

EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2



Computer Assisted Multiple Choice Exam System

Mukhammadkadir Zainabidinov

194140

The project which I have developed is called computer-assisted multiple choice exam system. It is a website application that I have created with the main idea for providing people to take exams online from any location who have a connection to the internet. According to my web application, I have created three user roles that interact with the website, those users are students, teachers, and an admin for administering web application parts. The main goal of the website is to ease the job of teachers who create exams and for students who can take exams online from any convenient location they prefer.

Supervisor

Zafer Erenel

Publish Date

Table Of Contents

Mukhammadkodir Zainabidinov.....	1
194140.....	1
Zafer Erenel.....	1
Publish Date.....	2
Introduction.....	3
1.1 Problem definition.....	3
1.2 Goals.....	3
2. Literature Survey.....	4
3. Background Information.....	5
3.1 Required & Used software.....	5
3.2 Other software.....	5
4. Design Documents.....	6
4.1 Data flow diagram.....	6
4.2 Database diagram.....	8
5. Methodology.....	9
6. Conclusion.....	30
6.1 Benefits.....	30
b. Benefits to me:.....	30
6.2 Ethics.....	31
6.3 Future Works.....	31
7. References.....	32

Introduction

1.1 Problem definition

- Accessibility problems arise when it comes to exams that make students go to a specific place to write an exam however with online exams students have no need to travel to an examination place and can easily write wherever they want.
- Time problems, traditional exams require more effort to organize a place and time to conduct an examination of students especially when it comes to paperwork of exams. However, it is much less work with online exams for examiners that with a few clicks and exam questions can organize an online exam which is very convenient for teachers.
- Immediate feedback, when people take online exams, they answer for questions, and at the end of their exam they have a chance to learn about their performance about how well they did during the exam this is a good part of online exams that by learning their strengths and weaknesses instantly they get to know where they should work more, unlike the traditional exams that the student due to exam stress may forget how well they did with their exam.
- Cost-effectiveness is one of the main benefits the online examination systems can bring for example there are exams that make people to travel to other countries spending extra time and resources but more importantly it can be impossible for some people who cannot afford that or even have no ability to travel due to disabilities they may have but with online exams, they can take exams from the comfort of their own homes.

1.2 Goals

In my application I was aiming to achieve the following goals so it benefited users:

- To provide students with immediate feedback upon completion of their exams, assisting them in determining their strengths and weaknesses, so students can better understand these areas when received with immediate feedback.
- Provide with a secure and efficient way of conducting exams to a large number of students in order to ensure a fair grading process.
- To automate the grading process so it can save time for teachers by not checking and grading every student's work.

2. Literature Survey

Compare1: I have tried some examination applications that one of them is called ExamSoft software which is similar to mine but my project allows users with more customization features for exam questions like they can create multiple-choice, true-false, and fill in the blank type of questions and answer choices.

Compare2: Most examination systems online do not allow the student to learn their exam grades and the correctness of their answers, in my project, however, it lets students how well they did in exam by pointing out where they did wrong, correct, and even where they missed a question without giving any answer or selecting an option from answer options.

Compare3: Unlike other examination applications my project stands out with its user-friendly interface and experience for both teachers and students. With the help of an intuitive user interface, the teachers and students can easily navigate in the website without any burden so with that students can check their results and see their performance.

3. Background Information

3.1 Required & Used software

React.js:

For client-side I chose to use React.js as it is very efficient with working states and rendering Of components with the help of virtual DOM (Document Object Model)

Nodejs:

For server-side I used Node.js for development of my project because it makes development fast and efficient due to its good handling asynchronous processes.

- **Express.js:**

Express.js is a Node.js framework, this tool was used to handle HTTP requests that acts as a middleware between client and server sides of web application.

- **MongoDB:**

For storing data in the database, I utilized MongoDB which data is stored in JSON-like format and uses NoSQL that data is stored differently than SQL relational database systems.

3.2 Other software

- **Figma:**

For designing the appealing User Interface and User Experience I used Figma, well-known tool for creating good UI & UX designs.

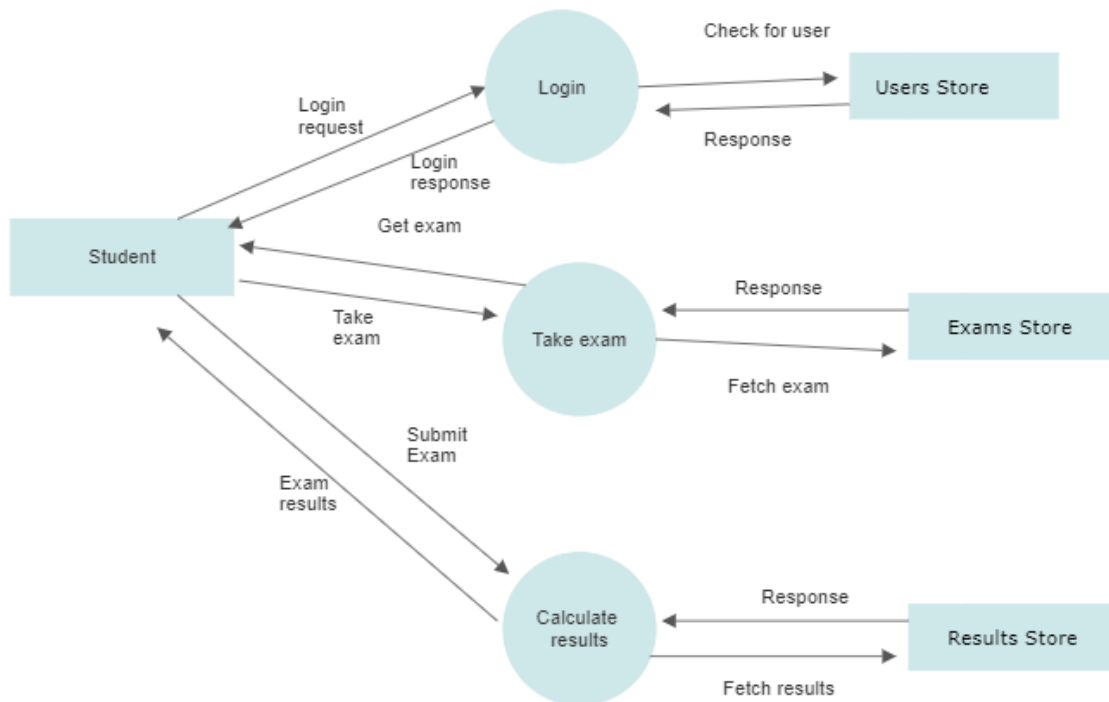
- **Git & Bitbucket:**

For version control of my code, I used these two technologies that help to easily track the code changes made.

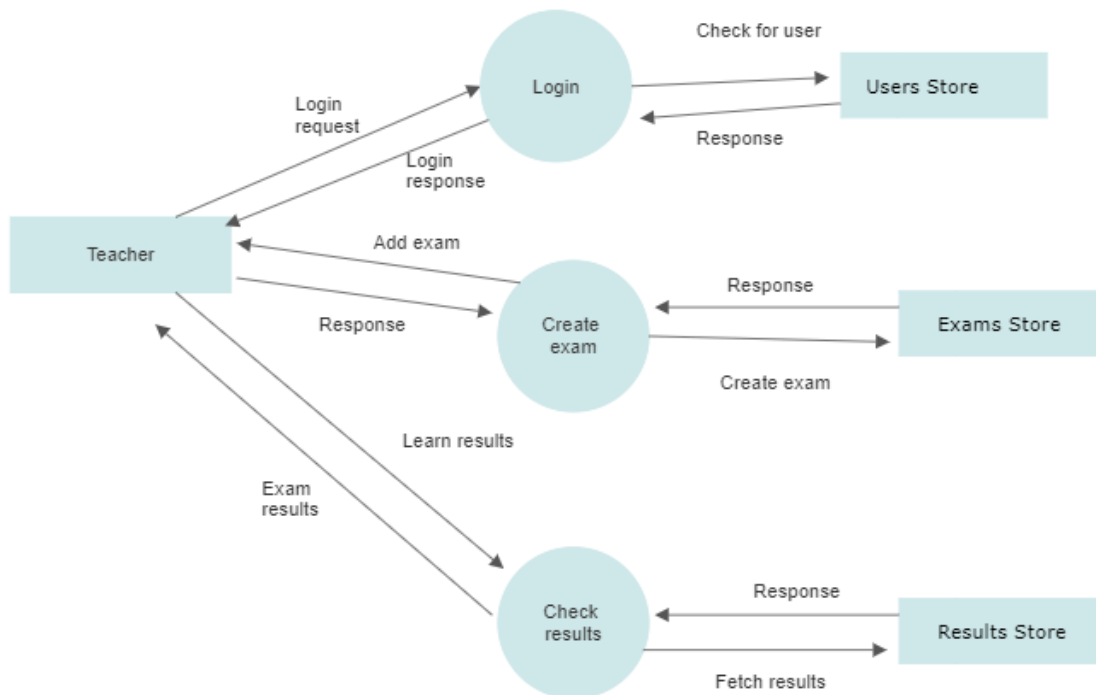
4. Design Documents

4.1 Data flow diagram

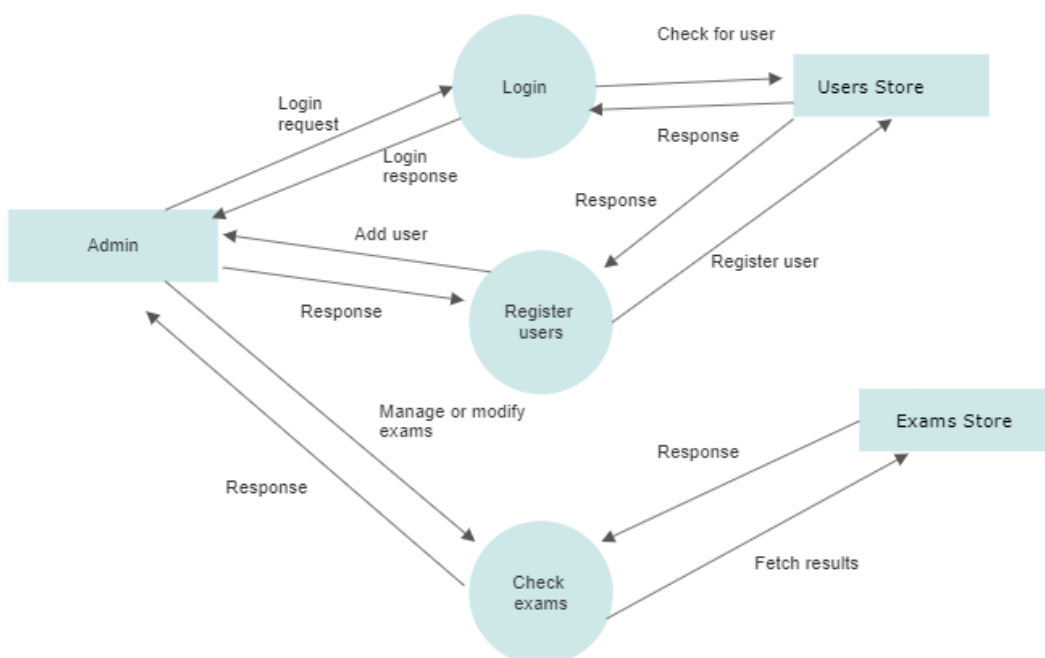
The data flow of the user whose role is a student: Demonstration of how the login, taking exam, and learning exam results are conducted



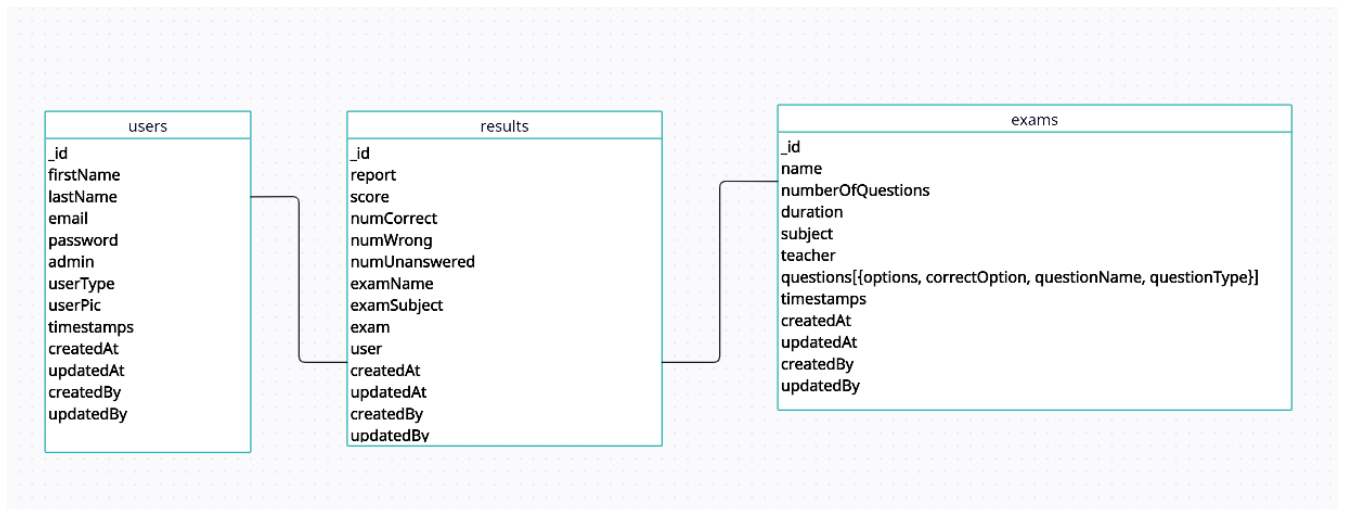
The data flow of user whose role is a teacher: Demonstrates how the teacher logged in and creating exams, followed by learning exam results of students.



The data flow of user whose role is admin: in data flow diagram show visualization of how the admin creates users and how he is checking the exams created by teachers.



4.2 Database diagram



5. Methodology

How does this application work?

Starting from here there will be explanations about how my application works on the visual side users interact with the application regarding how it behaves and what kind of functionalities it offers in detail supported by informative screenshots of the project.

Login Page:

Welcome to our platform!

Email

Password

Log in



The first page the project will start is the login page where users no matter with which role they were registered to the system will log in to the same page, it will ask users to enter their email first and the passwords and if the credentials they enter are correct they will be redirected to the home page with a success message, if the entered credentials are wrong the system will prevent user from entering to the system with then alert message shown below.

Welcome to our platform!

Email

Password

Log in

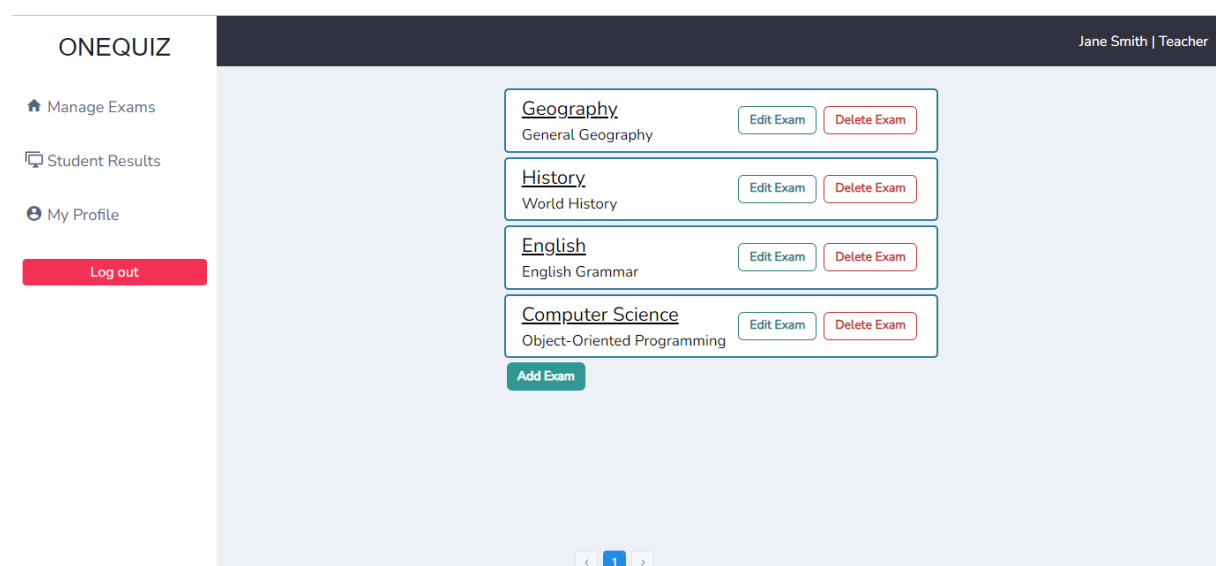


Entered password is invalid

If correctly filled the form details there will be shown the success message and signs the user in

Home Page (Teacher)

The main home page for different types of users has a similar user interface, the difference is that sidebar on the left side of the page varies by type of user that directs them to different routes that are tailored for specific user roles. To increase usability and intuitiveness for users there have been created and implemented a set of consistent user interface components like buttons, the colors of components that have been thoughtfully chosen in order to promote clarity and emphasize important elements. On the top right corner, there can be seen the logged-in user's full name followed by his or her user role in the website. The screenshot below shows the home page of how the teachers page looks with the first page where there will be a page they can work on exams.



For a teacher to create an exam he can use the button namely “Add Exam” by clicking on it he can start to create by giving the main details about the exam

The screenshot shows a list of four existing exams, each with an 'Edit Exam' button and a 'Delete Exam' button. Below the list is a green 'Add Exam' button.

Subject	Topic	Edit Exam	Delete Exam
<u>Geography</u>	General Geography	Edit Exam	Delete Exam
<u>History</u>	World History	Edit Exam	Delete Exam
<u>English</u>	English Grammar	Edit Exam	Delete Exam
<u>Computer Science</u>	Object-Oriented Programming	Edit Exam	Delete Exam

Add Exam

Upon completing with creating an exam the teacher will be notified that he successfully created an exam

The screenshot shows the 'Create Exam' form in the ONEQUIZ application. The form is titled 'Create Exam' and is located in the center of the screen. The form has a sidebar on the left with navigation links: 'Manage Exams', 'Student Results', 'My Profile', and 'Log out'. The form itself has a header 'ONEQUIZ' and a user name 'Jane Smith | Teacher'. The form contains the following fields:

- Subject: Computer Science (dropdown menu)
- Name of Exam: Programming in JavaScript (text input)
- Number of Exam Questions: 10 (text input)
- Duration of Exam: 25 (text input)

Create Exam

The screenshot shows the 'Create Exam' form with a success message. The form is titled 'Create Exam' and is located in the center of the screen. The form has a sidebar on the left with navigation links: 'Manage Exams', 'Student Results', 'My Profile', and 'Log out'. The form itself has a header 'ONEQUIZ' and a user name 'Jane Smith | Teacher'. The form contains the following fields:

- Subject: Computer Science (dropdown menu)
- Name of Exam: Programming in JavaScript (text input)
- Number of Exam Questions: 10 (text input)
- Duration of Exam: 25 (text input)

Create Exam

Exam successfully created

As soon as a new exam gets created the exam will be added and shown in the list of created exams where it can be deleted or further be edited with exam along with exam questions. On the home page of teachers where list of exams is rendered, teachers can only see the exams created by themselves and have no access to the other teachers' exams they created.

Edit Exams Page

ONEQUIZ Jane Smith | Teacher

Manage Exams Student Results My Profile Log out

Name of Exam	Number of Exam Questions	Duration of Exam	Edit Exam
Programming in JavaScript	10	25	

No Questions For Exam Added Yet Add Question

Upon clicking on the edit exam button teacher will be redirected to the page where they can work on their created exams by adding questions and even changing some exam details if it is required.

If teacher has to change exam details he can easily proceed with it by clicking on “Edit Exam” button, the modal icon will be open with previous values of the exam as placeholders so he can remember and make better decisions before editing.

When adding questions, the teacher has three options on creating a type of question he wants to add, he can add a multiple-choice, true-false, or fill-in-the-blank question type.

Add Question

Name of Question

What is the correct way to declare a variable in Java!

Question Type

Select question type

Add Question

Name of Question

What is the correct way to declare a variable in Java!

Question Type

Fill In The Blank

Correct Option

Correct option

Submit

Add Question

Name of Question

What is the correct way to declare a variable in Java!

Question Type

Multiple Choice

The Result of Created Questions

As questions get added to exam the app will instantly send them to the database and retrieve back updated exam contents to show the user the changes he made.

Name of Exam	Number of Exam Questions	Duration of Exam	Edit Exam
<u>Programming in JavaScript</u>	<u>10</u>	<u>25</u>	

Question: What is the correct way to declare a variable in JavaScript?
Correct Option: All of the above

[Delete Question](#)

[Add Question](#)

Students Results Page:

Teachers can check students' results by clicking on the student results item on sidebar and by being redirected to the page. In the page the teacher can see the name of who took the exam, the exam and subject he took, what he scored and how many correct answers he gave. In addition to these results, teachers can also find out specifically how each question the student answered in exam in the view feedback page in detail.

Jane Smith | Teacher

Students Exam Results					
Student	Exam	Subject	Score	Correct Answers	View Feedback
John Smith	General Geography	Geography	75%	3/4	View Feedback
John Smith	Programming in JavaScript	Computer Science	70%	7/10	View Feedback
Chloe Mitchell	Programming in JavaScript	Computer Science	60%	6/10	View Feedback

< 1 >

Exam Feedback Page:

Jane Smith | Teacher

Exam Feedback

Exam: Programming in JavaScript

1. What is the correct way to declare a variable in JavaScript?

☐ var x; ☐ let x;

☐ const x; ☒ All of the above

✓ Correct

Correct Answer: All of the above

2. Which operator is used for concatenating strings in JavaScript?

☒ + ☐ -

☐ * ☐ /

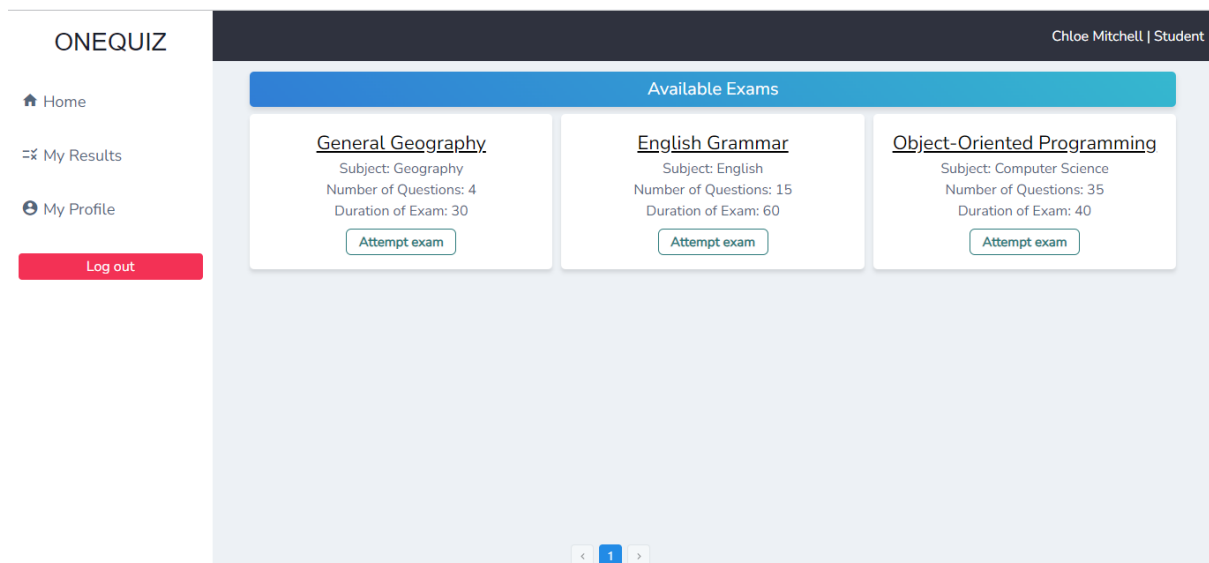
✓ Correct

Correct Answer: +

3. JavaScript is a case-sensitive programming language.

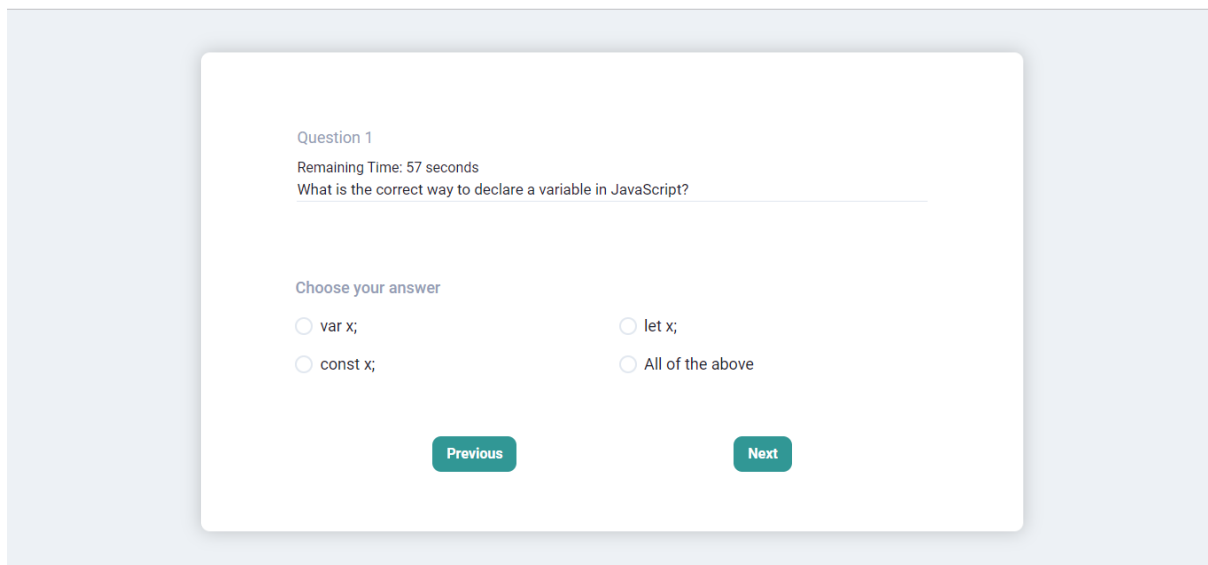
Home Page (Student)

This is a home page for users with the role created as students, as it can be seen the sidebar and pages differ from the teachers', and on the top right corner it shows the name of student and points out that the current user is a student. On the main home page students will see the available exams created by teachers, the exams will be shown as a grid layout by 3x2 and each exam shown with its name, subject of exam, number of questions, and how long it takes to complete the exam. Upon clicking "Attempt Exam" button it will prompt students again with more instructions about exam that he is about to take, so by their will they can start exam or in some cases reconsider with taking exam later.



Exam Session Pages

During the exam student will have to answer all questions in a given time and submit his answers in order to successfully finish the exam, there is also a second case that even if a student hasn't answered to all exam questions within the allotted time the exam will finish and the answers he could answer will be submitted leaving other questions if weren't answered as wrong. At the end of exam, the student will be able to see the results from the exam immediately.



Question 1

Remaining Time: 57 seconds

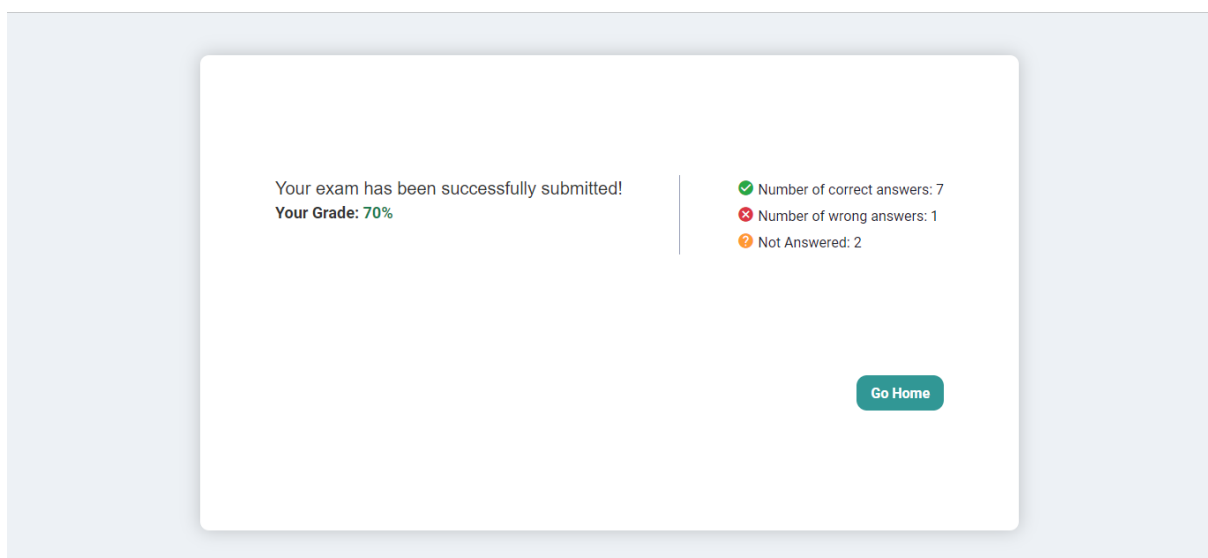
What is the correct way to declare a variable in JavaScript?

Choose your answer

☐ var x; ☐ let x;

☐ const x; ☐ All of the above

[Previous](#) [Next](#)



Your exam has been successfully submitted!

Your Grade: 70%

✔ Number of correct answers: 7
✘ Number of wrong answers: 1
🔍 Not Answered: 2

[Go Home](#)

My Results Page

In the page of my results there is a table of records from exams that a student has taken and in this page the student can only see his results unlike with teacher's student results page where he can check all the records of exams of every student. In this page student can learn what his grade is, how many questions he answered correctly and also can view feedback of exam questions where he can check which questions he answered correctly, wrong and left unanswered so he can learn his strengths and weaknesses in the courses.

ONEQUIZ

[Home](#)
[My Results](#)
[My Profile](#)
[Log out](#)

Chloe Mitchell | Student

My Results

Exam	Subject	Score	Correct Answers	View Feedback
Programming in JavaScript	Computer Science	70%	7/10	View Feedback
General Geography	Geography	67%	2/3	View Feedback

< 1 >

Chloe Mitchell | Student

2. Which operator is used for concatenating strings in JavaScript?

☒ +

☐ -

☐ *

☐ /

✓ Correct

Correct Answer: +

3. JavaScript is a case-sensitive programming language.

☐ True

☒ False

✗ Wrong

Correct Answer: true

4. The "typeof" operator in JavaScript returns the data type of a variable.

☐ True

☐ False

❗ Not Answered

Correct Answer: true

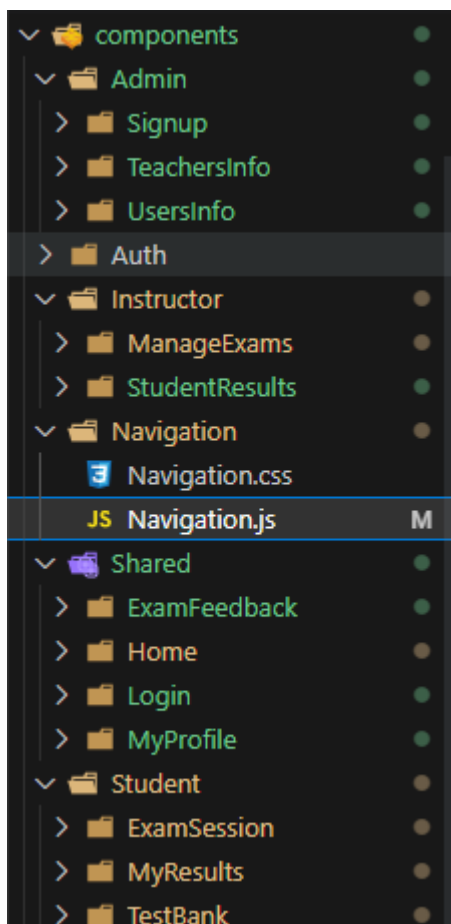
How the front-end side of the project was built?

For the front-end of my project, I used React as a primary framework for building the reusable user interface components. React is a component-based framework, components are nothing but JavaScript functions that let developers to create reusable bits of code that one can write once in some parts of application and can use anywhere where it can be useful. The components are very useful especially when you need to update your page sections, React is a very helpful tool in such cases that re-renders appropriate components whenever their state changes. There is also another tool I have in client side of my project along with React that is Redux. It is a state management tool that gives a simplicity in handling the data of components for example if the project gets bigger in size a developer will need to send data from one parent component down to many children components till the one that requires the specific data, it is called props drilling and this may bring performance issues that make users to wait so to avoid that issues I used redux. Redux is another JavaScript framework or library that usually used with React, it works as a central hub for storing data which makes it easier to manage data at one place and get it from there and sending to any component without the need to pass from one component down multiple levels of components.

Components:

As I have mentioned before, components are defined as building blocks of a user interface and one of the features that distinguishes React from other frameworks or libraries is its components especially the use of JSX (JavaScript XML). JSX allows to use of both HTML and CSS with JavaScript as if it were a markup language, as it is an extension for JavaScript it makes possible to a developer to create virtual DOM by using XML syntax. In my project, I have extensively used React components as the main structure for creating the user interface. I have put all of my components in a folder called components and put all the business logic of client side there in which some of them are just complementing blocks of components that are used together with some other ones, for example I have implemented a navigation bar component that can be leveraged by all three types of users to use as a sidebar in the main homepage component. The working of navigation component is when a home page gets its main user state variable set with user's data from database it will send data to navigation bar as a prop, as the navigation component receives the user data it will decide which list items of navigation to render on condition of user roles.

```
1 import React from "react"; 6.9k (gzipped: 2.7k)
2 import "../Navigation.css";
3
4 const Navigation = ({ currentUser, activeNavItem, onNavItemClicked }) => {
5   return (
6     <div className="sidebar-navigation">
7       <nav>
8         {currentUser && currentUser.admin === true ? (
9           <ul>
10             <li
11               className={activeNavItem === "home" ? "active" : ""}
12               onClick={() => onNavItemClicked("home")}
13             >
14               <span className="mdi mdi-account-multiple-outline"></span> Users
15             </li>
16             <li
17               className={activeNavItem === "quizzes" ? "active" : ""}
18               onClick={() => onNavItemClicked("quizzes")}
19             >
20               <span className="mdi mdi-account-circle"></span> Exams
21             </li>
22           </ul>
23         ) : (
24           <ul>
```



Handling Navigation of pages by the use of React Router

For navigation and routing with pages in my web application project I utilized one of the standard core packages which is react router. This tool enables with the navigation of project among various components as it allows with synchronizing and changing the browser URL with the corresponding rendered component. This react router tool offers to define routes in a declarative approach for the application routes. You can even create some hierarchical routes for components by creating a nested structure where some certain components come within other ones. In my project I created routes inside main react component that all my components go into this component and I defined all my routes inside this and also nested the components that are rendered within my home component as child components.

```
import Signup from "../components/Admin/Signup/Signup";
import Login from "../components/Shared/Login/Login";
import Home from "../components/Shared/Home/Home";
import MyProfile from "../components/Shared/MyProfile/MyProfile";
import ManageExams from "../components/Instructor/ManageExams/ManageExams";
import StudentResults from "../components/Instructor/StudentResults/StudentResults";
import MyResults from "../components/Student/MyResults/MyResults";
import CreateExams from "../components/Instructor/ManageExams/CreateExams";
import EditExam from "../components/Instructor/ManageExams/EditExam";
import ExamSession from "../components/Student/ExamSession/ExamSession";
import ExamFeedback from "../components/Shared/ExamFeedback/ExamFeedback";
import Users from "../components/Admin/UsersInfo/Users";
```

```
function App() {
  return (
    <React.Fragment>
      <ChakraProvider>
        <BrowserRouter>
          <Routes>
            <Route path="/" element={<Login />} />
            <Route path="/signup" element={<Signup />} />
            <Route path="exam-session/:id" element={<ExamSession />} />
            <Route path="/home/" element={<Home />} />
            <Route path="profile" element={<MyProfile />} />
            <Route path="quizzes" element={<ManageExams />} />
            <Route path="quizzes/create" element={<CreateExams />} />
            <Route path="student-results" element={<StudentResults />} />
            <Route path="exam-results" element={<MyResults />} />
            <Route path="quizzes/edit-exam/:id" element={<EditExam />} />
            <Route path="exam-feedback/:id" element={<ExamFeedback />} />
            <Route path="users-info" element={<Users />} />
          </Routes>
        </BrowserRouter>
      </ChakraProvider>
    </React.Fragment>
  );
}

export default App;
```

As my home page is highest in nested hierarchical structure it comprises navigation on the left and any other component on the right center of home page. The pages to be rendered withing home page are conditionally assigned by a function inside home component that changes by actions of users and gets mounted and as a result that specific assigned React component will be rendered in home page.

```
const renderContent = () => {
  if (currentUser && currentUser.admin === true) {
    return (
      <div>
        {activeNavItem === "home" && (
          <Users
            activeNavItem={activeNavItem}
            onNavItemClick={handleNavItemClick}
          />
        )}
        {activeNavItem === "quizzes" && (
          <ManageExams
            activeNavItem={activeNavItem}
            onNavItemClick={handleNavItemClick}
          />
        )}
      </div>
    );
  } else {
    if (currentUser && currentUser.userType === "student") {
      return (
        <div>
          {activeNavItem === "home" && (
            <TestBank
              activeNavItem={activeNavItem}
              onNavItemClick={handleNavItemClick}
            />
          )}
          {activeNavItem === "student-results" && (
            <MyResults
              activeNavItem={activeNavItem}
              onNavItemClick={handleNavItemClick}
            />
          )}
        </div>
      );
    }
  }
}
```

```
const Navigation = ({ currentUser, activeNavItem, onNavItemClick }) => {
  return (
    <div className="sidebar-navigation">
      <nav>
        {currentUser && currentUser.admin === true ? (
          <ul>
            <li
              className={activeNavItem === "home" ? "active" : ""}
              onClick={() => onNavItemClick("home")}
            >
              <span className="mdi mdi-account-multiple-outline"></span> Users
            </li>
            <li
              className={activeNavItem === "quizzes" ? "active" : ""}
              onClick={() => onNavItemClick("quizzes")}
            >
              <span className="mdi mdi-account-circle"></span> Exams
            </li>
          </ul>
        ) : (
          <ul>
            <li
              className={activeNavItem === "home" ? "active" : ""}
              onClick={() => onNavItemClick("home")}
            >
              <span className="mdi mdi-home"></span> Home
            </li>
            <li
              className={
                activeNavItem === "student-results" ? "active" : ""
              }
            >
              <span className="mdi mdi-graduation-cap"></span> Student Results
            </li>
          </ul>
        )}
      </nav>
    </div>
  );
}
```

```
const handleNavItemClick = (navItem) => {
  if (navItem !== "home") {
    if (navItem.startsWith("quizzes/edit-exam")) {
      setActiveNavItem("quizzes/edit-exam/:id");
      navigate(`/home/${navItem}`);
    } else if (navItem.startsWith("exam-feedback")) {
      setActiveNavItem("exam-feedback/:id");
      navigate(`/home/${navItem}`);
    } else if (navItem.startsWith("exam-session")) {
      setActiveNavItem("exam-session/:id");
      navigate(`/home/${navItem}`);
    } else {
      setActiveNavItem(navItem);
      navigate(`/home/${navItem}`);
    }
  } else {
    setActiveNavItem("home");
    navigate("/home");
  }
};
```

Communication of client side with the server

In my project for communication purposes between frontend and backend I have used RESTful API, which stands for Representational State Transfer, used for standardized way with its architectural style for communication purposes over the internet. To simplify handling of HTTP requests and responses

I have used Axios which is a JavaScript library. With the help of Axios I have utilized all four main HTTP methods namely POST, DELETE, PUT, and GET. The Axios library acts as the communication layer on the frontend side that makes possible with communicating with the backend, I created HTTP requests in frontend with routes that match with server-side end points as it is how they interact with each other. Good part of Axios I liked is it doesn't require to write much code but rather with less code you can just create your endpoints for both sides of app and make communication to happen, and I could use asynchronous operations including in them Axios processes for example when I create a user or fetch created user due to end of Axios process I can be sure to get from backend with response message. Therefore Axios have been used widely when I needed to send new data to database and get existing one from it. Below as an example from screenshots it is clear how registration of users is constructed and communication between client and server done.

```
const accounts = require("../routes/users");
const exams = require("../routes/exams");

app.use("/api/account", accounts);
app.use("/api/quizzes", exams);
```

```

const api = axios.create({
  baseURL: "/api",
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem("token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const signUp = async (payload) => {
  try {
    const response = await api.post("/account/register", payload);
    return response.data;
  } catch (error) {
    if (error.response) {
      return error.response.data;
    } else {
      throw error;
    }
  }
};

```

```

const generateToken = (id) => {
  return jwt.sign({ id }, process.env.ACCESS_TOKEN_SECRET, {
    expiresIn: "30d",
  });
};

router.post("/register", async (req, res) => {
  try {
    const userExists = await User.findOne({ email: req.body.email });
    if (userExists) {
      return res.status(200).send({
        message: "This email address is already registered.",
        success: false,
      });
    }

    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(req.body.password, salt);
    req.body.password = hashedPassword;

    const newUser = new User(req.body);
    await newUser.save();
    res.send({
      message: "You've successfully created an account!",
      success: true,
    });
  } catch (error) {
    res.status(500).send({
      message: error.message,
      data: error,
      success: false,
    });
  }
});

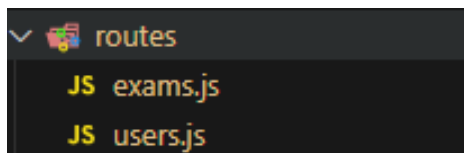
```

How the back-end side of the project was built?

The backend was created by utilizing tools such as Express.js, Node.js, and MongoDB. Starting from Express.js, it is a robust web framework built for using with Node.js, it offers a set of features like working with routing in backend, handling errors, HTTP requests and also with middleware tasks. Node.js is a main server side web framework that utilizes Express.js and accomplishes various tasks with it. I used Node.js in my project as it acts as a runtime environment to run the same client-side programming language JavaScript code on backend side, so with this tool I created async/await asynchronous operations and APIs as well as it has support for them. As JavaScript doesn't offer concurrency Node.js provides good support for asynchronous programming. In terms of database usage in my project I used MongoDB, a NoSQL database that stores the data in format similar to JSON. This database is very flexible in terms of creating schemas in contrast to relational databases requires predefined schema. After I have created MongoDB schema models I have used it to store data of users, exams and the results of exams in database.

Routers:

Routers are an integral part of my project, I have created the server side route endpoints that client can access so that it can perform specific operations on the various endpoints defined on server-side. The routes are implemented with the help of Express.js for working on API requests and responses, handling errors, and along with Express.js there have also been used mongoose for interacting with MongoDB database. I have put my routers in a folder and divided into two files that one of which is responsible with user routes and another with exams and reports, both of the route files works on HTTP requests like GET, POST, PUT, and DELETE for example when registering a new user there have been used a POST method, and for example editing exam related tasks there have been used PUT method to update and make changes of exam stored in a MongoDB database.



```
const handleDeleteExamButton = async (quizId) => {
  try {
    const token = localStorage.getItem("token");
    if (!token) {
      throw new Error("No token found");
    }

    const response = await axios.delete(`/api/quizzes/deleteQuiz/${quizId}`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    });

    const { success, message, data } = response.data;

    if (success) {
      displayNotification(message, "success");
      dispatch(setExam(data));

      setExamData((prevExamData) => {
        prevExamData.filter((exam) => exam._id !== quizId)
      });
      console.log("The redux after update", examData);

      console.log("this is a specific exam", data);
    } else {
      displayNotification(message, "error");
    }
  } catch (error) {
    displayNotification(error.message, "error");
  }
};
```



```

router.delete("/deleteQuiz/:quizId", async (req, res) => {
  try {
    const tokenHeader = req.headers.authorization;
    const token = tokenHeader.split(" ")[1];
    const decoded = jwt.verify(token, process.env.ACCESS_TOKEN_SECRET);
    req.body.userId = decoded.id;

    const { quizId } = req.params;

    let exam = await Exam.findById(quizId);

    if (!exam) {
      return res.status(404).send({
        success: false,
        message: "No such exam found in the database",
      });
    }

    exam = await Exam.deleteOne({ _id: quizId });

    res.status(200).send({
      success: true,
      message: "Exam has been deleted",
      data: exam,
    });
  } catch (error) {
    res.status(500).send({ message: error.message, success: false });
  }
});

```

User Authorization

User authorization is a crucial aspect of applications by providing each user his own unique account in the system, a user needs account so no one else besides him can have access to the account he owns. In order to provide users of my application I utilized so called JWT tokens which means JSON Web Tokens. The primary part of JWT is generating a token when user registers in successfully, but it can also be used for other purposes as well where there is a need a protection of routes. The example of I implemented and made it work is given below through the screenshots.

```

import axios from "axios"; 55.4k (gzipped: 20.3k)

const api = axios.create({
  baseURL: "/api",
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem("token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const signUp = async (payload) => {
  try {
    const response = await api.post("/account/register", payload);
    return response.data;
  } catch (error) {
    if (error.response) {
      return error.response.data;
    } else {
      throw error;
    }
  }
};

```

```

const router = require("express").Router();
const User = require("../models/usersModel");
const bcrypt = require("bcryptjs"); 21.6k (gzipped: 9.8k)
const jwt = require("jsonwebtoken"); 125.5k (gzipped: 41.1k)

const generateToken = (id) => {
  return jwt.sign({ id }, process.env.ACCESS_TOKEN_SECRET, {
    expiresIn: "30d",
  });
};

router.post("/register", async (req, res) => {
  try {
    const userExists = await User.findOne({ email: req.body.email });
    if (userExists) {
      return res.status(200).send({
        message: "This email address is already registered.",
        success: false,
      });
    }

    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(req.body.password, salt);
    req.body.password = hashedPassword;

    const newU = new User(req.body);
    await newU.save();
    res.send({
      message: "You've successfully created an account!",
      success: true,
    });
  } catch (error) {
    res.status(500).send({

```

From the above it is shown how registering a new user takes place. In the client using Axios the data is sent to server side, then in server side it checks if user the provided email exists if it doesn't it hashes a password of user and creates a new user in database. When trying to log in the server side receives payload which will be checked in server end point if password after decryption by bcrypt method matches it will generate a unique token and will send it back to user as a response.

```

router.post("/login", async (req, res) => {
  try {
    const { email, password } = req.body;

    const user = await User.findOne({ email: email });
    if (!user) {
      return res.send({ message: "Invalid email address.", success: false });
    }

    if (!(await bcrypt.compare(password, user.password))) {
      return res.send({
        message: "Entered password is invalid",
        success: false,
      });
    }

    const token = generateToken(user._id);

    res.send({
      message: "Login successful!",
      success: true,
      data: token,
    });
  } catch (error) {
    res.status(500).send({
      message: error.message,
      data: error,
      success: false,
    });
  }
});

```

For storing data of users, exams, and reports I have created the MongoDB exam models in such a way shown below.

```
const userSchema = mongoose.Schema({
  firstName: {
    type: String,
    required: true,
  },
  lastName: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  admin: {
    type: Boolean,
    default: false,
  },
  userType: {
    type: String,
    enum: ["student", "teacher"],
    default: "student",
  },
  userPic: {
    type: String,
    default: "",
  },
}, {
  timestamps: true,
});

const userModel = mongoose.model("User", userSchema);
```

```

const examSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
    },
    numberOfQuestions: {
      type: Number,
      required: true,
    },
    duration: {
      type: Number,
      required: true,
    },
    subject: {
      type: String,
      required: true,
    },
    teacher: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    questions: [
      {
        options: {
          type: [String],
          required: true,
        },
        correctOption: {
          type: String,
          required: true,
        },
        questionName: {
          type: String,
          required: true,
        },
        questionType: {
          type: String,
          enum: ["multipleChoice", "trueFalse", "fillBlank"],
          default: "multipleChoice",
        },
      },
    ],
  }
);

```

6. Conclusion

6.1 Benefits

a. Benefits to users:

1. Help teachers with conducting exams efficiently compared to traditional physical exams especially proctoring exam sessions of large auditories.
2. Help teachers to save time with grading exam paper of students by the help automatic grading of the system instead of manually checking and grading the paper.
3. Beneficial for people who are disabled and cannot physically attend exams, with help of developed project they can write from the comfort of their home.

b. Benefits to me:

1. The main benefit I got from this project is that I have gained many valuable experiences throughout the development of the system I have learned how to use and include backend to the project and the way how APIs work. I gained valuable experience by leveraging into my project modern web development tools and frameworks.
2. It is a good application that I am proud to build it and can include it in my portfolio and resume.
3. I have improved the areas in programming and coding, I have learned which areas I was lacking in terms of knowledge, and got better at problem-solving skills.

6.2 Ethics

- Privacy of users: I worked on privacy of each user no matter what his user role in the system is, privacy concerns include keeping personal user data, exam results secure from unauthorized access or breaches.
- Misinformation: The teachers should create exam questions and answers for them that are accurate and reliable to avoid spreading misinformation.
- Discrimination: In system the exam questions should be created in a way that they don't include any forms of discrimination on other people's race, religion or gender. In addition the cultural sensitivities, stereotypes, and offensive contents must be avoided.
- Social Impact: Students who will be taking exams shouldn't promote negative behaviours or cheat during the exams

Why did I choose this project?

Initially I was working only with React.js library and developing mini projects, as I was further learning frontend development I wanted to include backend side as well and as MERN (MongoDB, Express.js, React.js, Node.js) stack is very popular nowadays I wanted to try my hands in it and as a result I quite liked it. Therefore, this project was a good fit to start and learn along the way the new tools.

6.3 Future Works

I want to work on the feature with adding courses as well to this web application so users would use the same app for studying and after that taking exams. I am planning to add pretty enhanced analytics of various aspects of examination portal and broaden the features the website could bring, also I have also aspirations to make it mobile friendly which will let users interact with the app right from their mobile phones.

7. References

1. (*WCAG 2 Overview | Web Accessibility Initiative (WAI)*, n.d.) Retrieved from <https://www.w3.org/WAI/standards-guidelines/wcag/>
2. (*Express Web Framework (Node.js/JavaScript) - Learn Web Development*, 2023) Retrieved from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
3. (*Welcome to the MongoDB Documentation*, n.d.) Retrieved from <https://docs.mongodb.com/>
4. (*Docs*, n.d.) Node.js Retrieved from <https://nodejs.org/docs/>
5. Discover the best collection of eLearning articles, concepts, software and eLearning resources. Retrieved from <https://elearningindustry.com/>
6. UX Collective. Retrieved from <https://uxdesign.cc/>
7. Learning Assessment Tools & Software. Retrieved from <https://examsoft.com/>
8. MongoDB University. Retrieved from <https://learn.mongodb.com/>