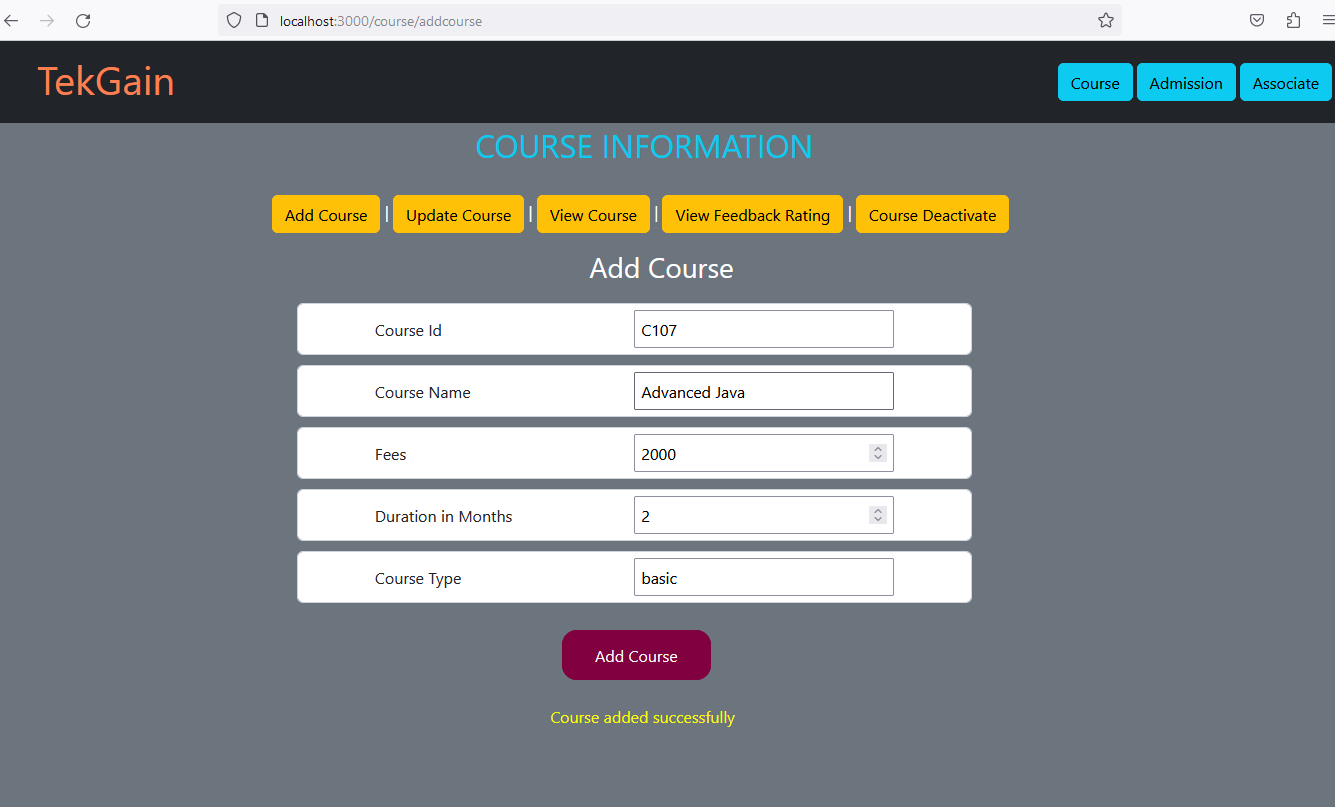
**File.AddCourse.js**

**Component Description**

**AddCourse** component should be rendered upon clicking the Add Course button link inside the Course component module. Required input values are taken from the user and store it in the database.

**Functional Requirements**

1. Design a form for adding the course details with reference to screenshot -3. On click of the Add Course button, the course details should be inserted into the DB.



**(Screenshot 4)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| Course Name | courseName | Text box | courseName |
| Fees | fees | Text box | fees |
| Duration in Months | duration | Text box | duration |
| Course Type | courseType | Text box | courseType |
| Add Course |  | Button | submit |

**Requirements**

* All specified element ids and element names MUST be used in the UI implementation.
* Create the state variables to handle the input change.

1. courseId
2. courseName
3. fees
4. duration
5. courseType

* Title “**Add Course**” should be displayed using <h3> tag.
* Handle the onChange event for respective fields.
* OnClick event should take place upon clicking the Add Course button.
* On clicking the Add Course button, the **handleSubmit** event handler method should invoke the method addCourseDetails(course) of CourseService to add the course to DB.Get the returned promise data and display the message as per the success scenario given below.
* **Post- Condition**: Course details should be stored in the DB. The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the message “**Course added successfully**” in the <div> tag with an id as **message**.**(Refer to screenshot 4)**
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**UpdateCourse Component**

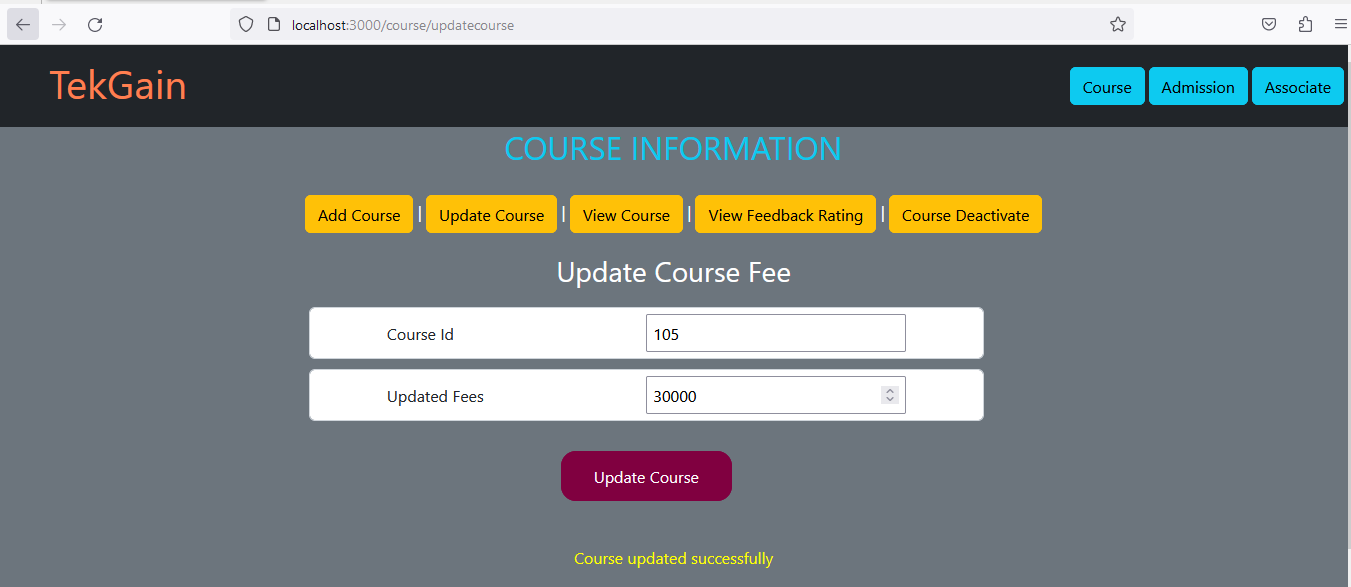
**File:UpdateCourse.js.**

**Component Description**

**UpdateCourse** component should be rendered upon clicking the Update Course button link inside the Course component module. This component should update the course fees for the given course id.

**Functional Requirements**

1. Design a UI to get input for course id and course fee. Update the course fee based on the given course Id in Db and also in the courses array

****

**(Screenshot 5)**

**Component Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| Update Fees | fees | Text box | fees |
| Update Course |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Create state variables to handle the input change.
  1. courseId
  2. fees
  3. message
* Title “**Update Course Fee**” should be displayed using <h3> tag
* Handle the onChange event for respective fields.
* On clicking the Update Course button, the **handleSubmit** event handler method should invoke the method updateCourse(courseId,fees) of CourseService to update the course fees.Get the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Course fees will be updated into the DB.The success/failure message should be rendered in the web page..
* **Success Scenario** :On successful updation, display the message “**Course updated successfully**” in the <div> tag with id as **message. (Refer to screenshot 5)**
* **Failure Scenario**: Display an appropriate error message in the same <div> tag with the id **“message”**.

**ViewCourse Component**

**File: ViewCourse.js**

**Component Description**

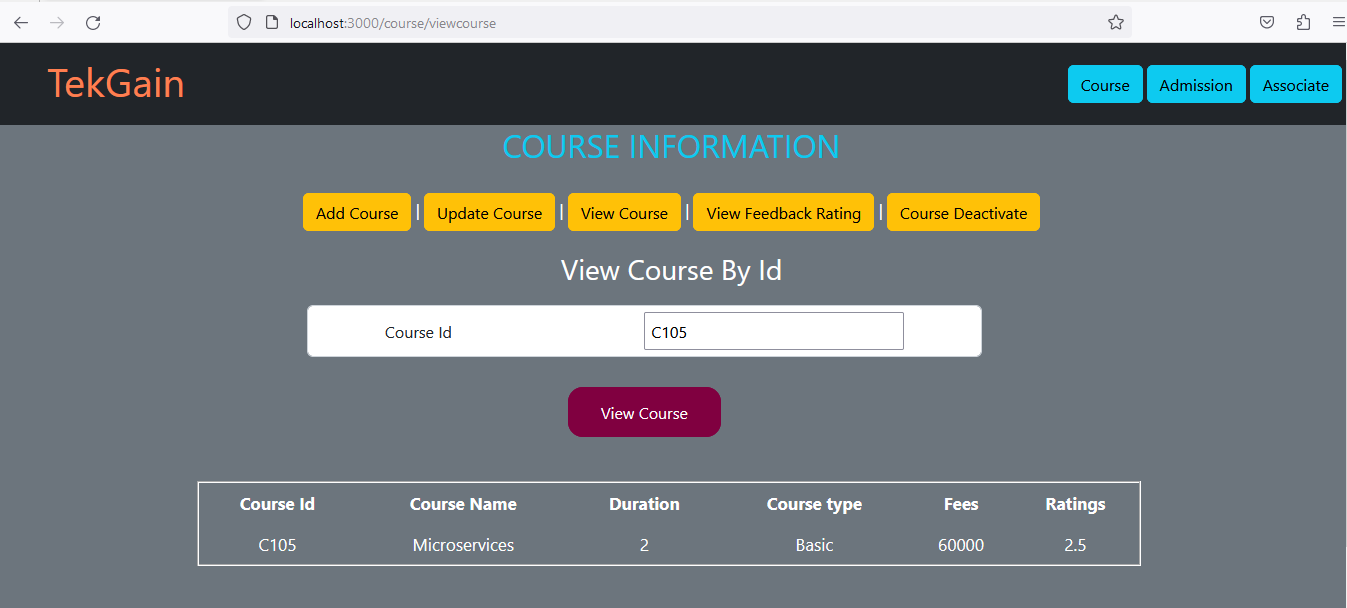
**ViewCourse** component should be rendered upon clicking the View Course button link inside the Course component module. This component should fetch the course details for the given courseId.

**Functional Requirements**

1. Input for courseId should be received from the user , Retrieve the courses which have the same given courseId .(Course id values should be auto populated from the courses array)

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| View Course |  | Button | submit |

****

**(Screenshot -6)**

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**View Course By Id**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. courseId
2. message

* Design the form using form-cotrol className.
* Handle the onChange event for respective fields.
* On clicking the View Course button, the **handleSubmit** event handler method should invoke the method viewCourseById(courseId) of CourseService to view the course details of the given Id.Get the course details from the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Course details will be displayed by iterating the course array.
* **Success Scenario:** Course details will be displayed in the tabular format for the entered course id. **(Refer to screenshot 6)**
* **Failure Scenario:** The error message has to be displayed in the <div> tag with the id **“message”**.

**CourseRating Component**

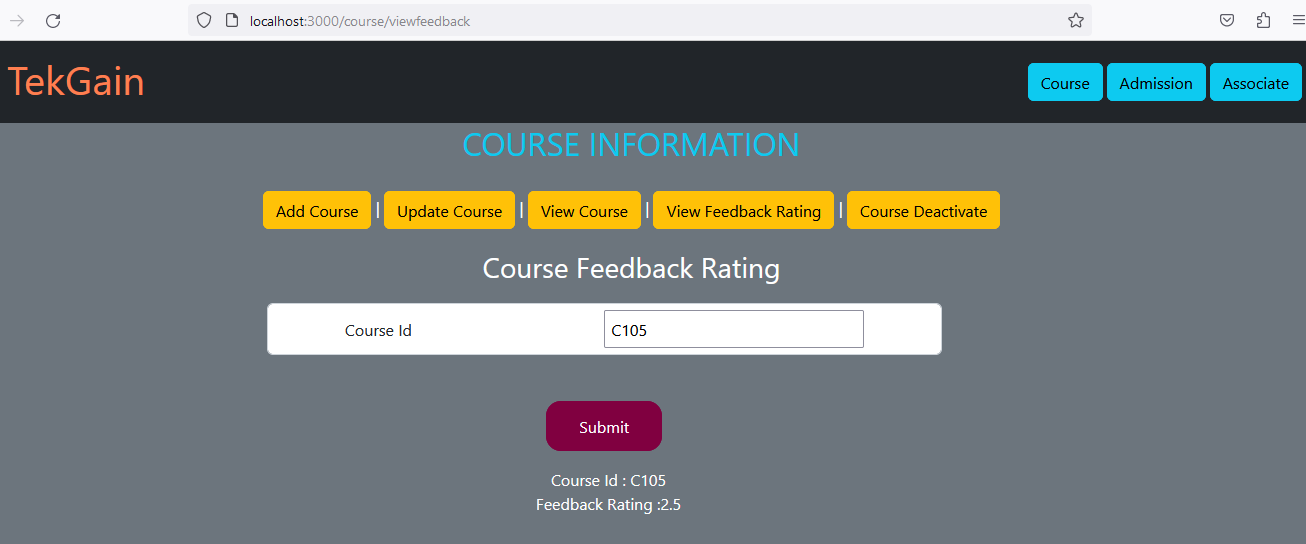
**File: CourseRating.js**

**Component Description**

**CourseRating** component should be rendered upon clicking the View Feedback Rating button link inside the Course component module.The component should display the feedback rating for the given courseId..

**Functional Requirements**

1. Input for courseId should be received from the user , Retrieve the course rating which has the same courseId as given .

****

**(Screenshot -7)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| Submit |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**Course Feedback Rating**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. courseId
2. rating
3. message

* Design the form using form-cotrol className.
* Handle the onChange event for respective fields.
* On clicking the View Course button, the **handleSubmit** event handler method should invoke the method viewFeedback(courseId) of CourseService to view the course feedback of the given course Id.Get the course feedback from the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Course feeback rating is displayed for the given course.id. The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the message **“Course Id: <<couese id>>**

**Feedback Rating: << rating value>> ”.** **(Refer to screenshot 7)**

* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**CourseDeactivate Component**

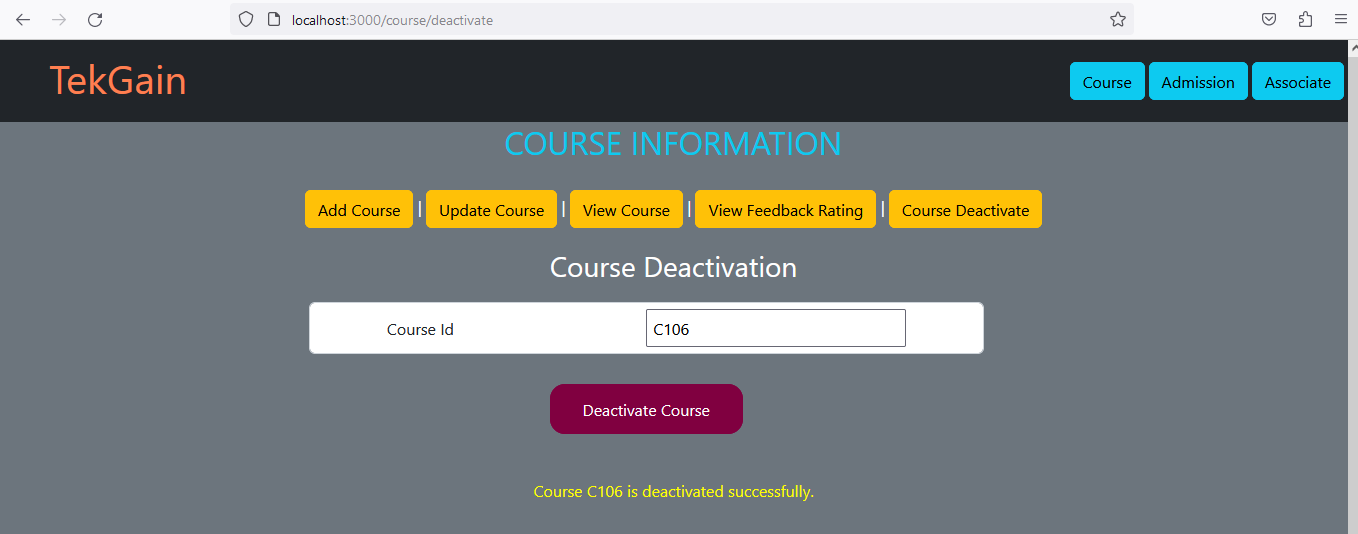
**File: CourseDeactivate**

**Component Description**

**CourseDeactivate** component should be rendered upon clicking the Course Deactivate button link inside the Course component module.The component should deactive the course for the given courseId.

**Functional Requirements**

1. Input for courseId should be received from the user , Deactivate the course which has the same courseId as given.

** (Screenshot -8)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text Box | courseId |
| Deactivate Course |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title **“Course Deactivation”** should be displayed using <h3> tag
* Create state variables to handle the input change.

1. courseId
2. message

Design the form using form-cotrol className.

* Design the form using form-cotrol className.
* Handle the **onChange** event for the respective input fields.
* On clicking the Add button, the **handleSubmit** event handler method should invoke the method deactivate(courseId) of CourseService to deactivate the course from the DB.Get the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Course should be deactivated in the DB. The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the message **“Course <<course id>> is deactivated successfully”** in the <div> tag with an id as **message**.
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**Service Specification**

**Course Service**

**File CourseService.js**

|  |  |
| --- | --- |
| **Service** | **Methods** |
| CourseService | addCourseDetails(course)  updateCourse(courseId,fees)  viewCourseById(courseId)  viewFeedback(courseId) |

1. The CourseService should have the following requirements

|  |  |  |
| --- | --- | --- |
| **Requirements** | **Method** | **Description** |
| Requirement 1 | addCourseDetails(course) | This service should invoke the microservice and insert the course object into DB.This method should return the reponse as promise.  Note: Use Http.post method to invoke the micro service |
| Requirement 2 | updateCourse(courseId,fees) | This service should invoke the microservice and update the course fee for the given coursed in the DB.This method should return the reponse as promise.  Note: Use Http.put method to invoke the micro service |
| Requirement 3 | viewCourseById(courseId) | This service should invoke the microservice and return the course for the given id from the DB.This method should return the reponse as promise.  Note: Use Http.get method to invoke the micro service |
| Requirement 4 | viewFeedback(courseId) | This service should invoke the microservice and display the course feed back of the given course id. This method should return the reponse as promise.  Note: Use Http.delete method to invoke the micro service |

**Note: For all the HTTP request add Authorization header with the received token**

**Associate Component**

**File:Associate.js.**

**Functional Requirements**

1. Display the title as “**ASSOCIATE INFO**” in an <h2> tag with the className “text-info”
2. Associate component should be designed with the following menus.

* Add Associate
* Update Associate
* View Associate

Router Link path should be created for the above given components in the Associate module as given in the following table.

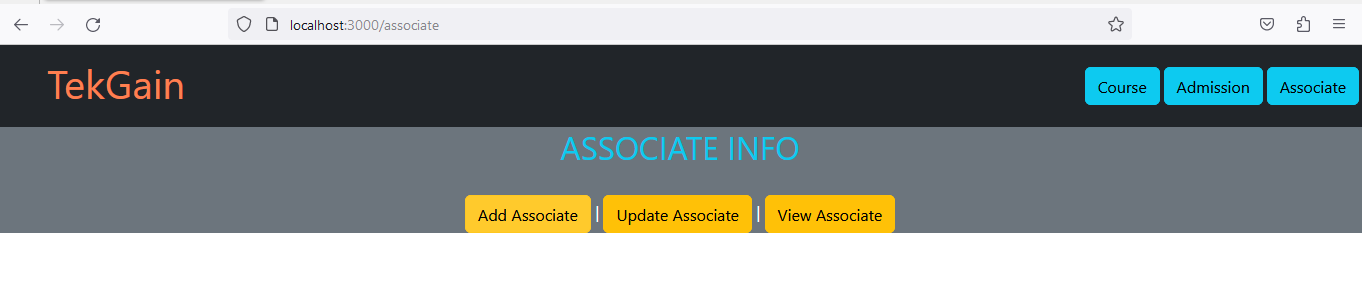
**Routing Link configuration**

|  |  |  |
| --- | --- | --- |
| Link | Router path | Route To |
| Add Associate | addassociate | AddAssociate Component |
| Update Associate | updateassociate | UpdateAssociate Component |
| View Associate | viewassociate | ViewAssociate Component |

On clicking the Add Associate link, the page should route to the AddAssociate Component

On clicking the Update Associate link, the page should route to UpdateAssociate Component.

On clicking the View Associate link, the page should route to ViewAssociate Component.



**(Screenshot-9 )**

**Requirements in Associate Component**

* All the above components must be created separately.
* All the above created components must be imported in the **Routing** component (given in the code template) and create routes for successful routing.
* Render the <Routing/> component, after the Link creation of the above components.
* Render the <Outlet/> component after the <Routing/> component.
* Each child component should be rendered using the <Outlet /> tag upon clicking the respective buttons (Routing Links).

**AddAssociate Component**

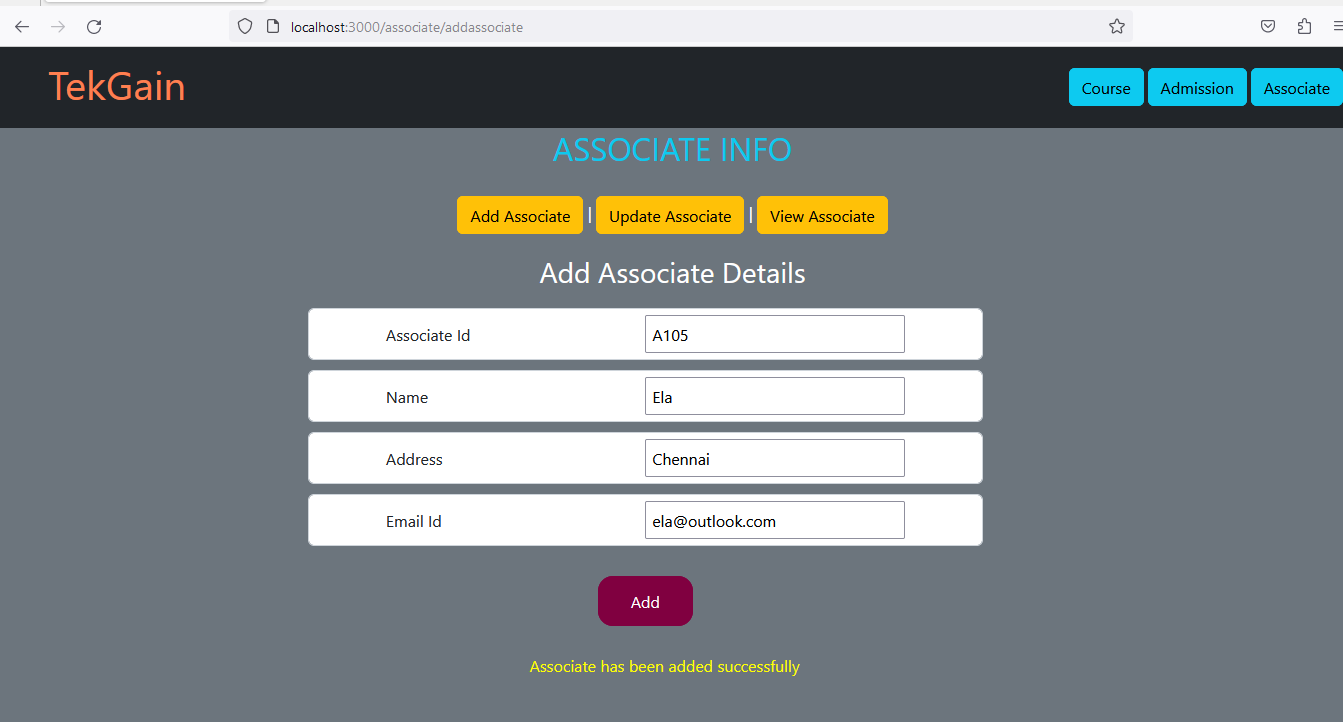
**File.AddAssociate .js**

**Component Description**

**AddAssociate** component should be rendered upon clicking the Add Associate button link inside the **Associate** component module. Required input values are taken from the user and store it in the database.

**Functional Requirements**

1. Design a form for adding the associate details with reference to the below screenshot . On click of the add associate button, the associate details should be inserted into the DB..



**(Screenshot 10)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Name** | **Element Name** | **Control type** | **Element id** |
| Associate Id | associateId | Text box | associateId |
| Name | associateName | Text box | associateName |
| Address | associateAddress | Text box | associateAddress |
| Email Id | associateEmailId | Text box | associateEmailId |
| Add |  | Button | addAsscoiate |

**Requirements**

* All specified element ids and element names MUST be used in the UI implementation.
* Create the state variables to handle the input change.

1. associateId
2. associateName
3. associateAddress
4. associateEmailId

* Title “**Add Associate Details**” should be displayed using <h3> tag.
* Handle the **onChange** event for the respective input fields.
* On clicking the Add button, the **handleSubmit** event handler method should invoke the method addAssociate(associate) of AssociateService to add the associate details to DB.Get the returned promise data and display the message as per the success scenario given below
* **Post- Condition**: Associate details should be added into the DB. The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the message **“Associate has been added successfully**” in the <div> tag with an id as **message**.
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**UpdateAssociate Component**

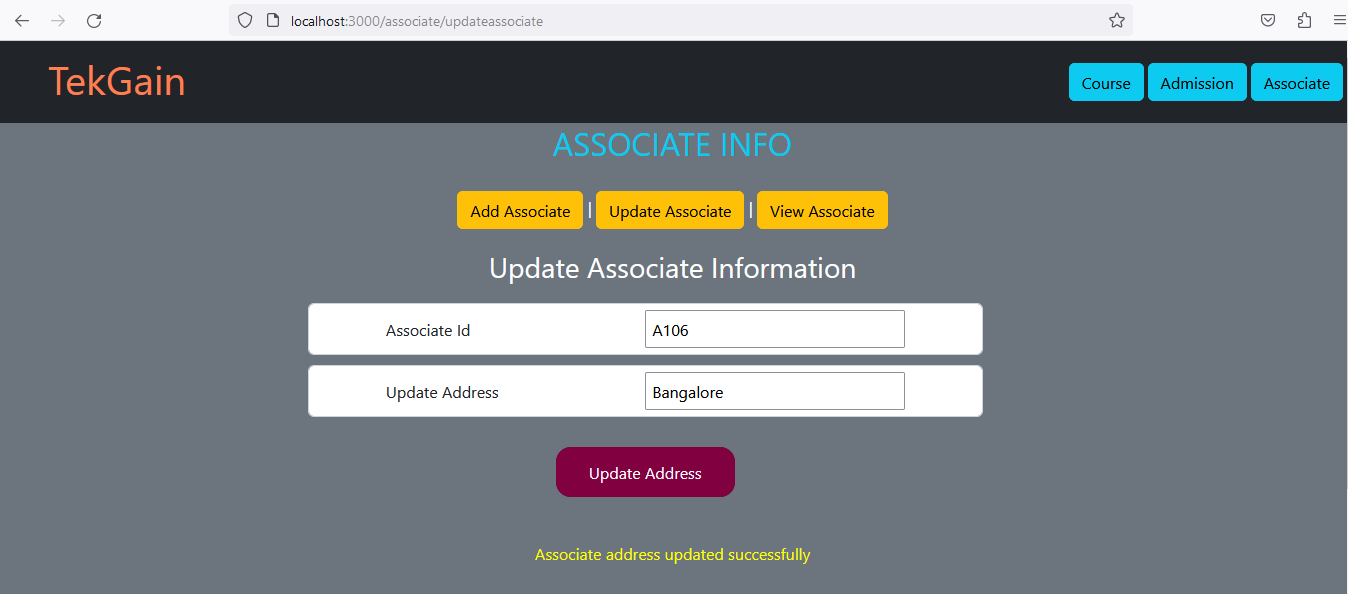
**File:UpdateAssociate.js.**

**Component Description**

**UpdateAssociate** component should be rendered upon clicking the Update Associate button link inside the **Associate** component module. This component should update the associate’s address for the given associate id.

**Functional Requirements**

1. Design a UI to get input for associate id and address. Update the associate details based on the given associate Id in the DB and also in the associate array

****

**(Screenshot 11)**

**Component Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Associate Id | associateId | Text box | associateId |
| Update Address | associateAddress | Text box | associateAddress |
| Update Address |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Create state variables to handle the input change.
  1. associateId
  2. associateAddress
  3. message
* Title “**Update Associate Information**” should be displayed using <h3> tag
* Handle the onChange event for respective fields.
* On clicking the Update Associate button, the **handleSubmit** event handler method should invoke the method updateAssociate(associateId,address) of AssociateService to update the associate address for the given associate Id to DB.Get the returned promise data and display the message as per the success scenario given below
* **Post- Condition**: Associate address will be updated into the DB.The success/failure message should be rendered in the web page.
* **Success Scenario** :On successful updation, display the message **“Associate address updated successfully”** in the <div> tag with id as **message.(Refrer to the screenshot 11)**
* **Failure Scenario:** The error message has to be displayed in the <div> tag with the id **“message”**.

**ViewAssociate Component**

**File: ViewAssociate.js**

**Component Description**

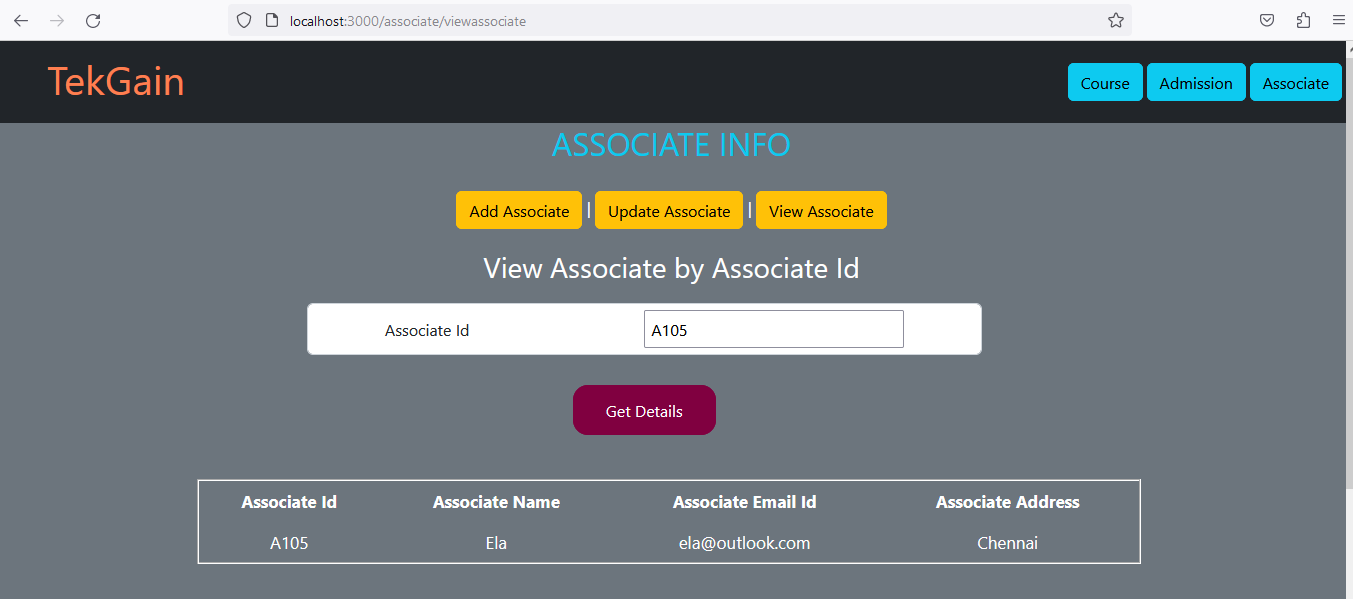
**ViewAssociate** component should be rendered upon clicking the View Associate button link inside the **Associate** component module. This component should update the course fees for the given course id.

**Functional Requirements**

1. Design an UI to get input for associate id. Display the associate details based on the given associate Id from the associate array

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Associate Id | associateId | Text box | associateId |
| Get Details |  | Button | submit |

****

**(Screenshot 12)**

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “View Associate by Associate Id” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. associateId
2. record
3. message

* Design the form using form-cotrol className.
* Handle the onChange event for respective fields.
* On clicking the ViewAssociate button, the **handleSubmit** event handler method should invoke the method viewAssociateById(associateId) of AssociateService to retrieve the associate information for the given associate id.Get the associate detail from the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Associate details will be displayed by iterating the associatesById array. The success/failure message should be rendered in the web page.
* **Success Scenario:** Associate details will be displayed for the entered associate id in the tabular form.**(Refrer to the screenshot 12)**
* **Failure Scenario:** The error message has to be displayed in the <div> tag with the id **“message”**.

**Service Specification**

**Associate Service**

**File AssociateService.js**

|  |  |
| --- | --- |
| **Service** | **Methods** |
| AssociateService | addAssociate(associate)  updateAssociate(associateId,address)  viewAssociateById(associateId) |

1. The AssociateService should have the following requirements

|  |  |  |
| --- | --- | --- |
| **Requirements** | **Method** | **Description** |
| Requirement 1 | addAssociate(associate) | This service should invoke the microservice and insert the associate object into DB.This method should return the reponse as promise.  Note: Use Http.post method to invoke the micro service |
| Requirement 2 | updateAssociate(associateId,address) | This service should invoke the microservice and update the address for the given associate in the DB.This method should return the reponse as promise.  Note: Use Http.put method to invoke the micro service |
| Requirement 3 | viewAssociateById(associateId) | This service should invoke the microservice and return the associate information for the given associate id from the db.This method should return the reponse as promise.  Note: Use Http.get method to invoke the micro service |

**Note: For all the HTTP request add Authorization header with the received token**

**Admission Component**

**File:Admission.js.**

**Functional Requirements**

1. Display the title as “**ADMISSION DETAILS**” in <h2> tag with the className “text-info”
2. Admission component should be designed with the following menus.

* Associate Registration
* Total Fees
* Add feedback
* Highest Fee Details
* View feedback
* Make Payment

Router Link path should be created for the above given components in the Admission module as given in the following table.

**Routing Link Configuration**

|  |  |  |
| --- | --- | --- |
| Link | Router path | Route To |
| Associate Registration | registration | AssociateRegistration Component |
| Total Fee | totalFee | TotalFees Component |
| Add Feedback | addfeedback | AddFeedback Component |
| Highest Fee Details | highestFee | HighestFees Component |
| View Feedback | viewfeedback | ViewFeedback Component |
| Make Payment | makepayment | MakePayment Component |

On clicking the Associate Registration link, the page should route to the AssociateRegistration Component

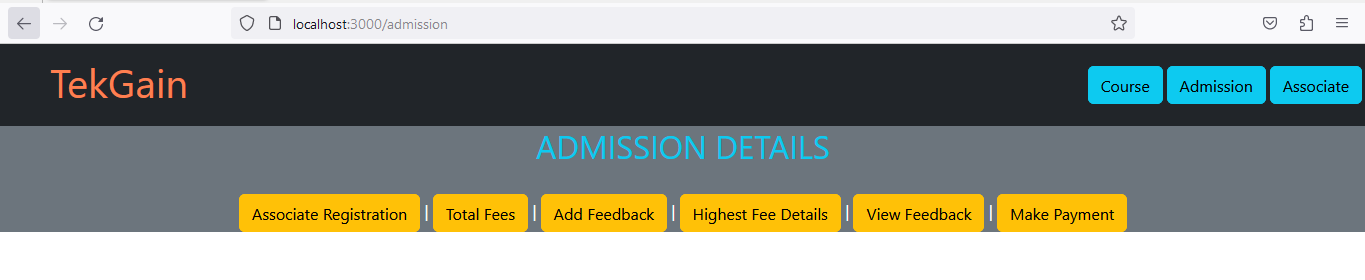
On clicking the Total Fee link, the page should route to TotalFees Component.

On clicking the Add Feedback link, the page should route to AddFeedback Component.

On clicking the Highest Fee Details Rating link, the page should route to FeeDetails Component.

On clicking the View Feedback link, the page should route to ViewFeedback Component.

On clicking the View Make Payment, the page should route to MakePayment Component.

****

**(Screenhsot 13)**

**Requirements in Admission Component**

* All the above components must be created separately.
* All the above created components must be imported in the **Routing** component (given in the code template) and create routes for successful routing.
* Render the <Routing/> component, after the Link creation of the above components.
* Render the <Outlet/> component after the <Routing/> component.
* Each child component should be rendered using the <Outlet /> tag upon clicking the respective buttons (Routing Links).

**AssociateRegistration Component**

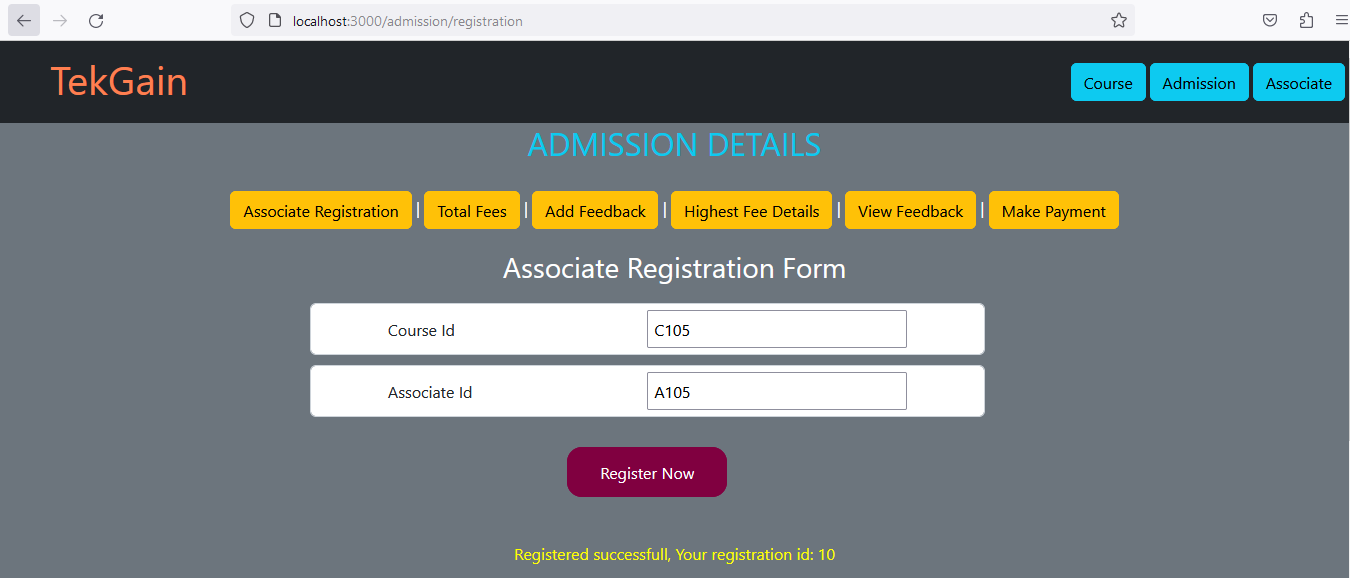
**File.AssociateRegistration .js**

**Component Description**

**AssociateRegistration** component should be rendered upon clicking the Associate Registration button link inside the **Admission** component module.The component should register an associate to he given courseId.

**Functional Requirements**

1. Design a form for adding the associate details with reference to the below screenshot . On click of the register button, the registration details should be inserted into the DB.



**(Screenshot 14)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| Associate Id | associateId | Text box | associateId |
| Register Now |  | Button | submit |

**Requirements**

* All specified element ids and element names MUST be used in the UI implementation.
* Create the state variables to handle the input change.

1. courseId
2. associateId
3. message

* Title “**Associate Registration Form**” should be displayed using <h3> tag.
* Handle the onChange event for respective fields.
* On clicking the Register Now button, the **handleSubmit** event handler method should invoke the method register(associateId,courseId) of AdmissionService to add the associate registration information for the given associate id and course id. Get the registration id from the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Course details should be stored in the DB. The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the message “**Registered successful, Your registration id: <id>**” in the <div> tag with an id as **message.**
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**TotalFees Component**

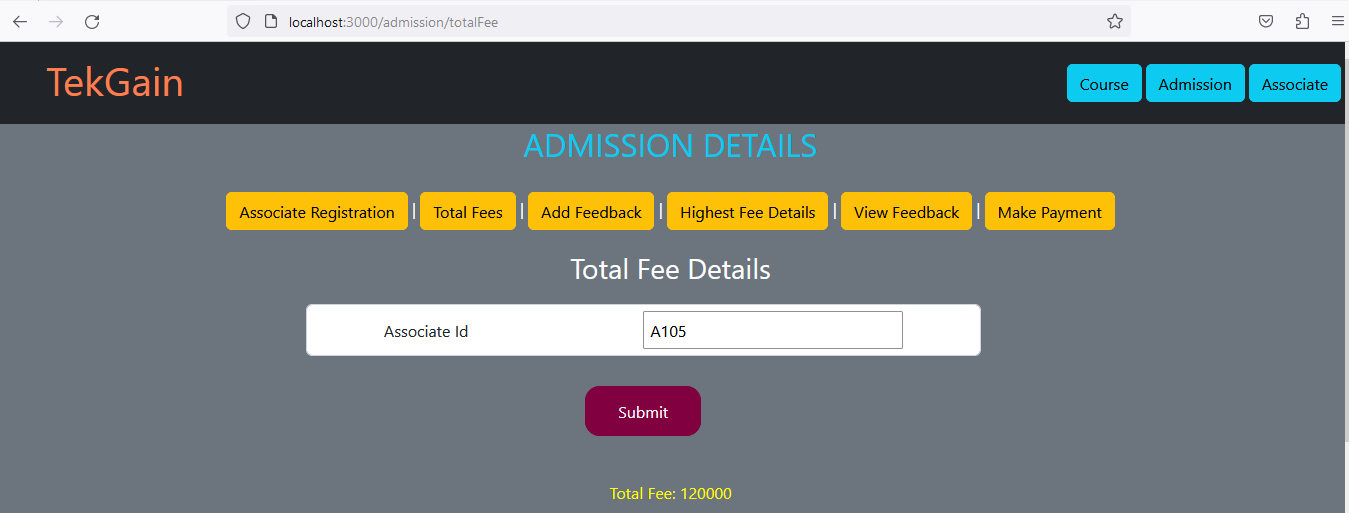
**File: FeeDetails.js**

**Component Description**

**TotalFees** component should be rendered upon clicking the Total Fees button link inside the **Admission** component module.The component should display the total fees paid by an associate for the various courses.

**Functional Requirements**

1. Design a UI to get input for associate id. Find all the courses registered for that associate and calculate the total fee.

****

**(Screenshot -15)**

**Component Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Associate Id | associateId | Text box | courseId |
| Submit |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Create state variables to handle the input change.
  1. associateId
  2. fee
  3. message
* Title “**Total** **Fee Details**” should be displayed using <h3> tag.
* Handle the onChange event for respective fields.
* On clicking the Get Fees button, the **handleSubmit** event handler method should invoke the method calculateFees(associateId) of AdmissionService to get the total fees of the given associate id.Get the fees from the returned promise data and display the information as per the success scenario given below.
* **Post- Condition**: Total amont of the entered associate id will be diaplayed. The success/failure message should be rendered in the web page.
* **Success Scenario** :On successful retrieval, display the message “Total Fee : <<fees>>” message must be displayed in the <div> tag with id as **message.**
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**AddFeedback Component**

**File:** **AddFeedback .js**

**Component Description**

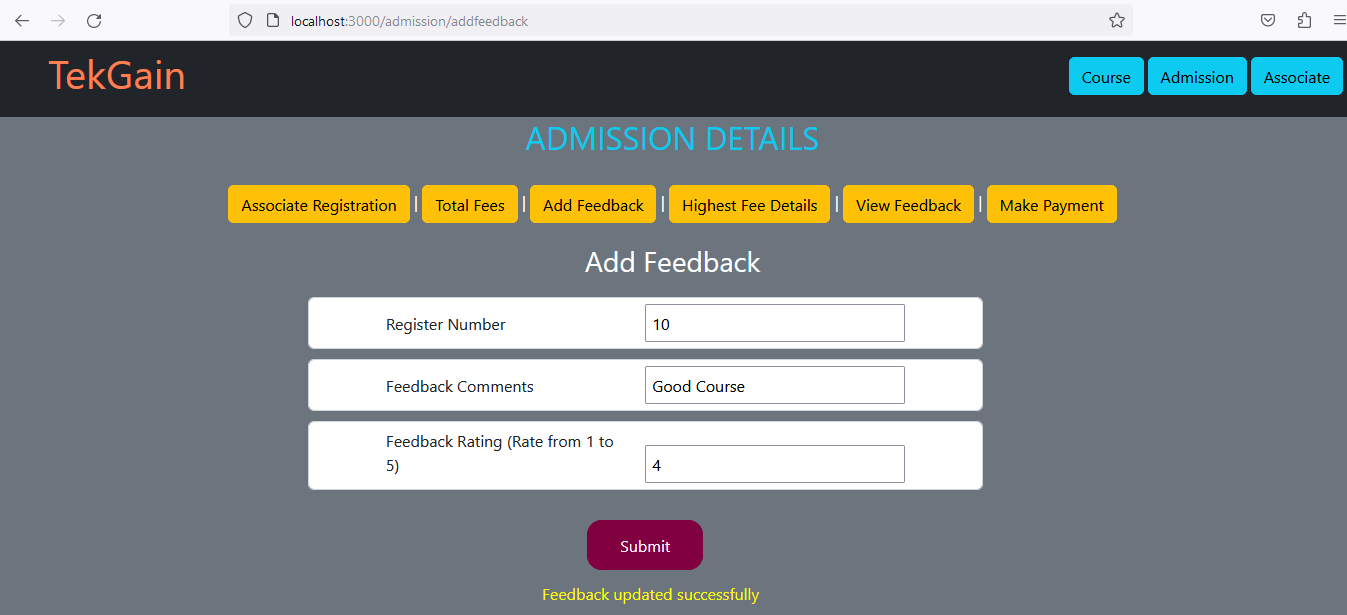
**AddFeedback** component should be rendered upon clicking the Add Feedback button link inside the **Admission** component module.The component should add the course feedback based on the registration id.

**Functional Requirements**

a.Design a UI to get input for feedback Add the details into the DB by using the service methods.

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Register Number | registrationId | Text box | registrationId |
| Feedback Comments | feedback | Text box | feedback |
| Feedback Rating (Rate from 1 to 5) | feedbackRating | Text box | feedbackRating |
| Submit |  | Button | submit |

****

**(Screenshot 16)**

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**Add Feedback**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. regNo
2. Feedback
3. feedbackRating
4. message

* Handle the onChange event for respective fields.
* On clicking the Submit button, the **handleSubmit** event handler method should invoke the method addfeedback(regNo,feedback,feedbackRating) of AdmissionService to add the feedback of the entered register number.Get the returned promise data and display the message as per the success scenario given below.
* **Post- Condition**: Feedback will be updated into the DB.The success/failure message should be rendered in the web page.
* **Success Scenario:** “**Feedback updated successfully**” message must be displayed in the <div> tag with id as **message. (should be -Feedback added successfully)**
* **Failure Scenario:** The error message has to be displayed in the <div> tag with the id **“message”**.

**HighestFeeCalculation Component**

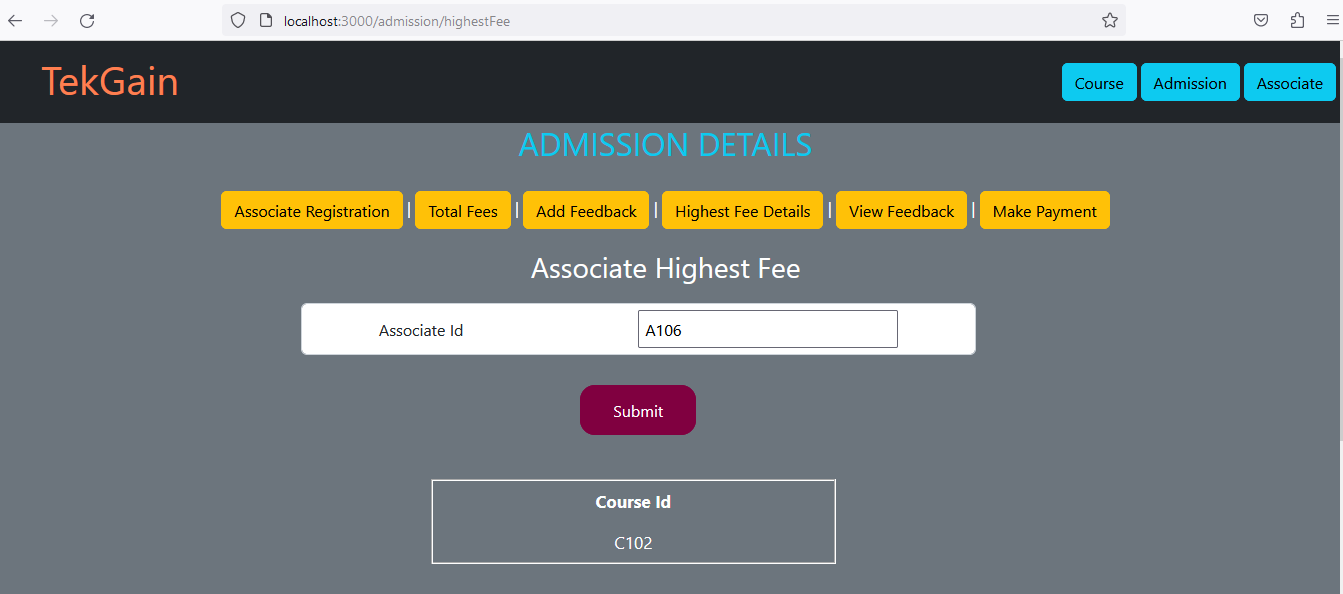
**File: HighestFeeCalculation.js**

**Component Description**

**HighestFeeCalculation** component should be rendered upon clicking the Highest Fee Details button link inside the **Admission** component module.The component should retrieve the highest course fee courseId of the given associate id.

**Functional Requirements**

1. Design a UI to get input for associate Id . Display the Highest fee course details of that associate by iterating the highestFeeArray.

****

**(Screenshot 17)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Associate Id | associateId | Text box | associateId |
| Submit |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**Associate Highest Fee**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. associateId
2. Message
3. records

* Handle the onChange event for respective fields.
* On clicking the Submit button, the **handleSubmit** event handler method should invoke the method highestFee(associateId) of AdmissionService to retrieve the course id of the highest fee for the entered associate id.Get the course id from the returned promise data and display the information as per the success scenario given below
* Render the output in tabular format **(refer to screenshot 17)**
* **Post- Condition**: Highest Fee Course id will be displayed by iterating the highestFeeArray array.The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the Course Id.**(Refer to Screenshot 17)**
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

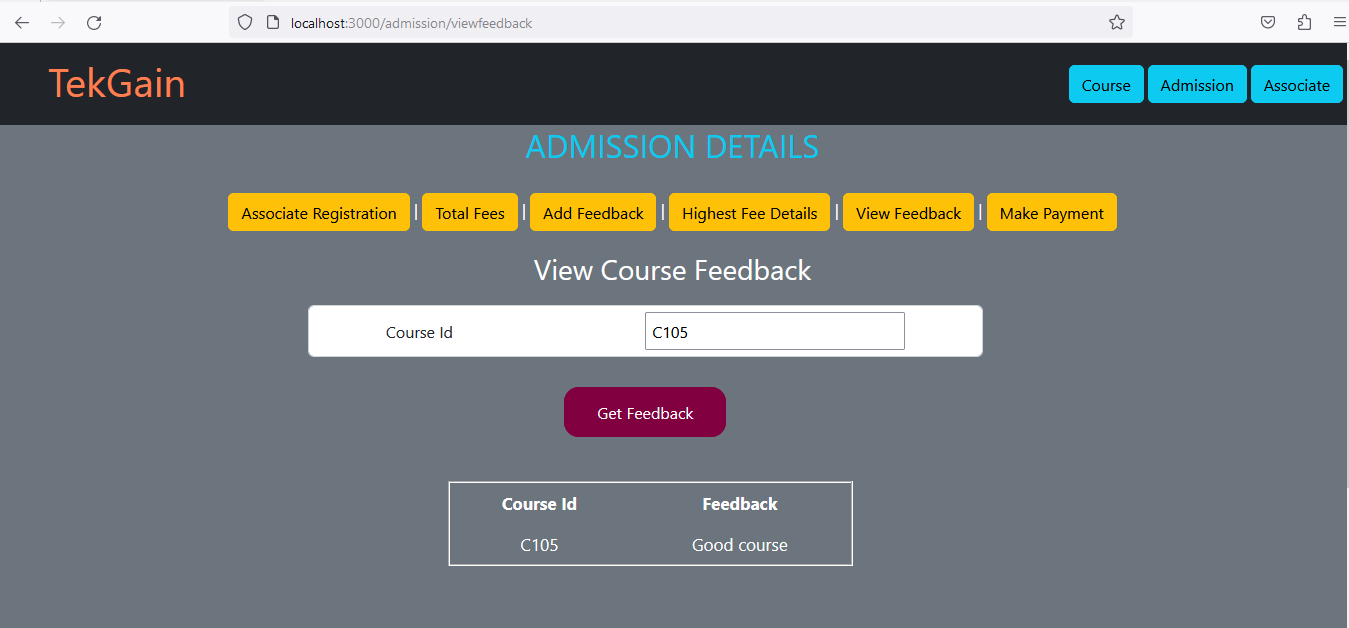
**ViewFeedback Component**

**File: ViewFeedback.js**

**Component Description**

**ViewFeedback** component should be rendered upon clicking the View Feedback button link inside the **Admission** component module.The component should retrieve the feedback details of the given courseId.

**Functional Requirements**

1. Display the Feedback Details based on the course id.  ** (Screenshot 18)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Course Id | courseId | Text box | courseId |
| Get Feedback |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**View Course Feedback**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. associateId
2. message

* Handle the onChange event for respective fields.
* On clicking the Submit button, the **handleSubmit** event handler method should invoke the method viewFeedbackByCourseId(courseId) of AdmissionService to retrieve the course feedback of the entered course id..Get the course feedback from the returned promise data and display the information as per the success scenario given below
* **Post- Condition**: Feedback of a course must be displayed for the entered course id.The success/failure message should be rendered in the web page.
* **Success Scenario** :On success, display the Course Id and its respective feedback.**(Refer to Screenshot 18)**
* **Failure Scenario**: Display an appropriate error message if any of the validation fails in the same <div> tag with an id as **message**.

**MakePayment Component**

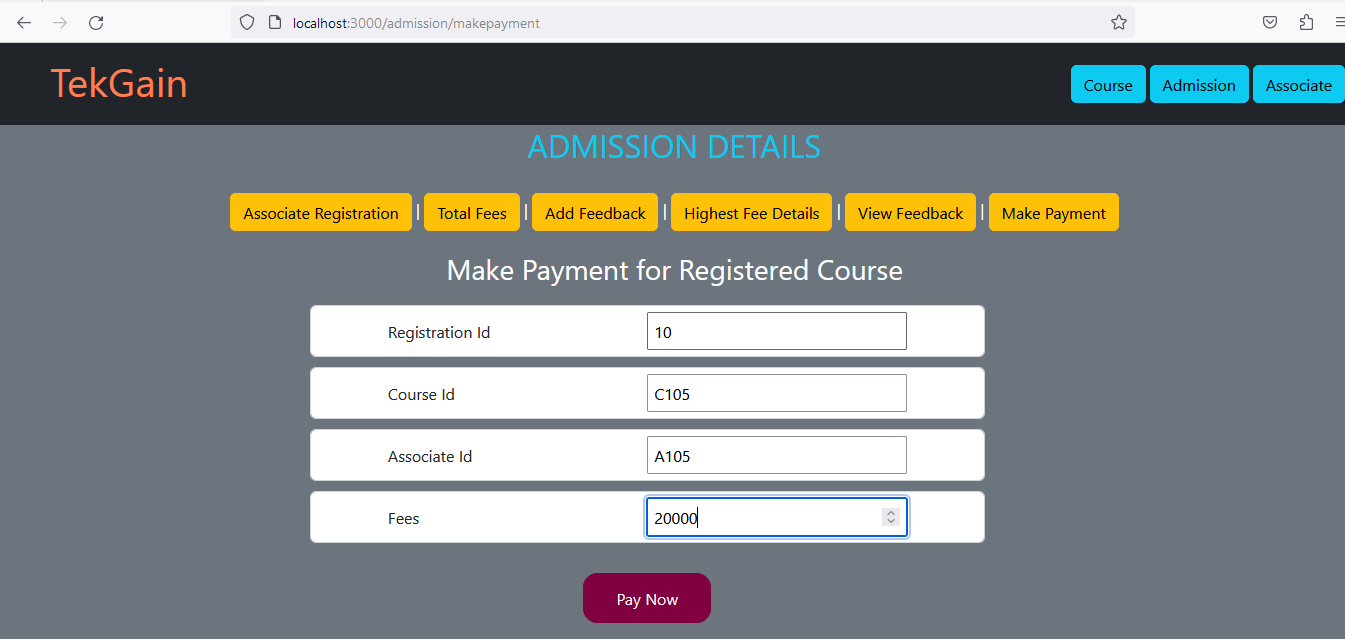
**File: MakePayement.js**

**Component Description**

**MakePayment** component should be rendered upon clicking the Make Payment button link inside the **Admission** component module.The component should add the payment details for the registered course.

**Functional Requirements**

1. Design an UI to get input for payment.Add the details into the DB by using the service methods .

** (Screenshot -19)**

**Component Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Label Nmae** | **Element Name** | **Control type** | **Element id** |
| Registration Id | registrationId | Text box | registrationId |
| Course Id | courseId | Text box | courseId |
| Associate Id | associateId | Text box | associateId |
| Fees | fees | Text box | fees |
| Pay Now |  | Button | submit |

**Requirements**

* All specified ids MUST be used in the UI implementation.
* Title “**Make Payment for Registered Course**” should be displayed using <h3> tag
* Create state variables to handle the input change.

1. registrationId
2. courseId
3. associateId
4. Fees
5. message

* Design the form using form-control className.
* Handle the onChange event for respective fields.
* On clicking the Submit button, the **handleSubmit** event handler method should invoke the method makePayment(registartionId) of AdmissionService to make payment for the entered registration id.Get the returned promise data and display the message as per the success scenario given below
* **Post- Condition**: Feedback will be updated into the DB for the give registration Id.The success/failure message should be rendered in the web page.
* **Success Scenario:** “**Payment Successful**” message must be displayed in the <div> tag with id as **message.**
* **Failure Scenario:** The error message has to be displayed in the <div> tag with the id **“message”**.

**Service Specification**

**Admission Service**

**File AdmissionService.js**

|  |  |
| --- | --- |
| **Service** | **Methods** |
| AdmissionService | register(associateId,courseId)  calculateFees(associateId)  addfeedback(regNo,feedback,feedbackRating)  highestFee(associateId)  viewFeedbackByCourseId(courseId)  makePayment(registartionId) |

1. The AdmissionService should have the following requirements

|  |  |  |
| --- | --- | --- |
| **Requirements** | **Method** | **Description** |
| Requirement 1 | register(associateId,courseId) | This service should invoke the microservice and insert the admission object into DB. This method should return the reponse as promise.  Note: Use Http.post method to invoke the micro service |
| Requirement 2 | calculateFees(associateId) | This service should invoke the microservice and get the total fee for the associate from the DB.This method should return the reponse as promise.  Note: Use Http.Put method to invoke the micro service |
| Requirement 3 | addfeedback(regNo,feedback,feedbackRating) | This service should invoke the microservice and insert the feedback details into DB.This method should return the reponse as promise.  Note: Use Http.post method to invoke the micro service |
| Requirement 4 | highestFee(associateId) | This service should invoke the microservice and get the higeshest paid course of an associate from the DB.This method should return the reponse as promise.  Note: Use Http.get method to invoke the micro service |
| Requirement 5 | viewFeedbackByCourseId(courseId) | This service should invoke the microservice and get the course feedbackfor the given courseId from the DB  Note: Use Http.get method to invoke the micro service |
| Requirement 6 | makePayment(registartionId) | This service should invoke the microservice and insert the payment object into DB.This method should return the reponse as promise.  Note: Use Http.post method to invoke the micro service |

**Note: For all the HTTP request add Authorization header with the received token**

**Error Handling**

Handle all the exceptions/errors and return the error response as a string in the respective components.

**Unit Testing**

**File: App.test.js**

Unit test all the functionalities in component and Service by using **Jest**. In code skeleton, we have provided the **App.test.js** file to write the unit test code.