

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.

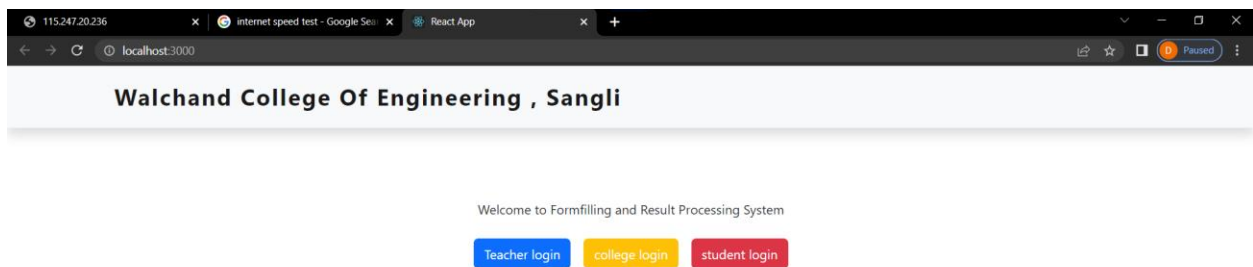
React allows for building reusable UI components, and uses a virtual DOM(Document Object Model) which optimizes updates and rendering of components. It uses syntax extension of JavaScript called JSX, which allows you to write HTML-like code within your JavaScript.

ExpressJS is a popular open source Node.js framework for building web applications. It is designed to simplify the process of 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-

115.247.20.236 x internet speed test - Google Se... x React App x +
localhost:3000/teacher-login

Walchand College Of Engineering , Sangli

Faculty Login

PRN

Password

Login

New User ? [Sign Up](#)

115.247.20.236 x internet speed test - Google Se... x React App x +
localhost:3000/teacher

Walchand College Of Engineering , Sangli

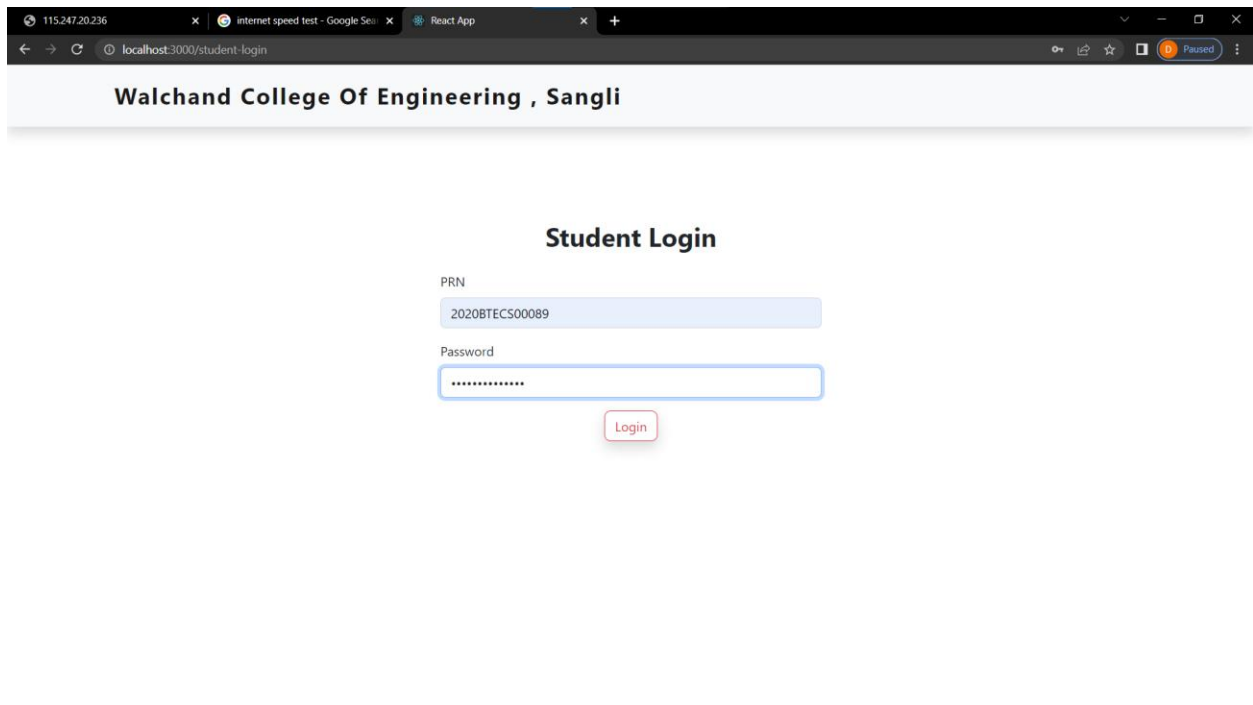
Insert Student Insert Marks Logout

Teaccher Dashboard

Welcome to Faculty DashBoard

Insert Student Insert Mark

Student login-



3

2. Create a form for taking input

Inserting the student into the database

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/insert-student'. The page header includes the text 'Walchand College Of Engineering , Sangli' and navigation links for 'Insert Student', 'Insert Marks', and 'Logout'. The main content area is titled 'Insert Student' and contains three input fields: 'username' with the value '2020BTECS00089', 'Password' with masked characters, and 'Re-enter password' with masked characters. A red 'Insert Student' button is positioned below the input fields.

The screenshot shows the 'Insert Mark' form in a web browser window. The address bar displays 'localhost:3000/insert-marks'. The page header is the same as the previous screenshot. A dark overlay message box in the center reads 'localhost:3000 says Successfully Inserted' with an 'OK' button. Below the message, the 'Insert Mark' form is visible, containing four input fields: 'username' (2020BTECS00089), 'Semister' (1), 'Subject' (chemistry), and 'marks' (80). A red 'Insert Mark' button is at the bottom.

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

```

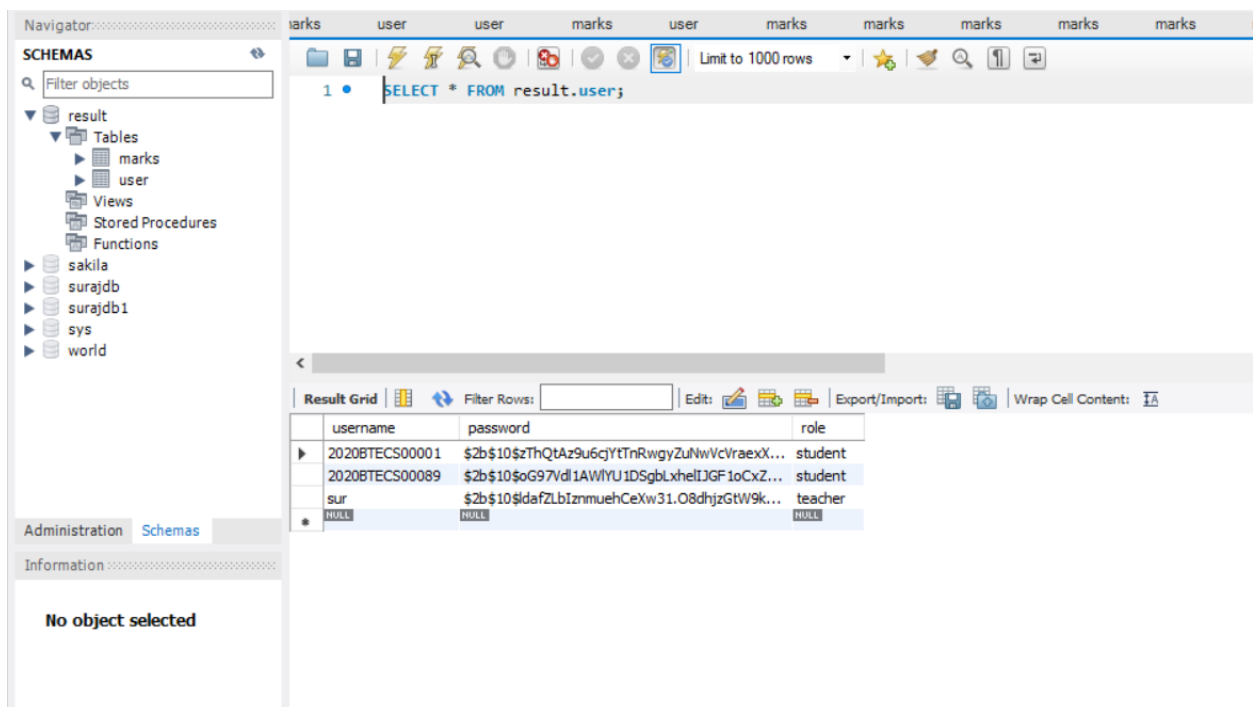
366 app.post('/login', function(req, res, next) {
367   passport.authenticate('local', function(err, user, info) {
368     if (err) {
369       res.send({message: "error in teacherlogin router"});
370     }
371     else if (!user) {
372       res.send({ message: "User not found" });
373     }
374     else {
375       req.login(user, function(err) {
376         if (err) {
377           console.log(err);
378           res.send({message: "error while login"});
379         }
380         else {
381           res.send({ message: `${req.body.role} login Successfull` });
382         }
383       });
384     }
385   })(req, res, next);
386 });
387

```

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



The screenshot shows a database management interface. On the left is a 'Navigator' pane with a 'SCHEMAS' section containing a tree view of databases: result, sakila, surajdb, surajdb1, sys, and world. The 'result' database is expanded, showing 'Tables' (marks, user) and 'Views'. The main window displays a query: `SELECT * FROM result.user;` and a 'Result Grid' with the following data:

| username | password | role |
|----------------|---|---------|
| 2020BTECS00001 | \$2b\$10\$zThQtAz9u6cjYtTnRwgyZuNwVcVraexX... | student |
| 2020BTECS00089 | \$2b\$10\$0G97Vdl1AW/YU1DSgblXhellJGF1oCxZ... | student |
| sur | \$2b\$10\$dafZLbIznmuehCeXw31.O8dhjzGtW9k... | teacher |
| NULL | NULL | NULL |

Below the grid, there are tabs for 'Administration' and 'Schemas', and a message 'No object selected'.

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 X    JS Student.js (File Saved • February 14, 2020)

client > src > JS App.js > ...
1  import './App.css';
2  import {BrowserRouter as Router, Routes, Route} from "react-router-dom";
3  import Navbar from './components/navbar/Navbar';
4  import Section from './components/section/Section';
5  import Slogin from './components/slogin/Slogin';
6  import Sregister from './components/sregister/Sregister';
7  import Student from './components/student/Student';
8  import Teacher from './components/teacher/Teacher';
9  import Tlogin from './components/tlogin/Tlogin';
10 import Tregister from './components/tregister/Tregister';
11 import InsertStudent from './components/createStudent/InsertStudent';
12 import InsertMarks from './components/insertMarks/InsertMarks';
13
14 import Header from './components/Header/Header';
15 import Home from './components/Home/Home';
16
17 function App() {
18   return (
19     <Router>
20       <Navbar />
21       { /* <ToastContainer /> */ }
22       <Routes>
23         <Route path="/section" element={<Section />} />
24         <Route path="/student-login" element={<Slogin />} />
25         <Route path="/student-register" element={<Sregister />} />
26         <Route path="/student" element={<Student />} />
27         <Route path="/teacher" element={<Teacher />} />
28         <Route path="/teacher-login" element={<Tlogin />} />
29         <Route path="/teacher-register" element={<Tregister />} />
30         <Route path="/insert-student" element={<InsertStudent />} />
31         <Route path="/insert-marks" element={<InsertMarks />} />
32
33
34
35         <Route path="/" element={<Home />} />
36       </Routes>
37     </Router>
38   );
39 }
40
41 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:

115.247.20.236 x Internet speed test - Google Se... x React App x +

localhost:3000/student

Walchand College Of Engineering , Sangli

Insert Student Insert Marks Logout

☰

Welcome to Student DashBoard

username: 2020BTECS00001

Show Result

| Max Marks | Sem | Course Name | Mark Obtain | Result |
|-----------|-----|-------------|-------------|--------|
| 100 | 1 | chemistry | 80 | Pass |

| Max Marks | Sem | Course Name | Mark Obtain | Result |
|-----------|-----|-------------|-------------|--------|
| 100 | 1 | physics | 90 | Pass |

| Max Marks | Sem | Course Name | Mark Obtain | Result |
|-----------|-----|-------------|-------------|--------|
| 100 | 2 | dsa | 85 | Pass |

| Max Marks | Sem | Course Name | Mark Obtain | Result |
|-----------|-----|-------------|-------------|--------|
| 100 | 3 | DM | 35 | fail |

➔ **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
```