A logo with a lion head

Description automatically generated

**Faculty of Computing, Engineering & Media (CEM)**

**Coursework Brief 2024/25**

|  |  |  |
| --- | --- | --- |
| **Name:** | **Ahmed Nasif Shahriar** | |
| **Student ID** | **P2719390** | |
| **Module name:** | **Development Project CTEC3451** | |
| **Title of the Assessment:** | **Development Project Final Deliverable** | |
| **Date & Time Due (the deadline):** | | **3rd May 2024** |

Contents

[1. Introduction 5](#_Toc166146305)

[1.1 Background 5](#_Toc166146306)

[1.2 What is the Product? 6](#_Toc166146307)

[1.3 Personal Motivation 6](#_Toc166146308)

[1.4 Objectives 7](#_Toc166146309)

[2. Research 8](#_Toc166146310)

[2.1 Functional requirement Analysis 9](#_Toc166146311)

[2.3 Tooling 9](#_Toc166146312)

[2.4 Development methodology 10](#_Toc166146313)

[3. Design 11](#_Toc166146314)

[3.1 Erd Diagram 11](#_Toc166146315)

[3.2 Activity Diagram 13](#_Toc166146316)

[4. Implementation 14](#_Toc166146317)

[4.1 How the project was set up. 14](#_Toc166146318)

[4.2 Font end set up. 18](#_Toc166146319)

[4.2.1 Using Axios 19](#_Toc166146320)

[4.2.2 Tailwind CSS 21](#_Toc166146321)

[4.3 The Database and Backend Set up. 22](#_Toc166146322)

[4.3.1 The SQL Script 24](#_Toc166146323)

[4.3.2 Node.js Setup with file structure 25](#_Toc166146324)

[4.3.3 MySQL set up. 25](#_Toc166146325)

[4.3.4 The end points and routing 26](#_Toc166146326)

[4.3.5 Node.js Example Code 27](#_Toc166146327)

[4.3.6 How the end points were set in the App.js file by using express.js. 28](#_Toc166146328)

[4.4 The Security measures 30](#_Toc166146329)

[4.4.1 The dotenv file 30](#_Toc166146330)

[4.4.2 The middleware. 30](#_Toc166146331)

[4.4.3 The Cors 31](#_Toc166146332)

[4.4.4 Using JWT’s 31](#_Toc166146333)

[4.4.5 The hashing 32](#_Toc166146334)

[4.5 The authentication flow 33](#_Toc166146335)

[4.5.1 The student system authentication 33](#_Toc166146336)

[4.5.2 Using Session for Authentication 35](#_Toc166146337)

[4.5.3 How the session was used in the login. 36](#_Toc166146338)

[4.6 The course browsing. 37](#_Toc166146339)

[4.7 Course enrollments 41](#_Toc166146340)

[4.7.1 Course View 41](#_Toc166146341)

[4.7.2 Shopping Cart 42](#_Toc166146342)

[4.7.3 The navigation of the course view 43](#_Toc166146343)

[4.7.4 The discussion forum 44](#_Toc166146344)

[4.7.5 Take Test 46](#_Toc166146345)

[4.7.6 The Learning Material 48](#_Toc166146346)

[4.8 Teacher system course Management 49](#_Toc166146347)

[4.8.1 Course Creation 50](#_Toc166146348)

[4.8.2 Content Creation 52](#_Toc166146349)

[4.8.3 The content, course view and management 54](#_Toc166146350)

[4.9 Interesting Piece of Code chapter 56](#_Toc166146351)

[4.9.1 The Live Search Implementation 56](#_Toc166146352)

[4.9.2 The Search engine code 60](#_Toc166146353)

[4.9.3 The recommendation algorithm in the backend 61](#_Toc166146354)

[4.9.4 Take Test Code. 66](#_Toc166146355)

[5. Testing Chapter 68](#_Toc166146356)

[5.1 The unit test 68](#_Toc166146357)

[5.2 Validation 69](#_Toc166146358)

[5.3 Error Handling 69](#_Toc166146359)

[6. Critical Evaluation 71](#_Toc166146360)

[6.1 The good 71](#_Toc166146361)

[6.2 The Enhancement 71](#_Toc166146362)

[7. Conclusion 72](#_Toc166146363)

[8. References 72](#_Toc166146364)

# 1. Introduction

## 1.1 Background

The rise of e-learning systems has dramatically transformed the educational landscape, fueled by technological advances, and changing societal demands. In the past, education was often restricted to the confines of physical classrooms, where factors like space and resources limited the pace and scope of learning. But with the internet and digital technology flourishing in the late 20th century, a new era in education was born. What started as simple online correspondences has now evolved into rich, interactive platforms offering multimedia courses that anyone can access from anywhere in the world. (History of E-Learning Platform, 2023)

These platforms are incredibly versatile, accommodating learners of all types—those juggling jobs, family responsibilities, or other personal commitments. They also provide a valuable resource for teachers, enabling them to create and share courses without the traditional costs of maintaining a physical classroom. Furthermore, these systems allow for the continuous updating of course materials, ensuring that content remains current with the latest discoveries and advancements across various fields—a notable improvement over outdated textbooks. (Cristin Perry, 2024)

Today, e-learning systems offer a dynamic and inclusive way to learn, meeting the modern student's need for flexibility and diverse learning options. These platforms effectively connect students with expert instructors from around the globe. Learners can choose courses that match their specific educational goals and schedules, helping them gain new skills or deepen their existing knowledge whenever it suits them best. Likewise, educators have the opportunity to design and distribute their own courses, reaching a worldwide audience beyond the limits of traditional education. This mutual exchange not only makes education more accessible but also creates a thriving community of lifelong learners and professionals. E-learning systems equip everyone with the tools needed to expand their horizons, advance their careers, and share their expertise with the world, all in an efficient and effective manner.

## 1.2 What is the Product?

At first in my contract I thought to add AI functionality but not I have decided to like personalized recommendation and automated assessment like automated question creation and feedback but these functionality are there but not AI based or complete AI for example the personalized learning is these but it is done using rule based approach and also assessment creation is there but not done using automation so I decided to drop the name AI-powered E-learning system instead I am calling it E-Learning system , it is web based system. Hence the product is an E-learning platform designed to emulate the success and functionality of established systems like Udemy and Coursera. This platform is a comprehensive online educational environment where students can browse and enroll in a wide variety of courses across numerous disciplines. Whether it's picking up interest, advancing professional skills, or pursuing academic subjects, our platform offers accessible and engaging content created by the Teachers. Teachers and industry professionals can create and distribute their own courses, reaching an audience, relevant information and practical skills. This system supports video lectures, interactive content, and community forums to enrich the learning experience, making education flexible and adaptable to each user’s needs.

## 1.3 Personal Motivation

My inspiration for developing this platform stems from the impressive capabilities and success of sites like Udemy and Coursera. These platforms have revolutionized the way people learn by making a wide array of courses available online, breaking down the barriers of traditional education. They offer flexibility and diversity in course selection that caters to all types of learners, from those seeking to improve job skills to others wanting to explore new areas of interest. This inclusivity and accessibility are what I find most compelling.

Personally, the usefulness of these platforms became evident through my own experiences. Being able to learn at my own pace and schedule allowed me to balance my educational pursuits with other personal and professional responsibilities. Moreover, the opportunity to learn from top-notch educators and industry leaders without the geographical and financial constraints associated with traditional schooling proved invaluable. These experiences underscored the potential of online learning platforms to empower others by providing them with the tools and knowledge to succeed in their personal and professional lives. Not only it is beneficial to but Tutors as well where they get earn by creating courses and selling them online.

Driven by these positive impacts, I was motivated to create a platform that would similarly empower individuals around the world, making learning accessible to anyone with an internet connection. My goal is to replicate and enhance the positive experiences provided by platforms like Udemy and Coursera, ensuring that everyone has the opportunity to learn, grow, and achieve their potential.

## 1.4 Objectives

As stated in the project contract there are several objectives that would need to be completed for this project to be a success. To summarize what was said in the contract they are:

* **Research on Existing Systems**: Conduct thorough research on current e-learning platforms like Udemy and Coursera to understand their features, user interface, and functionalities.
* **Propose the System**: Develop a clear proposal for the new system that outlines its purpose, target audience, and key features that distinguish it from existing solutions.
* **Gather System Requirements**: Collect detailed requirements for the system, functional , through analysis of competitor platforms.
* **Decide on the Technology**: Select appropriate technologies for building the system, considering factors such as scalability, reliability, and ease of maintenance.
* **Make Logical Design**: Create a logical design of the system that includes Erd , Activity to make the development process easy.
* **Implement the System**: Develop the system based on the designs like database and requirements previously established, writing code and integrating different software components.
* **Testing the System by Unit Testing**: Perform unit testing to ensure each component of the system functions correctly in isolation and debug any issues that arise, apply validation and error handling.
* **Run the System on the Server**: Deploy the system on a server to ensure it operates smoothly in a live environment.
* **Final Report**: Prepare a detailed final report that summarizes the research, development process, Explain the application, attach the updated designs ,Explain the interesting pieces of code testing

# 2. Research

Before the development of my e-learning system began, the literature I have performed researched deep about the topic .My initial step involved a broad market analysis to identify the major players in the e-learning sector. This analysis highlighted the popularity and extensive course offerings of Udemy and Coursera, which I then chose as my primary focus. I read different articles that helped me analyze the latest E-learning system features and functionality, solution and backgrounds.

Following this, I directly engaged with these platforms by creating accounts, enrolling in various courses, and actively using features such as video lectures, quizzes, forums, and certification processes. This hands-on exploration provided me with valuable insights into the user experience from both the learner's and instructor's perspectives. I also conducted a detailed feature comparison, cataloging specific functionalities such as course enrollment processes, search and navigation systems, communication tools, assessment methods, mobile compatibility, and accessibility features.

To gain deeper insights, I analyzed user reviews and feedback available on forums, blogs, and directly on the platforms. This helped me understand the perceived strengths and weaknesses from a user's perspective.

My technical assessment involved evaluating the underlying technologies of these platforms, including software frameworks, hosting environments, and data security measures, to understand their scalability and robustness. Furthermore, I reviewed competitor analysis reports, which provided a consolidated view of market trends and user demographics. Lastly, I sourced scholarly insights on the efficacy and pedagogical approaches of e-learning systems from academic journals proceedings.

This thorough pre-development research ensured that I had a well-rounded understanding of the existing e-learning landscape, enabling me to design a platform that could truly meet the needs of modern learners and educators.

## 2.1 Functional requirement Analysis

In the first deliverable I have done system requirement analysis were all the requirements are listed and all the requirements are implemented thoroughly throughout the system development process where all the functionalities are covered except one functionality automated assessment and feedback which is not completed because it was too completed for me to make and there was time constrain another things is the admin is system is not included because teacher and students can control the whole system by themselves like creating accounts to managing material so there I thought it necessary besides admin system was not included in the project contract it was just mentioned in the first deliverable. Other than that all the functionalities, including security measures are covered. These are the following functionalities in the current system.

## 2.3 Tooling

In developing my e-learning platform, I selected a suite of tools and technologies that synergize to create a robust, scalable, and user-friendly experience. **JavaScript** was chosen as the core programming language due to its versatility and wide adoption, which is essential for dynamic content management. For building user interfaces, **react** was my choice for its efficient rendering and component reusability, allowing for interactive UI development with minimal redundancy. The server-side runtime environment is powered by **Node.js**, which enables the use of JavaScript on the server and supports efficient, data-intensive real-time operations that are crucial for an e-learning system.

The backend database is managed by **MariaDB**, selected for its performance, advanced security features, and robust support for large volumes of data and complex queries, ensuring reliable management of user data and course materials. For web server software, I opted for **Apache Server** because of its stability, flexibility, and strong security configurations, which are vital for a secure learning environment.

**MAMP Pro** facilitates an easy-to-manage local server environment, which is invaluable during the development and testing phases, supporting technologies such as Apache and MariaDB seamlessly. Development is conducted in **Visual Studio Code**, a powerful IDE that supports a broad range of languages and tools, enhancing productivity with its integrated Git support and extensive plugin ecosystem. **HTML** provides the foundational structure for web pages, supported universally by browsers and manipulated effectively by JavaScript and CSS for a responsive design.

Lastly, **Tailwind CSS** was utilized for its utility-first approach to styling, enabling rapid UI development and ensuring consistency and responsiveness across devices. More Details of tooling are described in implementation.

2.4 Development methodology   
The development methodology I followed for creating the e-learning platform was heavily inspired by Agile principles, specifically adopting a flexible and iterative approach to software development. This methodology emphasizes continuous improvement, flexibility, and the delivery of functional software at the end of each development cycle, known as sprints.

In practice, this involved frequent collaboration and communication with my supervisor. After identifying the initial set of requirements, I began working on the core functionalities of the platform. Each functional component was developed in stages, with regular updates shared with my supervisor. This regular sharing was crucial for maintaining alignment with the project's goals and ensuring that each part of the system met the expected standards and requirements.

After completing a functionality, I would demonstrate the progress to my supervisor for review. This review process was not just a formality but a critical feedback loop. Based on the feedback, whether it involved enhancements, modifications, or even sometimes significant changes, I would revisit the developed functionalities to implement the necessary adjustments. This iterative cycle of development, review, and improvement ensured that the platform evolved in a manner that consistently aligned with the project’s objectives and quality standards. Moreover, this approach allowed for flexibility in handling unexpected challenges or changes in project requirements.

# 3. Design

## 3.1 Erd Diagram

Before implementing the system, especially one complex system platform, it's crucial to have a well-defined blueprint. So, in the first deliverable I have drawn an ERD which gave me an idea how to make database and how the data flow will work but throughout the development process there has been some changes made to the table in accordance with the system requirements. Hence the below picture represents the Updated ERD.

A diagram of a computer

Description automatically generated

A diagram of a person's diagram

Description automatically generated

## 3.2 Activity Diagram

This activity diagram is updated as well

# 4. Implementation

## A screenshot of a computer Description automatically generated4.1 How the project was set up.

Before setting up this project, I ensure that Node.js and npm (node package manager) are installed on your system. Which I downloaded and installed them from [Node.js official website](https://nodejs.org/).

**Note: Same say it was set up for teacher’s system these are example from the student’s system how it is done**

After installing the package manager, I named the student system parents folder to development project as you can see in the screenshot on the left.

A screenshot of a computer program

Description automatically generatedAfter creating the parent directory then I located inside that folder from the console and executed this following code below

**A screenshot of a computer

Description automatically generated**

**Dependencies**

Figure 2 dependencies.

This code created the student-Frontend folder with all the dependencies and script to run the react app, all the dependencies and script can be found inside the package. Json file the screen shot of the dependencies are attached below.

As you can see in the screenshot all the dependencies of the project student-frontend, but these were not all installed at ones these dependencies were installed step wise in development journey.

A screenshot of a computer

Description automatically generated

The above image shows another block called script inside the block. All the scripts are passed to start the react app and perform commands like start, build or test the app. Anyways to run the app we simply need to run the command below in the screenshot.

This command will simply start the app in the local host 3000 port which can be accessed by running <http://localhost:3000/> in the browser screen of the running app is attached below.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

4.2 Font end set up.

A screenshot of a computer

Description automatically generated

**The app router in the frontend**

In the application one of the most important pages is the App.js which is   
The App.js file in a React application is pivotal as it serves as the root component that encapsulates the entire application's structure and routing logic. This specific App.js file sets up the routing, guiding the rendering of different components based on the URL path accessed by the user. This routing is managed using react-router-Dom, a standard library in React for implementing declarative routing.

In this setup, the Browser Router wraps the Routes component, which in turn contains various Route elements. Each Route is responsible for mapping a specific path in the URL to a React component. For instance, accessing "/Login" renders the Login component, and visiting "/Register" brings up the Registration component. This approach allows users to navigate through the application and access different functionalities like enrolling in courses, participating in discussions, or viewing learning materials, all without the need for page reloads.

Moreover, some routes are dynamic, featuring parameters like :courseId, which enables the application to handle user-specific or content-specific data. This dynamism is crucial for creating personalized experiences for users as they interact with various parts of the application, such as discussion forums or specific learning materials.

### 4.2.1 Using Axios

I set up all the libraries required for the font end and one of the most important libraries that I needed was Axios which is a popular JavaScript library that helps to make HTTP requests easily from both web browsers and Node.js environments. The reason why I used it is because it easily manages asynchronous operations thanks to its promise-based nature. This means I can write cleaner, more readable code when dealing with operations that might take some time to complete, like fetching data from a server. Axios also provides interceptors which are great for doing something with requests or responses globally in your app—like adding headers or logging debug information.

A screen shot of a computer code

Description automatically generatedIt is a very handy library that was very important throughout the implementation. It allowed data to and from JSON format, simplifying the process of dealing with data on the web. I used it to send post requests most of the time which is send the body data or any data from the font end and getting result from the backend, here is one example how I used it send post request.

This is one example of sending Axios post request to login backend or end point what it is doing is it is handling login but transferring the username and password passed as Json parameter and then retrieving the response.data.message from the response if the response if true then it will navigate to Browse Course Page or else catch error and show error in the console of the browser and also set the error message in the app as well. This is a basic implementation of Axios in the system which has been used many times throughout the system. (Axios, 2021)

### A screen shot of a computer Description automatically generated4.2.2 Tailwind CSS

For the webpage I have utilized a styling library called Tailwind CSS, a highly customizable, low-level CSS framework that uses utility classes to expedite styling directly within HTML or JSX. This approach is highly beneficial as it allows for rapid development and maintains consistency across the UI without the complexities of traditional CSS setups. Tailwind employs a utility-first approach where styles are directly applied in the markup using predefined classes, facilitating faster development as there is no need to toggle between CSS files and HTML, and it promotes a more consistent design with less effort. Additionally, Tailwind is designed to be responsive out of the box and offers extensive customization options, allowing me to tailor the framework to fit the specific design needs of my project precisely. I can define my design system—including colors, fonts, and sizes—within Tailwind’s configuration. Furthermore, Tailwind integrates seamlessly with tools like Purge CSS to remove unused CSS, ensuring that my production builds are lightweight and quick to load. Here on relevant part where I have used tailwind and how simple and easy to use show in screen shot below. (css, n.d.)

A computer code on a white background

Description automatically generated

## 4.3 The Database and Backend Set up.

To set up the backend for the system, the development began with establishing the database infrastructure. I chose MAMP Pro as the local development environment because it offers a simple, streamlined way to manage the server and database components on both MacOS and Windows. MAMP Pro includes Apache server, MySQL, and PHP, which are needed for the backend development.

After installing MAMP Pro, I proceeded to create the necessary databases for the system using SQL queries. This involved defining the schema, which included tables for students, courses, enrollments, and other relevant entities, the screen shot the tables are attached below. Each table was designed to store specific data types and relationships that are integral to the system's functionality. For example, the students table might contain columns for student ID, name, email, and course enrollments, each with appropriate data types and constraints to ensure data integrity and accuracy.

A screenshot of a computer

Description automatically generatedThis is how the table structure looks like in the learning system database. Below are examples of how the database and tables were created using SQL

### A screenshot of a computer program Description automatically generated4.3.1 The SQL Script

Here is two examples with an explanation of how SQL used in the system. The SQL script provided outlines the creation of two database tables, teachers and users, each designed to store essential information about individuals in the system. The teachers table includes fields such as teacherID, username, name, password, email, created\_at, and updated\_at. The teacherID serves as a unique identifier with a VARCHAR(255) data type, ensuring each teacher has a distinct record. The table tracks when each record is created and updated with timestamp fields that automatically populate with the current time, and all fields except timestamps are required to be non-null, ensuring complete data entry. Similarly, the users table contains fields like userID, username, name, password, email, alongside created\_at, and updated\_at with the same characteristics as in the teacher’s table. Additionally, it includes an enrollmentID which is a non-null field linked as a foreign key to an enrollments table, indicating a relational link that likely stores specific enrollment details. This structure not only facilitates user management within the system by capturing and maintaining comprehensive user data but also supports functionalities such as authentication and record tracking, providing a robust framework for managing interactions and data integrity within the educational system.

### A screenshot of a computer Description automatically generated4.3.2 Node.js Setup with file structure

After successfully setting up the database for the system using SQL and ensuring that the necessary tables were in place, the next step involved setting up the backend server using Node.js. This setup is crucial as it provides the necessary API endpoints that the front-end application will use to interact with the database effectively.

The setup began with initializing a new Node.js project which is called student backend in the same parent folder the development. This was done by navigating to the running npm init in the terminal, which creates a package. Json file to manage project dependencies, scripts, and metadata. This file is essential as it outlines the project's dependencies and other configurations. The screen shot is placed left of the file structure.

After initializing my Node.js project, the next critical step was to install several key packages essential for building a robust web server. First, I installed **Express**, a flexible and minimalist web application framework for Node.js, which simplifies the creation of servers that can manage routes, requests, and views efficiently.

### 4.3.3 MySQL set up.

In setting up the MySQL dependency for my Node.js project, I started by installing the MySQL driver using npm, which allows Node.js to communicate with MySQL databases. The command npm install mysql added the MySQL package to my project, integrating it into the package.json file. Following the installation, I configured the database connection by setting up a connection object in my code, specifying the database's host, user credentials, and database name using the .env file which I will talk about in the security measures. This setup involved creating a connection using mysql.createConnection() and establishing the connection with connection.connect(), which also handled any potential errors during the connection process, below is a example how the connection was set up.

### A screenshot of a computer Description automatically generated4.3.4 The end points and routing

This figure shows a folder called router which contain all the logic of the server request and response, This folder appears to contain a collection of JavaScript files, each representing a different route handler in the backend of a student management system. Each file is responsible for handling specific types of requests related to its named function within the application.

A screenshot of a computer code

Description automatically generated

### 4.3.5 Node.js Example Code

The code on the left is part of a Node.js module that fetches a list of courses a user is enrolled in from a database. It starts off by pulling in a connect function from a utility module, which is used to establish a connection to the MySQL database. In this particular function, EnrolledCoursesView, a user ID is hardcoded (in this case, set to 1 for simplicity), which typically wouldn't be the case in a production environment where the user ID would dynamically match the logged-in user.

A SQL query is then constructed to pull the course ID, name, and instructor from the courses table, linking it through an inner join with the enrollments table to ensure only courses that the user is enrolled in are fetched. The database connection is activated, and the query is executed using a promise-based approach, which makes the asynchronous code cleaner and easier to manage.If the query successfully retrieves the data, the server sends back a successful response with the course’s data. If something goes wrong, like a connection issue or a syntax error in SQL, it catches the error, logs it for debugging, and tells the user that an internal server error occurred.

### 4.3.6 How the end points were set in the App.js file by using express.js.



A close-up of a text

Description automatically generated



In the app.js file of student backend, I set up an Express server to handle specific application routes, including a route for viewing enrolled courses. Initially, I created an instance of an Express application with const app = express();, which serves as the backbone for adding middleware and routes. I then imported a function called EnrolledCoursesView from a separate file within the Routes directory. This function is responsible for fetching the list of courses a user is enrolled in. I configured the server to handle POST requests to the /EnrolledCoursesView endpoint by setting up a route that invokes the EnrolledCoursesView function whenever this endpoint is accessed. Notably, there seems to be a small oversight in the order of parameters passed to this function; typically, the request object (req) should be the first parameter followed by the response object (res), which might need correction. This setup demonstrates a clean and modular approach to handling server routes, making the application scalable and easier to manage by separating concerns and allowing individual features to be modified independently.

A screenshot of a computer program

Description automatically generatedAfter setting up the Express server in the app.js file, I utilized Nodemon to run the server locally on port 3001, enhancing my development process significantly. Nodemon is a handy tool that monitors changes in the project files and automatically restarts the server, eliminating the need to manually restart it after every change. To integrate Nodemon, I first installed it as a development dependency using npm install --save-dev nodemon. Then I am able to start the server by running nodemon app.js script the example is shown in the screen shot left.

## 4.4 The Security measures

## A screenshot of a computer Description automatically generated4.4.1 The dotenv file

For secure environment management, I used **dotenv**, a module that loads environment variables from a .env file into process.env, ensuring sensitive information like database URLs and private keys are managed securely.

### 4.4.2 The middleware.

In my Express-based server setup, I've incorporated essential built-in middleware to effectively handle incoming HTTP requests. I utilized express.json(), a middleware that parses incoming requests with JSON payloads. This is vital for my server to correctly interpret JSON formatted data, which is a standard format for data exchange in modern web APIs. Additionally, I included express.urlencoded({ extended: true }), which parses incoming requests with URL-encoded payloads. The extended: true configuration employs the qs library, enabling the parsing of complex nested objects and arrays within URL-encoded data. This capability is particularly important for managing form submissions or POST requests that carry data in the HTTP body, ensuring that my application can seamlessly handle and process diverse client-sent data structures. These middleware choices are integral to the functionality of my application, enabling robust data handling and interaction capabilities.

### 4.4.3 The Cors

I added the **Cors** package to enable and configure Cross-Origin Resource Sharing (CORS), which is vital for controlling how resources on my server can be requested from different origins, thereby enhancing security and flexibility in client-server interactions.

### 4.4.4 Using JWT’s

A screenshot of a computer code

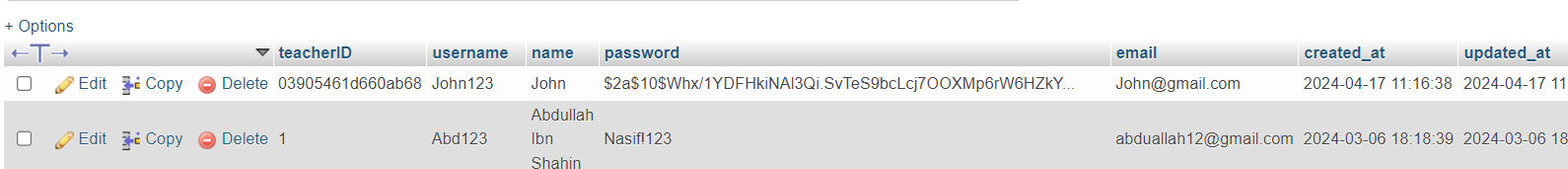
Description automatically generatedIn the teacher system I have used jsonwebtoken library to generate and verify JSON Web Tokens (JWTs), offering secure data transmission between parties. By using environment variables to store the signing secret, sensitive information is safeguarded against exposure in the source code, preventing potential leaks if the code is shared. JWTs generated through this approach facilitate user authentication and authorization by encoding user-specific data, like roles and permissions, into a signed token. This ensures that only authorized users access specific resources. The tamper-resistant signature included in each token provides integrity, confirming that the token hasn't been altered and was issued by a trusted source. Tokens are set to expire after a specified period, reducing the risk of abuse if intercepted or compromised. The stateless nature of JWTs eliminates the need for server-side session management, while their portability enables secure communication across multiple servers or services. Overall, by keeping the signing secret secure, using reliable cryptographic algorithms, and ensuring short token expiration times, this implementation establishes a robust authentication and authorization mechanism for applications.

### 4.4.5 The hashing

A screen shot of a computer code

Description automatically generated

In the system I have also hashed the password using the bcrypt library to hash passwords, which is a crucial security practice for protecting user information. This transforms the transforms a plain text password into a fixed-length string of characters that doesn't reveal the original password. This is done using a cryptographic hash function. bcrypt is particularly effective because it incorporates salt (a random value) to ensure that each hash is unique, even if the same password is hashed multiple times. This helps to defend against certain types of attacks, such as rainbow table attacks. Here is a screenshot from database how the hashed password looks like.



## 4.5 The authentication flow

### 4.5.1 The student system authentication

A screenshot of a computer

Description automatically generated

A computer screen shot with a lock

Description automatically generatedThe student flow authentication flows starts from the registration page where the use can register their details name, username, email address, password and a confirm pass. The user must fill the all the details to proceed to the next step and then click the register button. If the user has already registered, they can click on the existing member label and login from there.

The above image is the login page from the authentication, user can enter their username and password and then proceed to login which will take them to the login dashboard of the system. The user can check box remember me in which they don’t have enter the login details again it will appear automatically.

Ones the user is logged in it will take them to their dashboard page and until they logout again they will be logged in so when default page loads it will directly take them to their dashboard page without login again.

### 4.5.2 Using Session for Authentication

A screenshot of a computer code

Description automatically generated

In the student system, I used the express session which is a middleware is vital for handling user sessions, a core component for maintaining user authentication across various parts of the application. The session middleware is configured with a secret key, which is used to sign the session ID cookie. This signing process is crucial for enhancing security because it helps to ensure that the session ID cannot be tampered with by external parties. This security measure is particularly important because the session ID acts as an identifier for authenticated users, allowing them to access multiple parts of your application without having to repeatedly authenticate.

### A screenshot of a computer program Description automatically generated4.5.3 How the session was used in the login.

This is session is used in login function allows verifying user identity and maintaining their logged-in status across sessions. When a user logs in successfully, as indicated by a password match, the system initiates a session and sets key variables: the user's ID (userId), their username (username), and a loggedIn status to true. These variables are stored in the session, allowing the server to recognize and authenticate the user on subsequent requests without re-checking credentials. This continuity is facilitated by a session ID cookie set in the user's browser upon login. This cookie, managed by express session, is automatically sent with each request to the server, which retrieves the corresponding session and validates access based on the logged In status. Thus, sessions bridge the gap between the stateless nature of HTTP requests and the need to maintain a secure, continuous user experience.

## A screenshot of a computer Description automatically generated4.6 The course browsing.

In the system I have implemented a live searching feature. When I typed "mo" into the search bar, it automatically displayed a dropdown menu with relevant search results, including courses like "Mobile App Development," "Mobile App Development 101," etc.

This search functionality seems to be crafted using a mix of front-end and back-end technologies. As I type into the search bar, an event listener likely triggers a request to the back end. The implementation of this code of code explained in the “Interesting Pieces of Codes Chapter” since I found this code Interesting. Click here to navigate to that. [(Live Searching code)](#_4.9.1_The_Live)

**Search Track**

A close-up of a hand holding a phone

Description automatically generatedAfter search is complete if the user returns back to the dashboard where all the course recommendation like explained in the previous section of the search view same way it recommends the courses based most popular enrolments. The feature utilizes search tracking to enhance user experience.

A screenshot of a computer

Description automatically generatedDuring the registration process, users are prompted to select their topics of interest. This can range from specific subjects like "Mobile App Development" to broader categories like "Web Development." This selection is important as it informs the platform of the user’s preferred areas of learning as I have personalized learning recommendations in the system.

A screenshot of a computer

Description automatically generated

Once registered, the user is directed to their dashboard, which includes personalized course suggestions based on the interests they selected. This tailored approach helps in creating an engaging and relevant educational experience from the start. The section is called Top Ten Suggestions: The platform uses the user’s specified interests to generate a "Top Ten" list of course suggestions. These recommendations are likely powered by an algorithm that matches the platform’s available courses with the user’s interests. This algorithm consider factors such as course popularity by enrollment, relevance to the interested topics, user ratings, and perhaps even the historical preferences of similar users. The implementation of this code is also explained in the “Interesting Pieces of Codes Chapter” since I found this code Interesting as well. Click here to navigate to that, It shows how the how the algorithm was designed in both frontend and backend. Click here to navigate there ([Recommendation Algorithm Code](#_4.9.3_The_recommendation))

## 4.7 Course enrollments

### A screenshot of a computer Description automatically generated4.7.1 Course View

This is the course page on the platform provides a quick introduction to what the course will cover. Below this, the course boasts a high rating of 4.60 from student reviews and shows that 46 students are enrolled, highlighting its popularity and the trust other students have in it. The other details course is created by whom, which adds credibility to the content. In the pricing section, the course is priced at $129.99, clearly displayed above a conspicuous blue "Add to Cart" button that invites users to enroll.

### A screenshot of a computer Description automatically generated4.7.2 Shopping Cart

After browsing in the previous page when the user decides to add the course to his cart it has added the "Web Development Basics" course. The layout has featuring a "Remove" button for easy removal of the course if the user changes their mind. The total price, identical to the course price, is displayed on the right, along with a "Checkout" button, directing users towards completing the purchase.

### A screenshot of a computer Description automatically generated4.7.3 The navigation of the course view

After the purchase user can see the navigation menu "Module Home" for accessing comprehensive learning materials, "Discussion Forum" for interactive engagement with peers and instructors, and "Take Test" for evaluating their understanding through quizzes and tests. Additionally, a "Go to Course" button with a welcoming message invites students to begin their learning journey, providing direct access to the course content.

### A screenshot of a computer Description automatically generated4.7.4 The discussion forum

A white background with blue lines

Description automatically generated with medium confidence

The discussion forum featured to provide collaborative learning environment where students can engage deeply with course material and each other. In the forum, students can view a list of posted questions, such as those displayed in the interface, with each entry showing a brief overview, the time it was posted, and the latest activity. By clicking on "View body and replies," students can access the full discussion thread, reading through detailed questions and the responses they've garnered.

Additionally, the platform facilitates active participation through an "Ask Question" feature, where students can post their inquiries. This tool is enhanced by the ability to attach visual aids or documents to their posts, as indicated by an attachment icon next to the message box. This capability is invaluable for subjects that require visual explanations or when students need to share specific issues they are encountering, such as code snippets or error messages. After drafting their question and attaching any necessary files, students can submit their contributions to the forum with a simple click on the "Send" button.

A screenshot of a computer

Description automatically generated

### A screenshot of a computer Description automatically generated4.7.5 Take Test

This is the test page which displays a quiz interface where students can engage in testing their knowledge through timed questions. In this example, the page presents a question about JavaScript .The students are given multiple-choice options. Notably, the page also features a timer, counting down from an 60 start time to zero. This helps students practice managing their time effectively during tests. Once an answer is selected, the student can click the green A screenshot of a computer

Description automatically generated"Next" button to proceed to the following question, facilitating a smooth transition between test items and keeping the student engaged with the assessment process and at last it shows the overall score.

### A screenshot of a computer Description automatically generated4.7.6 The Learning Material

These are the Learning materials. On the left side of the page, there is a navigation panel labeled "Topics," which organizes the course content into collapsible and expandable sections such as "Introduction to Kotlin," "Videos," "Download Slides," and "Kotlin Operator." Which is dynamically created from database The code is provided below how it was implemented. This structured layout allows students to easily access specific topics by expanding each section to reveal additional resources like video lectures and downloadable slides.

In the main view, there is a featured video displayed of the topic. Below the video, options to download slides.

## A screenshot of a computer Description automatically generated4.8 Teacher system course Management

The above showcases the Teacher's Dashboard, designed to offer teachers a streamlined view of their activities and tools for managing and creating courses. This allows quick access to all the navigation possible the system has. Also show a overview of his created courses and their details.

### A screenshot of a computer Description automatically generated4.8.1 Course Creation

A screenshot of a computer

Description automatically generatedThis is course creation. Teachers are required to enter the initial details of a new course they wish to create that is asking for the course's name, a brief description, the category under which the course falls, and an option to upload an image for the course. These fields are crucial for setting up the basic identity and classification of the course, making it easier for potential students to find and understand what the course offers at a glance.

The next course creation process where teachers can provide more specific details about the course. This includes entering the course length, setting a price, selecting the difficulty level from a dropdown menu, choosing the language in which the course will be taught, and describing the expected outcomes for students who complete the course. These details are essential for students to gauge whether the course matches their learning goals, availability, and proficiency level.

### A screenshot of a computer Description automatically generated4.8.2 Content Creation

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedTop of Form

A screenshot of a computer

Description automatically generated

These screenshots show content management options that include adding notes and assessments. The "Add Notes" option likely allows teachers to provide supplementary Videos and presentation Slide of the course topic. The "Add Assessments" option is for adding assessment such as test.

Then this form include fields for entering the title of the content, a detailed description, and options for uploading files or videos. Like I said earlier, this setup allows teachers to seamlessly integrate various types of educational resources, such as PDF documents, PowerPoint slides, or instructional videos, directly into their courses.

### A screenshot of a computer Description automatically generated4.8.3 The content, course view and management

After content has been added to the course In this example, the content for the "First Week" of the course is displayed, labeled with a brief description "Good Course" under a section heading. Below this, there is a video player integrated into the page containing educational content relevant to the first week's topics. There is functionality for "Edit" and "Delete" adjacent to the video player, giving teachers the flexibility to modify or remove content as needed.

A screenshot of a computer

Description automatically generatedAdditionally, tabs labeled "Content" and "Assessments" at the top of the page suggest that teachers can toggle between viewing the course's instructional content and assessing materials, ensuring they have control over all aspects of the course structure. A button labeled "Add Content" is prominently displayed, indicating that teachers can continue to augment the course with new materials throughout the duration of the class.

**Just like the content, the course details can be viewed and edited as well.**

## 4.9 Interesting Piece of Code chapter

### A screenshot of a computer Description automatically generated4.9.1 The Live Search Implementation

Previous described in the screen shot of live searching bar here it shows the live searching implementation. The provided code snippet integrates a backend live search with the frontend React component to show the live searching bar.

**The backend function:** Recommendation is designed to handle search queries from the frontend. It starts by validating the incoming search keyword from the request's body. If the keyword is missing or invalid, it returns an error response. If the keyword is valid, it constructs a SQL query to search the courses table for any course whose category, name, or language matches the keyword. The search is performed using the SQL LIKE operator, which allows for partial matches, enhancing the flexibility of the search function. The query limits the results to the top 10 matches, focusing on relevance and performance. The function then executes the query, retrieves the results, and sends them back to the client in a successful response. If an error occurs during this process, it logs the error and returns an internal server error response. The function ensures the database connection is properly closed in a finally block to avoid resource leakage.

**A screenshot of a computer

Description automatically generated****Frontend Functionality of the live searching:**

On the frontend, the Search View component in React manages the state and interaction of the search feature. It uses hooks like useState for managing local states like the search keyword, courses list, and filter options, and useEffect for reacting to changes in state or performing side effects like fetching data.

Hence in the front end the implementation live searching SearchResultBox, is designed to handle search operations and display search results dynamically on a web interface. It accepts a keyword prop, which triggers a sequence of operations to fetch and display unique categories and names from a search index.

A screenshot of a computer

Description automatically generatedHere’s a detailed breakdown of the component’s functionality:

1. State Initialization: The component initializes three pieces of state using the useState hook: resultSearch holds the raw search results, categories holds processed data for unique categories derived from the search results, and names holds unique names from the search results.
2. Effect Hook for Search Triggering: An useEffect hook listens for changes to the keyword prop. If a keyword is present, it calls the handleSearch function which performs the actual search operation.
3. Search Functionality (handleSearch): This function uses Axios to send a POST request to a backend endpoint (http://localhost:3001/SearchBarRecom) with the keyword. On receiving a successful response (HTTP 200), it logs the data for debugging, updates the resultSearch state with the raw data, and then processes this data to extract unique categories and names using the getUniqueCategories and getUniqueName functions.
4. Utility Functions for Unique Entries:
   * getUniqueCategories: Extracts unique categories from the search results to avoid redundancy in the UI. It uses a JavaScript Set to ensure uniqueness efficiently.
   * getUniqueName: Similarly, extracts unique course names from the search results.
5. Navigation Handling: The component includes two functions, handleCategoryClick and handleNameClick, which are triggered when a user clicks on a category or a name in the UI. These functions set the searchKeyword in localStorage and then navigate the user to the /SearchView route. This is intended to refresh the search view with the new keyword, this is helpful in tracking searches.
6. Rendering: The UI of a floating division that renders the unique categories and names as clickable items. Each item, when clicked, triggers the corresponding handler function which goes to the search view and shows the results based on the searched keyword.

### A screenshot of a computer code Description automatically generated4.9.2 The Search engine code

In the search view there is a prioritization of course search on different details where name being the first priority then description , category and then course language .At first It extracts a search keyword from the request's body and splits this keyword into individual words to enable a more granular search across several fields of the courses table, including course name, description, category, and language.

For each keyword, it constructs WHERE clauses that check for the presence of the keyword within these fields. These clauses are then combined with an OR operator, allowing the database query to return courses that match any of the keywords in any of the specified fields. Additionally, a CASE statement within the SELECT query assigns a priority to each result based on the field where the keyword was found, prioritizing matches in the course name over other fields. This helps in sorting the search results such that more relevant matches are displayed first.

### 4.9.3 The recommendation algorithm in the backend

A screenshot of a computer

Description automatically generatedThe provided code snippets is another interesting another piece of code in the for Recommendation system showing how it is implemented.

A computer screen shot of text

Description automatically generated

First, it is utilizing a database connection to fetch and process data related to course recommendations based on student interests. Here's a breakdown of each function and its role within the system:

1. TopTenCoursesEnrolledSelectByCategory Function

This asynchronous function is responsible for fetching the top ten courses based on enrollment numbers within specified categories. It interacts with a MySQL database using the connect function imported from '../Utils/Db.js'.

* Parameter Validation: It starts by checking if the categories parameter is present, is an array, and is not empty. If any of these conditions fail, it sends a 400 Bad Request response with an error message.
* Query Construction: It constructs a SQL query string to select course details. The categories are mapped to SQL-safe strings because all the categories are returning in and are included in the SQL IN clause to filter courses by their categories. The courses are then grouped by courseId and courseCategory, ordered by the number of students enrolled in descending order, and limited to the top ten. The query will return all the course based on courseCatagory which is crucial to show all the course related to that that category.
* Query Execution: The function uses the database connection to execute the query asynchronously. If successful, it sends back a 200 OK response containing the course data. If there's an error during the execution, it logs the error and returns a 500 Internal Server Error.
* Resource Management: Finally, it ensures the database connection is closed, whether the query succeeds or fails, to prevent resource leaks.

2. GetInterest Function

This function aims to retrieve a student's interests from the database and then fetch courses matching these interests.

* Fixed Student ID: It currently uses a hardcoded studentid of 1, which implies it fetches interests for this specific student. In a real-world scenario, this would be dynamically retrieved, likely from session data.
* Fetching Interests: It connects to the database, retrieves the student's interests using a prepared SQL query, and then extracts these from the query's result.
* Flattening Data: After fetching, it flattens the array of interests since the result might be nested due to the structure of the returned data this is done because if the data that it is going to check against is array.
* Fetching Courses: It then calls TopTenCoursesEnrolledSelectByCategory using the flattened array of interests to get relevant courses.
* Error Handling and Resource Management: Similar to the previous function, it includes comprehensive error handling and ensures the database connection is closed in the finally block.

A screenshot of a computer program

Description automatically generated**The front end code : shows how recommendation is shown on the client side on the Top ten courses Section on the browsing course**

Data Fetching with useEffect:

The useEffect hook runs initially when the component mounts and subsequently whenever the reloadCourses state changes. Inside the hook, the fetchTopCoursesOnInterest function is called to retrieve courses tailored to the user's interests. Additionally, the hook checks for the presence of a saved search keyword in local Storage which is used to track search and recommendation based on the keyword this refers to Seach engine code in the previous section.

**Grouping Courses by Category:**

The groupCoursesByCategory function organizes courses into categories. It iterates over an array of course objects, checking if a category has already been initialized within a groupedCourses object. If not, it initializes a new array for that category. Each course is then added to its respective category array with structured data including ID, image, title, description, enrollments, rating, and a click event handler (sliderClick). This categorization aids in presenting courses on category since recommendation shown based on interests.

Rendering Courses:

By Category: The renderCoursesByCategory function utilizes the categorized courses data to render UI elements. For each course category, a title is displayed with the category name, followed by a slider component (CoursesSlider) that visually presents courses within that category. Each slider is populated with course data enhancing recommendation experience.

A screen shot of a computer code

Description automatically generated

### A screen shot of a computer code Description automatically generated4.9.4 Take Test Code.

This is another interesting piece of the the font end React “ take test” component code utilizes several hooks for effective state management and UI control in a in the take test page. It employs useState hooks to manage the current question index (currentQuestion), the user's selected option (selectedOption), the user's cumulative score (score), whether to display the score (showScore), and a countdown timer (timer) for each question set initially to 60 seconds. The useEffect hook monitors changes in the timer and showScore, decrementing the timer each second until it reaches zero for elimination of test, at which point it transitions the UI to display the final score, signaling the end of the quiz. User interactions are handled through handleOptionSelect, which updates the selected option upon user clicks, and handleNextQuestion, which checks if the selected option is correct to adjust the score accordingly before moving to the next question or concluding the quiz if all questions are answered. This setup ensures a dynamic and interactive quiz experience, efficiently handling user inputs, timing, and transitions between questions and final scoring. Below is the html code showing how it works there

A screenshot of a computer code

Description automatically generated

# 5. Testing Chapter

## 5.1 The unit test

In the system testing I performed Unit testing which was useful because the architecture of my each system was component based it allowed me to test efficiently, particularly for maintaining high-quality code during the development process. By isolating and testing individual units of source code—like functions or methods unit tests ensure that each component behaves as expected, I mainly used it in the backend which maintaining connection with the database. Before creation the backend I made sure to follow the test-driven method in which I wrote the test early and then wrote the code. Some screenshots are attached how the unit testing was testing was used.

**How it helped me:**

1. Early Bug Detection: When wrote the code I was expecting an right output from database writing unit test helped me detect bugs
2. Development Efficiency: It helped me to reduce the time spent on debugging and manually verifying functionality, especially as when the application was getting bigger

**Vitest for JavaScript Testing:**

For JavaScript applications, choosing the right testing library is key to achieving effective unit testing. Vitest is a modern unit test framework that has gained popularity for testing JavaScript projects, particularly those using frameworks like Vue or React. (Vitest, 2024)

Key Features of Vitest:

* Performance: Vitest is designed to be a fast unit test framework, utilizing Vite's native ES modules for incredibly fast startup and updates. This is particularly useful in larger projects where traditional tools might slow down due to heavy processing.
* Environment Emulation: Vitest can run in both node environments and simulated browser environments using Happy DOM. This makes it highly versatile for testing frontend JavaScript as well as Node.js backend code.
* Integration: Vitest integrates seamlessly with modern front-end development workflows, supporting popular frameworks and tools out of the box.

## 5.2 Validation

Before implementing form I made sure Validation to validate form and input field especially in pages like discussion forum and file uploads which is crucial. It was very crucial for maintaining data integrity and ensuring that the application behaves as expected. (Geeks for Geeks, 2024)

**Key Aspects of Validation:**

1. Input Validation: Ensures that all user inputs or data fetched from external sources meet predefined formats and constraints.
2. Business Rules Validation: Enforces specific business logic rules
3. Security Implications: Proper validation is critical for security. It helps prevent common vulnerabilities like SQL injection, cross-site scripting (XSS), and other malicious data input attacks.

## 5.3 Error Handling

If you take a look at my backend code, you will find that I have added try and catch block in almost every function it was very crucial helping me to find errors inside execute code also allowed used to manage and respond to runtime errors in a controlled and predictable manner. It helped me to handle possible errors and prevents the application from crashing and for providing users with understandable feedback when something goes wrong. Below is an example.

**Key Aspects of Error Handling:**

1. Try-Catch Blocks: These are used to catch exceptions that occur during program execution and prevent the program from crashing. Developers can define specific responses to different exceptions.
2. Logging: Errors and exceptions should be logged to an external system. This helps in diagnosing issues by providing a historical record of what went wrong and under what circumstances.
3. Fail Safely: In case of failure, the system should revert or rollback to a safe state where no corrupt data is saved, and no critical operations remain partially completed.
4. Graceful Degradation: In scenarios where not all system features can function correctly (perhaps due to third-party service failure), the system should degrade gracefully, allowing users to continue using other features.

# 6. Critical Evaluation

[In section 1.4](#_1.4_Objectives) , it's noted that achieving the objectives outlined in the initial contract is crucial for the project's success, as failure to meet these objectives would deem the project unsuccessful (the complete list of objectives is detailed in Table 4 of the appendix). The contract's aims are also essential components of the system. In summary, the project goals were to create an E-Learning web Application with a frontend developed in ReactJS, a backend with Node.js, and MySQL as database.

- add the objectives

Analysis shows that all these aims have been met, indicating the project's success. While going based off the initial aims and objectives could be counterintuitive, for example, if the project had serious changes halfway through where in my case there has been only change in the name of the project, and one functional requirement got replaced other than the E-Learning application “Brain Zone” has remained as the same idea and principle all the way from the start.

## 6.1 The good

Since the project has been finished there have been many successes throughout the development process. Since the project was very big I thought I wont be able to finish on the times scale but it was finished on time scale. Some functions which were crucial throughout the development was difficult to implement but at the end I managed to complete them. At the end of the product works perfectly fine following industrial standards, I take a great deal of pride in saying this has been my project.

## 6.2 The Enhancement

Despite everything going according to the aim there are some improvement such as the designs which could better in section such as progression. There are other improvements in functionality which could be made such as adding AI functionalities that automates processes like auto assessment systems and adding a chatbot for asking question to it regarding the topics or course but these functionalities are just modern version of E-Learning applications would bring innovation , however that would be future development due to time constraints. In authentication a few improvements could be made such as adding password reset and email verification, but I skipped it because I had to focus on other code functionalities ones again due to time constraint. The file structure or architecture was modular, but the naming of the files could be better for cleanliness. Personalized learning could be enhanced enabling the use experience more personalized. Non-functional requirements such as page loading is good but can be enhanced by other better structure. In the backend retrieving data from database is linear since it is being populated in an array first which might caught time delay when the user increase but in this case the user is very limited , in future development the data structure and algorithm can be improved.

# 7. Conclusion

The project has proven to be an enjoyable and instructive experience. The knowledge gained throughout this and past year is expected to significantly benefit future career opportunities. A client/server web application application was effectively developed, enabling simultaneous connections. Although there were a few minor setbacks and areas that might benefit from refactoring, the code performs effectively and fulfills the initial objectives established at the project's onset.

# References

Axios. (2021, July 13). *How to use axios with react*. Retrieved from https://www.freecodecamp.org/news/how-to-use-axios-with-react/#:~:text=Here%20are%20five%20reasons%20why%20you%20should%20use,the%20server%20as%20well%20as%20the%20client.%20

Cristin Perry, L. H. (2024, May 9). *Online Learning Platforms: What Are They And What Are The Benefits?* Retrieved from forbes: https://www.forbes.com/uk/advisor/education/online-learning-platforms/

css, t. (n.d.). Retrieved from https://dev.to/haszankauna/using-tailwind-css-with-reactjs-a-concise-guide-33j

*Geeks for Geeks*. (2024). Retrieved from Validation.

*History of E-Learning Platform*. (2023, January 26). Retrieved from Academy Ocean Blog: https://academyocean.com/blog/post/history-of-e-learning-system#:~:text=While%20the%20question%20of%20who%20invented%20online%20classes,Illinois%20and%20developed%20a%20program%20for%20computer%20learning.

Vitest. (2024). Retrieved from https://vitest.dev/guide/

# Appendix