TY B.Tech. (CSE) – II [2022-23]

5CS372: Advanced Database System Lab.

Assignment No. 3

PRN: 2020BTECS00075

Name: Tushar Rajendra Patil

Batch: T7

AIM: To Design and implement a web-enabled (portal) student MIS (Management Information System) for University schema attached in separate file

Theory: For creating the web portal I have used ReactJS for the frontend and ExpressJS for the backend

ReactJS is a JavaScript library for building user interfaces. It was developed and is maintained by Facebook, and is widely used for building single-page applications and mobile applications.

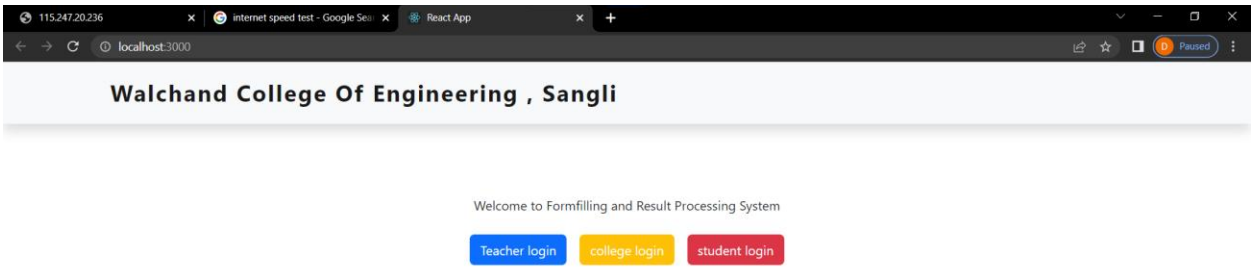
React allows for building reusable UI components, and uses a virtual DOM(Document Object Model) which optimizes updates and rendering of components. It uses syntax extension of JavaScript called JSX, which allows you to write HTML-like code within your JavaScript.

ExpressJS is a popular open source Node.js framework for building web applications. It is designed to simplify the process od building and deploying web applications and APIs, and provides a robust set of features for creating HTTP servers.

One of the key features of Express is its ability to handle HTTP requests and route them to appropriate handlers, allowing you to build complex routing systems for your application. Express also provides support for middleware, which are functions that can modify incoming requests or handle responses before they reach final destination.

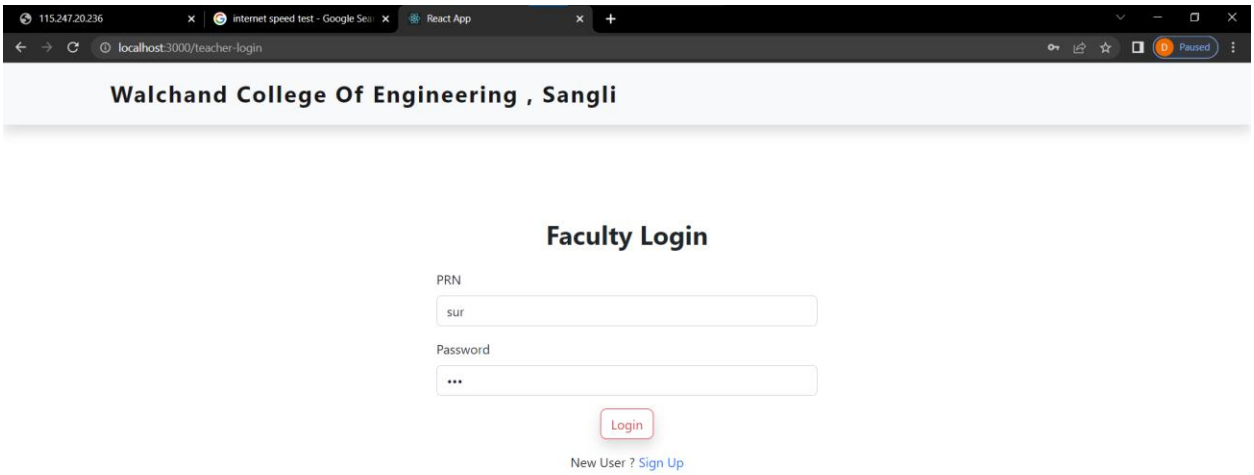
For the implementation of web-enabled Student MIS we need to design a frontend for taking user input from form data and then send it to the backend which we need to build which will connect to the oracle database

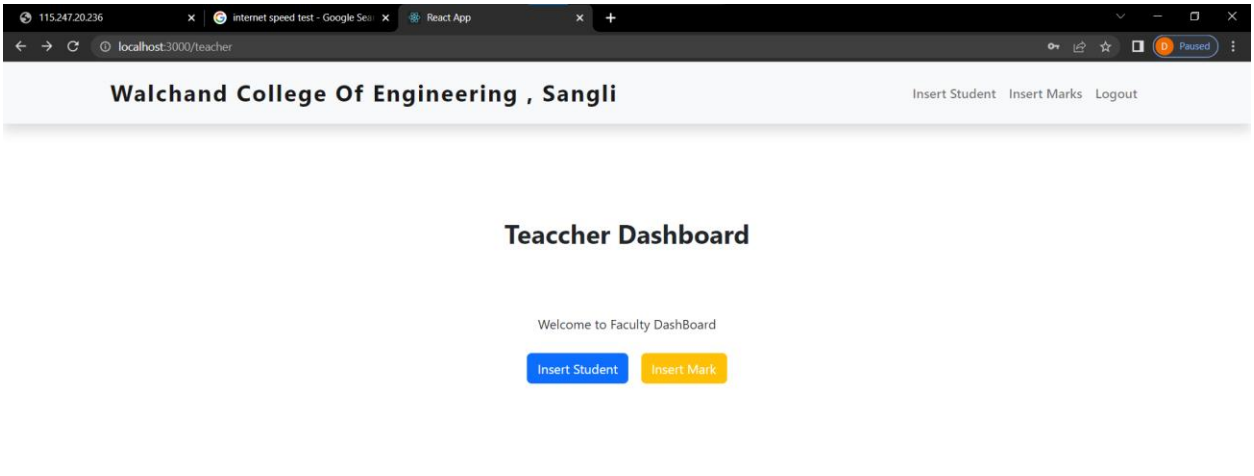
1. Create a frontend:



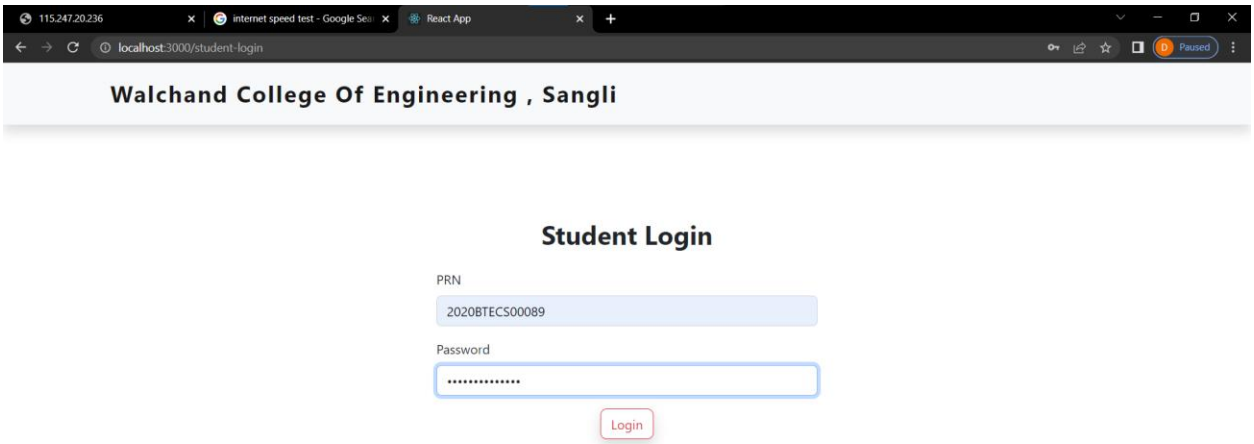
This is a simple page built by using react and it contains 3 buttons to authorize according to the privilege of the user and having the required credentials.

Teacher login-





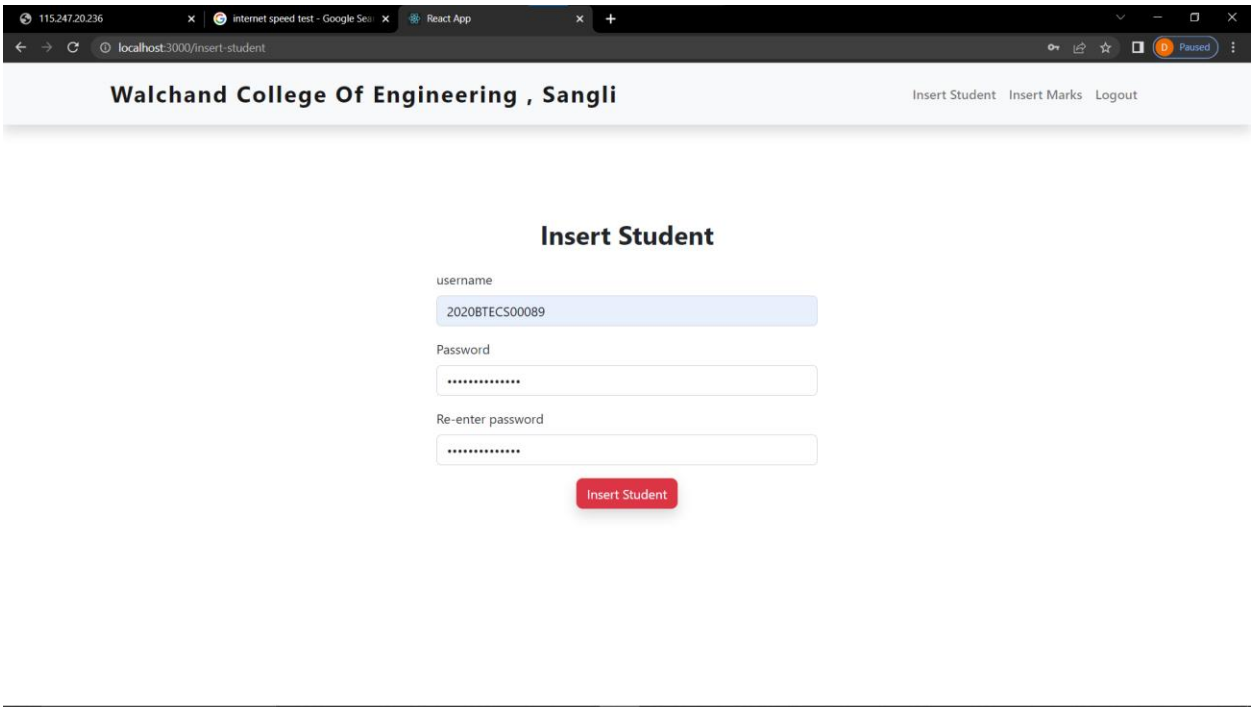
Student login-

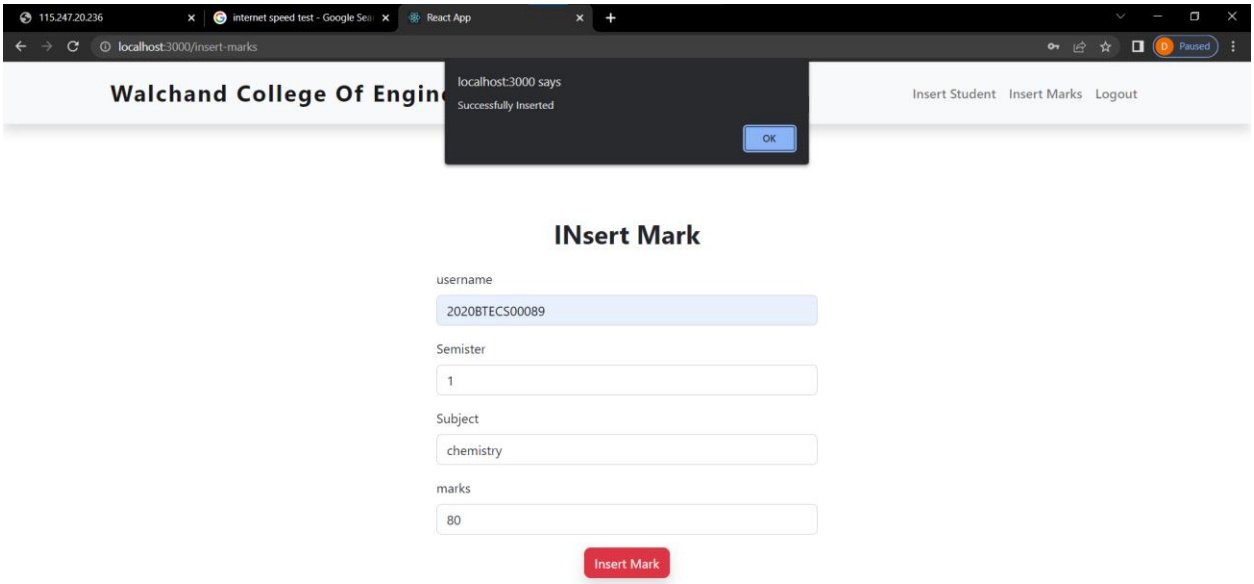


3

2. Create a form for taking input

Inserting the student into the database





This form will take the input to be sent to the backend and it will then toggle the database accordingly.

3. Sending data from frontend to backend

```
function handleClick() {
    axios.post("/login", { username: user.username, password: user.password, role: "teacher" })
        .then(res => {
            if (res.data.message === "teacher login Successfull") {
                alert("login Successfull");
                navigate("/teacher");
            }
            else {
                alert(res.data.message);
            }
        })
        .catch((error) => {
            console.log(error);
        });
}
```

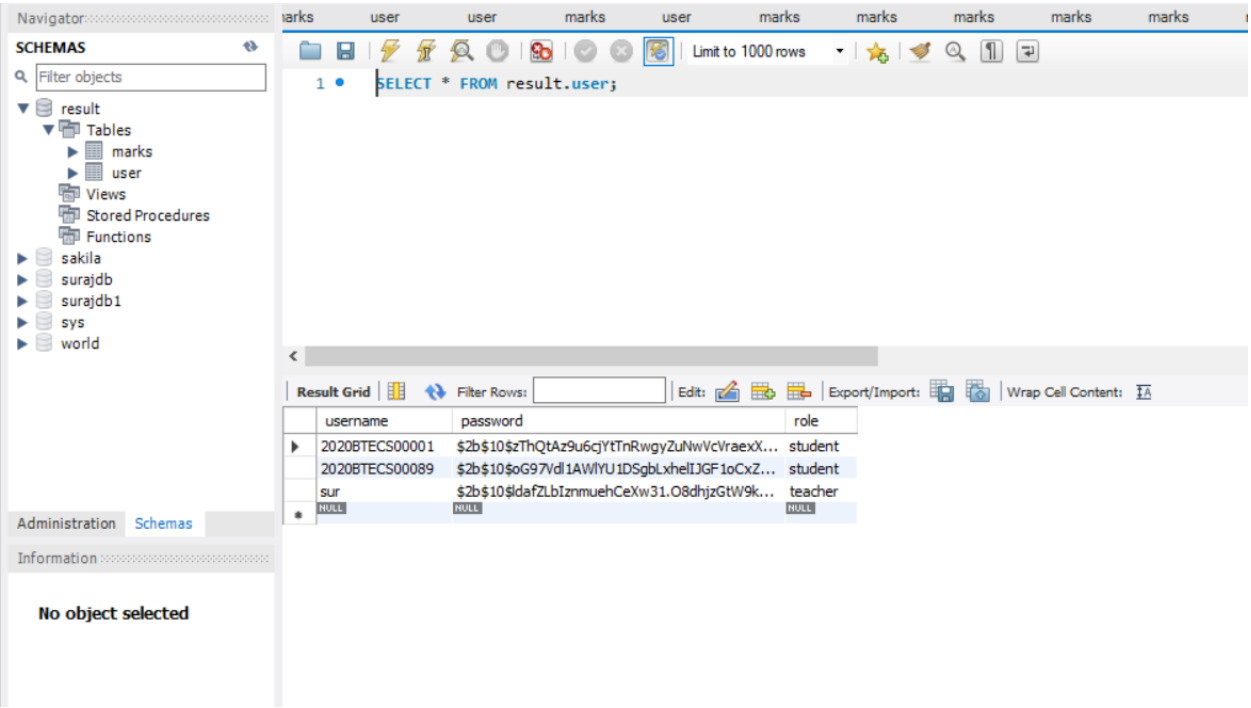
Sending data from backend to frontend

```
app.post('/login', function(req, res, next) {
    passport.authenticate('local', function(err, user, info) {
      if (err) {
        res.send({message: "error in teacherlogin router"});
      }
      else if (!user) {
        res.send({ message: "User not found" });
      }
      else {
        req.login(user, function(err) {
          if (err) {
            console.log(err);
            res.send({message: "error while login"});
          }
          else {
            res.send({ message: `${req.body.role} login Successfull` });
          }
        });
      }
    })(req, res, next);
});
```

This is the code in ExpressJS for the submission of Form Data onto the oracle database using the oracledb package available in node.

We are storing cookies using express-session and passport-local

4. Create Roles in your target database to grant to the registered users



Make sure to provide different paths for different kinds of authorization.

```
JS Slogin.js U   JS Student.js U   JS Navbar.js U   JS App.js M ×   JS Student.js (File Saved • February 14, 2
client > src > JS App.js > ...
import './App.css';
import {BrowserRouter as Router, Routes, Route} from "react-router-dom";
import Navbar from './components/navbar/Navbar';
import Section from './components/section/Section';
import Slogin from './components/slogin/Slogin';
import Sregister from './components/sregister/Sregister';
import Student from './components/student/Student';
import Teacher from './components/teacher/Teacher';
import Tlogin from './components/tlogin/Tlogin';
import Tregister from './components/tregister/Tregister';
import InsertStudent from './components/createStudent/InsertStudent';
import InsertMarks from './components/insertMarks/InsertMarks';

import Header from './components/Header/Header';
import Home from './components/Home/Home';

function App() {
  return (
    <Router>
      <Navbar />
      {/* <ToastContainer/> */}
      <Routes>
        <Route path="/section" element={<Section />} />
        <Route path="/student-login" element={<Slogin />} />
        <Route path="/student-register" element={<Sregister />} />
        <Route path="/student" element={<Student />} />
        <Route path="/teacher" element={<Teacher />} />
        <Route path="/teacher-login" element={<Tlogin />} />
        <Route path="/teacher-register" element={<Tregister />} />
        <Route path="/insert-student" element={<InsertStudent />} />
        <Route path="/insert-marks" element={<InsertMarks />} />



        <Route path="/" element={<Home />} />
      </Routes>
    </Router>
  );
}

export default App;
```

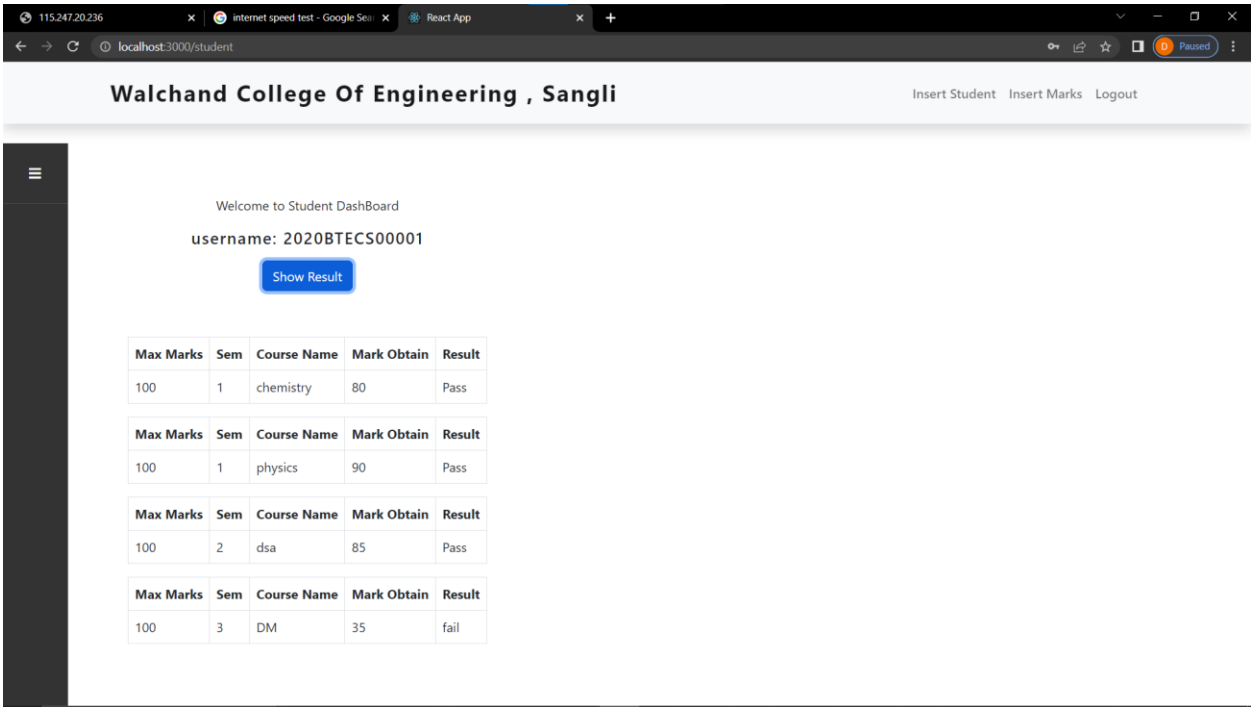
Here the paths for the three levels of users is given accordingly in the frontend.

5. When the details are to be displayed run the query on the database throughout the backend server to display data according to the privileges assigned.

Result Grid | Filter Rows: | Export:

	username	sem	subject	marks
▶	2020BTECS00001	1	chemistry	80
	TEACHER0001	2	Arduino	90
	2020BTECS00001	1	physics	90
	2020BTECS00001	2	dsa	85
	2020BTECS00001	3	DM	35
	2020BTECS00089	1	chemistry	80

Displaying Result:



➔ **CONCLUSION:** Thus we have created a web-enabled student MIS with different levels of management which is done by using restriction on privileges granted to the users signed up.
The frontend sends the data to the backend and the backend connects to the database to carry out the operations and take the role of granting privileges and making new users with proper authentication.

Relevant Code:

1. Granting Privileges:

```
const express = require('express');
const cors = require('cors');
const path = require('path');
const bcrypt = require('bcrypt');
const session = require(''express-session'');
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const mysql = require('mysql2');

const app = express();
app.use(express.urlencoded({extended: true}));
app.use(express.json());

app.use(cors());

app.use(session({
  secret: 'your-secret-key',
  resave: false,
  saveUninitialized: true
}));

// Create a MySQL connection pool
const pool = mysql.createPool({
  host: 'localhost',
  user: 'root',
  password: 'suraj',
  database: 'result'
});

const saltRounds = 10;

passport.serializeUser((user, done) => {
```

```
  done(null, user.username);
});

passport.deserializeUser(function(username, done) {
  pool.query('SELECT * FROM user WHERE username = ?', [username], function(err, results) {
    done(err, results[0]);
  });
});

function hashPassword(password) {
  return bcrypt.hashSync(password, saltRounds);
}

function checkPassword(password, hashedPassword) {
  return bcrypt.compareSync(password, hashedPassword);
}

// Use the LocalStrategy with Passport
passport.use(new LocalStrategy(
  function(username, password, done) {
    // Check if the username and password match a record in the database
    pool.query('SELECT * FROM user WHERE username = ?', [username], function(err, results) {
      if (err) { return done(err); }
      if (!results.length) {
        return done(null, false, { message: 'Incorrect username.' });
      }
      if (!checkPassword(password, results[0].password)) {
        return done(null, false, { message: 'Incorrect password.' });
      }
      return done(null, results[0]);
    });
  }
));

// Define the checkPassword function for checking the hashed password

// Use Passport's sessions for tracking user's login status
app.use(passport.initialize());
app.use(passport.session());

// Define the route for the login page
// app.get('/login', function(req, res) {
//   res.render('login');
// });
```

```
// Define the route for handling the login process

app.post("/", function(req, res) {
  res.send("hello");
})

app.post('/login', function(req, res, next) {
    passport.authenticate('local', function(err, user, info) {
      if (err) {
        res.send({message: "error in teacherlogin router"});
      }
      else if (!user) {
        res.send({ message: "User not found" });
      }
      else {
        req.login(user, function(err) {
          if (err) {
            console.log(err);
            res.send({message: "error while login"});
          }
          else {
            res.send({ message: `${req.body.role} login Successfull` });
          }
        });
      }
    })(req, res, next);
});


app.post('/register', function(req, res) {
    const username = req.body.username;
    const password = req.body.password;
    const role = req.body.role;

    // Hash the password before saving it to the database
    const hashedPassword = hashPassword(password);

    // Insert the new user into the database
    if(role === "teacher") {
      pool.query('INSERT INTO user (username, password, role) VALUES (?, ?, ?)', [username,
hashedPassword, role], function(err, results) {
        if (err) {
          // Handle the error and return an error message
          console.log("error in /register");
          res.send({ message: 'An error occurred while registering the user.' });
```

```
      }
      else {
        // Return a success message
        console.log("successs");
        res.send({ message: 'teacher registered successfully.' });
      }
    });
  }
  else {
    pool.query('INSERT INTO user (username, password, role) VALUES (?, ?, ?)', [username,
hashedPassword, role], function(err, results) {
      if (err) {
        // Handle the error and return an error message
        console.log("error in /register");
        res.send({ message: 'An error occurred while registering the user.' });
      }
      else {
        // Return a success message
        console.log("successs");
        res.send({ message: 'student registered successfully.' });
      }
    });
  }
 });

// app.post("/teacherlogin", (req, res) => {
//     const username = req.body.username;
//     const password = req.body.password;
// });

app.post("/insert-marks", (req, res) => {
 const {username, sem, subject, marks} = req.body;

 pool.query('INSERT INTO marks (username, sem, subject, marks) VALUES (?, ?, ?, ?)', [username, sem,
subject, marks], function(err, results) {
  if (err) {
    // Handle the error and return an error message
    console.log(err);
    res.send({ message: 'An error occurred while inserting marks.' });
  }
  else {
    res.send({ message: 'Mark Inserted' });
  }
 })
})
```

```
app.post("/getresult", (req, res) => {
  if (req.isAuthenticated()) {
    const username = req.user.username;
    pool.query('SELECT * from marks WHERE username = ?', [username], function(err, results) {
      if (err) {
        // Handle the error and return an error message
        console.log("error in /get results");
        res.send({ message: 'An error occurred while get result.' });
      }
      else {
        // res.send({ message: 'Mark Inserted' });
        res.send({message: "success", result: results});
        console.log(results);
      }
    })
  } else {
    console.log("not authenticated");
  }
});


app.post("/checkforauthentication", (req, res) => {
  if (req.isAuthenticated()) {
    res.send({message: "authenticated"});
  }
  else {
    res.send({ message: 'not authenticated' });
  }
});


// Define the route for the dashboard
app.get('/dashboard', function(req, res) {
  if (!req.user) {
    return res.redirect('/login');
  }
  res.render('dashboard', { username: req.user.username });
});

// Define the route for the logout process
app.post('/logout', function(req, res) {
  console.log("hello");
  req.logout(function(err) {
    if(err) {
```

```
      console.log(err);
    }
    else {
      res.send({message: "logout successfull"});
    }
  });
});

app.post("/insert-")

// Start the Express server
app.listen(9002, function() {
  console.log('Server running at http://localhost:9002');
});
```

```
Server running at http://localhost:9002
successs
```