Student Grades Website

Title Of Your Project: Student Grading

Name: Amy Waldron

Group Members: None

Supervisor: Owen Foley

GitHub Link: https://github.com/AmyWaldron01/StudentGrades

Contents

Project requirements

My project is a web application for managing your students grades as a lecturer. The requirements of the project include the ability to view all grades, add grades, and edit grades. The application allows users to log in and out, with login credentials as **username** :“admin” **password** :“admin”. This ensures you can only see the grades as the lecturer. The frontend of the application is built using React and Bootstrap, while the backend is built using Node.js and MongoDB.

The specific requirements of the project can be broken down into the following:

1. View all grades: Users should be able to view a list of all the grades in the system. This functionality is implemented in the **ViewGrades** component.
2. Add grades: Users should be able to add new grades to the system. This functionality is implemented in the **AddGrades** component.
3. Edit grades: Users should be able to edit existing grades in the system. This functionality is implemented in the **EditStudents** component.
4. Login and authentication: Users can only login with the correct log in information. Which is admin, admin.
5. CalculateGrade: The grades are calculated based on the total marks obtained by the student in each assessment. The **calculateGrade** function takes the total marks obtained by the student as an argument and uses a switch statement to determine the corresponding letter grade based on the following ranges: A (90 or above), B (80-89), C (70-79), D (60-69), F (0-50). The function returns the letter grade, which is then displayed in the student's information along with the percentage of marks obtained in each assessment. The overall grade for the course is determined by adding up the percentages obtained in each assessment, with assignments 1 and 2 each contributing 25% and the project contributing 50%.

Overall, is robust and functional web application for managing grades. The frontend and backend are both well-designed and implemented, and the project provides a good example of how to build a full-stack web application using modern web development technologies.

Technologies used.

This project is written in JavaScript and uses the React library to create a user interface. The code imports the necessary dependencies and components from external libraries, such as axios for making HTTP requests, react-router-dom for client-side routing, reactjs-popup for creating popup windows, and react-bootstrap for styling components.

Design methodologies implemented.

Server.js

- First, the code imports required dependencies such as Express, body-parser, mongoose, and cors using the require function.
- It then sets up an instance of the Express application by creating an object of the express function.
- The application is configured to use port 4000 to listen to incoming requests.
- The bodyParser middleware is used to parse incoming request bodies in JSON and urlencoded format.
- The CORS middleware is used to enable Cross-Origin Resource Sharing (CORS).
- A connection is established to the MongoDB database using the Mongoose library.
- A schema is defined for the "STUDENTS" collection in the database.
- A model is created from the schema using the mongoose.model() method.
- The application defines a series of RESTful API endpoints for creating, reading, updating, and deleting student grades.
- Finally, the application is started and set to listen on the specified port using the app.listen() method.

Add Grade

The code defines a class component named AddGrades, which extends the React.Component class. The constructor function is used to initialize the state of the component, setting the values for the student, module, asst1, asst2, and project properties to empty strings.

The component includes several functions to handle events triggered by the user interface, such as onChangeStudent, onChangeModule, onChangeAsst1, onChangeAsst2, and onChangeproject. These functions update the corresponding state property with the value entered by the user.

The handleSubmit function is called when the user submits the form to add a new grade. This function prevents the default behavior of the form, logs the values of the state properties to the console, creates a new grade object using the values of the state properties, and sends an HTTP POST request to the server to add the new grade. If the request is successful, the state properties are reset to their initial empty string values.

The render function returns the JSX code that defines the user interface of the component. The component includes a Popup component that displays a login form when the user clicks on a button. The form includes two input fields for the username and password and a submit button. The component also includes a form for adding a new grade that includes input fields for the student name, module name, and three grades. Finally, the component returns a button to submit the form.

Edit Student

This is a React component written in JavaScript that allows the user to edit student grades in a database.

The first line imports the Axios library which is used to make HTTP requests to the server. The next three lines import the useEffect and useState hooks from the React library, as well as the useParams hook from the React Router library.

The EditStudents function is then defined as a React component that returns JSX, which is a syntax extension to JavaScript that allows for the creation of HTML-like elements in JavaScript code.

Inside the component, the useParams hook is used to get the id of the student whose grades are being edited from the URL. Then, five pieces of state are defined using the useState hook to store the current values of the student name, module name, and three grades.

The useEffect hook is then used to fetch the current data from the server when the component is first loaded. It sends an HTTP GET request to the server using Axios, passing in the student id obtained from the URL. If the request is successful, the data is stored in the state variables using the set functions.

The handleSubmit function is defined to handle the form submission when the user clicks the "Edit Grades" button. It first creates an alert message displaying the new values of the student name, module name, and grades entered by the user. Then, it creates an object containing the updated data and sends an HTTP PUT request to the server using Axios, passing in the student id obtained from the URL.

Finally, the component returns a form with input fields for each piece of data to be edited. Each input field is associated with a state variable using the value and onChange attributes. When the user types into an input field, the corresponding state variable is updated using the set function. When the form is submitted, the handleSubmit function is called.

Home Page

This is a basic React component that renders a homepage. It has a class called **HomePage** that extends the **React.Component** class. Inside the **HomePage** class, there is a **render** method that returns a JSX expression.

The **render** method returns a ** ** element that contains a heading, some text about the app, and an image. The heading says "Welcome to Student Repository". Below the heading, there is a subheading that displays the current local time using **new Date().toLocaleTimeString()**.

The text below the subheading explains what the app does. It tells users that they can keep track of their students' grades by adding them to a repository. It also explains how to add grades and view them. Additionally, there is a list that explains the purpose of the "Edit" and "Delete" sections. Lastly, there is an image that is displayed at the bottom of the page.

This component is exported using the **export** keyword and can be imported and used in other parts of the application.

Student Item

This is a React component that defines the rendering of a single student's information on a web page. The component imports React, Card, Button, Link from react-bootstrap, and axios.

The class is called "StudentItem" and extends the React.Component class. It contains a constructor that binds the "DeleteGrade" method to the component.

The "DeleteGrade" method makes a DELETE request to a specific endpoint on the server to delete the student's information. If successful, the component will reload using the "Reload" method that is passed as a prop to the component.

The "calculateGrade" method calculates the grade of the student based on the weighted grades of their assessments and project.

The "render" method returns a Card from the React Bootstrap library that displays the student's name, module, and marks. It also calculates the grade using the "calculateGrade" method. The render method also includes a "Link" component that allows the user to edit the student's information and a "Button" component that calls the "DeleteGrade" method to delete the student's information.

Students

This page is a React component that renders a list of student items using the **StudentItem** component. It imports **React** and the **StudentItem** component.

The **Students** component receives an array of **students** as props and uses the **map()** method to loop through each student in the array. For each student, it renders a **StudentItem** component and passes the student object and a unique key as props. It also passes a **Reload** function as a prop to the **StudentItem** component, which is used to reload the student list after a deletion or update.

Overall, this component is used to display a list of students by rendering each student item as a separate component.

View Grade

This page is a React component that displays a list of students' grades. It imports the **React** library and the **Students** component from a local file named **students.js**. It also imports the **axios** library to fetch data from an API endpoint.

The **ViewGrades** class extends **React.Component** and has a **constructor** method that calls the **super** method and binds the **componentDidMount** method to the component. The **componentDidMount** method is called after the component has mounted and fetches data from the API endpoint using the **axios.get** method. If the request is successful, the component's **state** is updated with the response data. If the request fails, an error message is logged to the console.

The **state** object contains an empty **students** array. The **render** method displays a header and passes the **students** array and a **Reload** method to the **Students** component as props. The **Reload** method is set to **componentDidMount** to ensure that the component is reloaded after data is fetched from the server. The **Students** component then maps over the **students** array and creates a new **StudentItem**

component for each student with the student's data as props. Finally, the **StudentItem** components are returned as an array and displayed on the page.

App.js

This page is the main app file for my web application. It imports React, Bootstrap, and various other components and pages. The class **App** is defined as a child of the **React.Component** class, which will render the web page.

The **render()** method of the **App** class returns the entire webpage, which includes a navbar with links to different pages, and a **Routes** component that maps different URLs to specific pages of the application.

The different pages imported from other files are linked to the corresponding paths through the **Routes** component. When a user clicks a link in the navbar, the **Routes** component will render the appropriate page component.

This is all wrapped in a **Router** component from the **react-router-dom** package, which provides client-side routing and navigation for the web app. The **BrowserRouter** is used to manage the application's URLs and handle the rendering of the pages.

Architecture of the solution

My web application uses a client-server architecture, where the frontend is built using React and the backend is built using Node.js and MongoDB.

The frontend is divided into several components, each of which has a specific functionality. The **App** component serves as the main component and is responsible for rendering the different pages of the application. The **HomePage**, **ViewGrades**, **AddGrades**, **Login**, and **EditStudents** components are responsible for rendering their respective pages.

React Router is used for client-side routing, which allows the user to navigate between pages without causing a full page to reload. Bootstrap is used for styling.

Axios is used for making HTTP requests to the backend API. The **ViewGrades** component, for example, uses Axios to make a GET request to the **/api/grades** endpoint to fetch data from the backend.

On the backend, the server is built using Node.js and Express.js. The API endpoints are defined using Express.js and the data is stored in MongoDB. The **grades** collection in the MongoDB database stores the data for the application.

The API endpoints are responsible for handling requests from the frontend and performing operations on the data in the MongoDB database. For example, the **GET /api/grades** endpoint returns all the grades from the **grades** collection.

In summary, the architecture of the solution is a client-server architecture, where the frontend is built using React and the backend is built using Node.js and MongoDB. The frontend uses React Router for client-side routing and Axios for making HTTP requests to the backend API. The backend uses Express.js to define the API endpoints and MongoDB to store the data for the application.

Project Management Style and Limitations

In the face of unforeseen circumstances and a group project where the other member dropped out, I acted proactively and took charge of the situation. I didn't let the situation get the better of me, instead, I remained focused and organized.

First, I assessed the situation and the tasks at hand, then I planned a course of action to ensure the project was completed within the given deadline. I started by breaking down the tasks into smaller, manageable parts and then prioritized them according to their level of importance. I then set clear goals and deadlines for each task and regularly checked in with myself to ensure that I was on track. I also made sure to communicate regularly with the project supervisor and keep them updated on the progress of the project. By taking charge of the situation and acting proactively, I was able to complete the project on time and to a high standard. This experience taught me the importance of adaptability, resilience, and taking initiative, even in the face of challenging situations.

Test Cases

1. Test that the Home page is rendered correctly by checking if the text "Welcome to our grading system!" is displayed.
2. Test that the View Grades page is rendered correctly by checking if a list of grades is displayed after fetching data from the server.
3. Test that the Add Grades page is rendered correctly by checking if a form is displayed with the appropriate input fields.
4. Test that the Edit Students page is rendered correctly by checking if a form is displayed with the data of the selected student.
5. Test that the Login page is rendered correctly by checking if a form is displayed with the appropriate input fields for name and password.
6. Test that clicking on the "View Grades" link in the Navbar redirects the user to the View Grades page.
7. Test that clicking on the "Add Grades" link in the Navbar redirects the user to the Add Grades page.
8. Test that clicking on the "Edit" button on the View Grades page redirects the user to the Edit Students page with the correct student data.
9. Test that submitting the Add Grades form with valid inputs adds the new grade to the list on the View Grades page.
10. Test that submitting the Edit Students form with valid inputs updates the student data on the View Grades page.

Conclusions

In conclusion, the project was aimed at creating a web application for managing student grades. The code I studied involved the implementation of different components such as adding new grades, viewing all grades, and editing students' information.

Through this project, I learned the following:

- Understanding of the React framework and its components
- Ability to implement RESTful API endpoints for CRUD operations
- Improved skills in using Bootstrap for creating a responsive UI
- Improved skills in using Axios for making API requests

From a learning perspective, I gained knowledge in building full-stack web applications using React, Node.js, and MongoDB. This project also helped me to improve my problem-solving skills taught me the importance of adaptability, resilience, and taking initiative, even in the face of challenging situations.

Future work could involve adding features such as user authentication, implementing data visualization tools, and improving the UI/UX of the application.

Overall, this project was a great learning experience, and I am proud of what was achieved.