RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Assignment

Course Code	CSC311A
Course Name	Database Systems
Programme	B.Tech
Department	CSE
Faculty	FET

Name of the Student	Satyajit Ghana
Reg. No.	17ETCS002159
Semester/Year	06/2020
Course Leader(s)	Ms. Ami Rai E.

Declaration Sheet

Student Name	Satyajit Ghana		
Reg. No	17ETCS002159		
Programme	B.Tech	Semester/Year	06/2020
Course Code	CSC311A		
Course Title	Database Systems		
Course Date		to	
Course Leader	Ms. Ami Rai E.		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

List Of Figures

1 Question A

Solution to Question A

1.1 Functional and Data Requirements

1.1.1 Functional Requirements

1. The system should allow the staff to login and students to login/register
2. The system should allow the students to enter their group project details in a form
3. The system should allow the students to register their group project
4. The system should display the room and table no. allotted to the group
5. The system should allow the students to cancel their registration
6. The system should allow the staff to view all the registered group projects
7. The system should allow the staff to view the allotted room and table no of a group project
8. The system should allow the staff to cancel a group registration

Requirement Tag	**FR1**
Requirement Description	The system should allow the staff to login and students to login/register
Dependent on	None
User/System interacting with the requirement	Staff, Student

Requirement Tag	**FR2**
Requirement Description	The system should allow the students to enter their group project details in a form
Dependent on	FR1
User/System interacting with the requirement	Student

Requirement Tag	FR3
Requirement Description	The system should allow the students to register their group project
Dependent on	FR1, FR2
User/System interacting with the requirement	Student

Requirement Tag	FR4
Requirement Description	The system should display the room and table no. allotted to the group
Dependent on	FR1, FR2, FR3
User/System interacting with the requirement	Student

Requirement Tag	FR5
Requirement Description	The system should allow the students to cancel their registration
Dependent on	FR1, FR3
User/System interacting with the requirement	Student

Requirement Tag	FR6
Requirement Description	The system should allow the staff to view all the registered group projects
Dependent on	FR1, FR3
User/System interacting with the requirement	Staff

Requirement Tag	FR7
Requirement Description	The system should allow the staff to view the allotted room and table no of a group project
Dependent on	FR1, FR3
User/System interacting with the requirement	Staff

Requirement Tag	FR8
Requirement Description	The system should allow the staff to cancel a group registration
Dependent on	FR1, FR3
User/System interacting with the requirement	Staff

1.1.2 Data Requirements

Requirement Tag	DR1
Item Name	Staff Details
Item Description	Maintains the staff details `id, username, hashed_password`
Item Type	Integer, String, String
User/System interacting with the requirement	Staff

Requirement Tag	DR2
Item Name	Student Login
Item Description	Maintains the student login details `id, username, hashed_password`
Item Type	Integer, String, String
User/System interacting with the requirement	Student

Requirement Tag	DR3
Item Name	Student Details
Item Description	Maintains the student details `id, reg_no, name, department, course, contact_no`
Item Type	Integer, String, String, String, String, String
User/System interacting with the requirement	Student

Requirement Tag	DR4
Item Name	Project Details
Item Description	Maintains the project details `id, project_leader_reg_no, project_name, mentor_name, department, category`
Item Type	Integer, String, String, String, String, String
User/System interacting with the requirement	Student and Staff

Requirement Tag	DR5
Item Name	Exhibition Details
Item Description	Maintains the Rooms available for exhibition `room_id, room_name, capacity`
Item Type	Integer, String, Integer
User/System interacting with the requirement	Student and Staff

Requirement Tag	DR6
Item Name	Project Student Register
Item Description	Maintains the Students registered for a Project `room_id, student_reg_no`
Item Type	Integer, String
User/System interacting with the requirement	Student

Requirement Tag	DR7
Item Name	Project Exhibition
Item Description	Maintains the Table no. allotted to a Project `room_id, project_id, table_no`
Item Type	Integer, Integer, Integer
User/System interacting with the requirement	Student

1.2 Design

1.2.1 ER Diagram

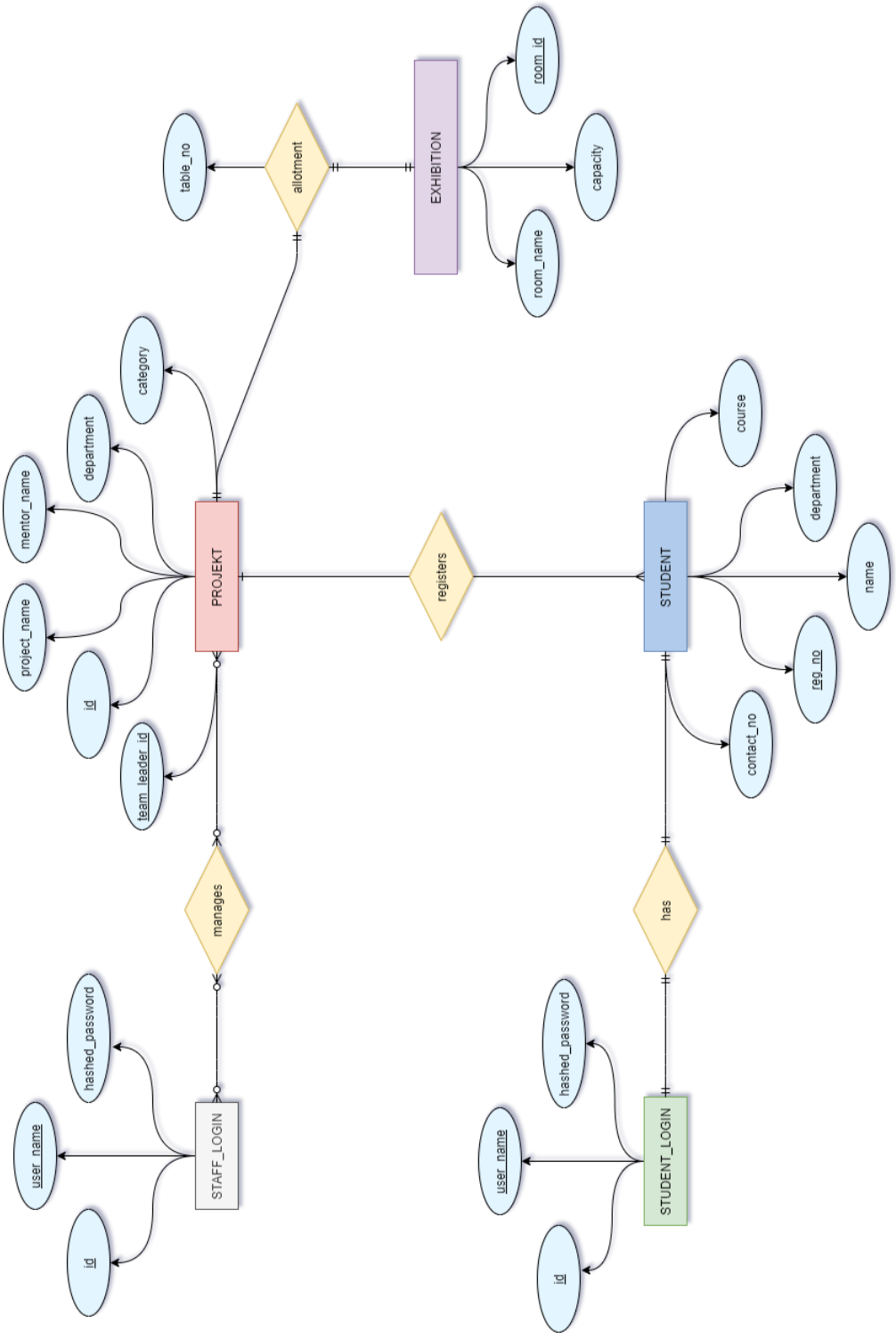

Figure 1-1 Project Exhibition ER Diagram

1.2.2 Relational Schema

STAFF_LOGIN

id	user_name	hashed_password

STUDENT_LOGIN

id	user_name	hashed_password

STUDENT

fk_id	reg_no	name	department	course	contact_no

PROJEKT

fk_team_leader_id	id	project_name	mentor_name	department	category

PROJECT_STUDENT_REGISTER

fk_project_id	fk_student_id

EXHIBITION

room_id	room_name	capacity

PROJECT_EXHIBITION

fk_room_id	fk_project_id	table_no

1.2.3 Table Attributes

Implementation in MySQL

```
DROP TABLE IF EXISTS PROJECT_EXHIBITION;
DROP TABLE IF EXISTS EXHIBITION;
DROP TABLE IF EXISTS PROJECT_STUDENT_REGISTER;
DROP TABLE IF EXISTS PROJEKT;
DROP TABLE IF EXISTS STUDENT;
DROP TABLE IF EXISTS STUDENT_LOGIN;
DROP TABLE IF EXISTS STAFF_LOGIN;

CREATE TABLE STAFF_LOGIN
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(20) UNIQUE KEY NOT NULL,
    hashed_password CHAR(60) NOT NULL
);

CREATE TABLE STUDENT_LOGIN
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(20) UNIQUE KEY NOT NULL,
    hashed_password CHAR(60) NOT NULL
);

CREATE TABLE STUDENT
(
    id INT(5) UNIQUE KEY NOT NULL,
    reg_no CHAR(12) PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    department ENUM('CSE', 'EEE', 'ECE', 'CIVIL'),
    course ENUM('B.Tech', 'M.Tech') NOT NULL,
    contact_no VARCHAR(10) NOT NULL,
    FOREIGN KEY(id) REFERENCES STUDENT_LOGIN(id)
);

CREATE TABLE PROJEKT
(
    id INT(5) PRIMARY KEY AUTO_INCREMENT,
    project_leader_regno CHAR(12) UNIQUE KEY NOT NULL,
    project_name VARCHAR(100) UNIQUE KEY NOT NULL,
    mentor_name VARCHAR(30) NOT NULL,
    department ENUM('CSE', 'EEE', 'ECE', 'CIVIL') NOT NULL,
    category VARCHAR(30) NOT NULL
);

CREATE TABLE PROJECT_STUDENT_REGISTER
(
    project_id INT(5) NOT NULL,
    student_reg_no CHAR(12) NOT NULL,
    FOREIGN KEY(project_id) REFERENCES PROJEKT(id),
    FOREIGN KEY(student_reg_no) REFERENCES STUDENT(reg_no)
```

```
);

CREATE TABLE EXHIBITION
(
    room_id INT(5) PRIMARY KEY AUTO_INCREMENT,
    room_name CHAR(20) UNIQUE KEY NOT NULL,
    capacity INT(5) NOT NULL
);

CREATE TABLE PROJECT_EXHIBITION
(
    room_id INT(5) NOT NULL,
    project_id INT(5) UNIQUE KEY NOT NULL,
    table_no INT(5) CHECK ( table_no > 0 AND table_no < ( SELECT * FROM EXHIBITION WHERE EXHIBITION
.room_id = room_id LIMIT 1 ) ),
    FOREIGN KEY(room_id) REFERENCES EXHIBITION(room_id),
    FOREIGN KEY(project_id) REFERENCES PROJEKT(id)
);
```

Inserting Data into the Tables

```
INSERT INTO `STUDENT_LOGIN` (`id`, `user_name`, `hashed_password`) VALUES
(1, '17ETCS002159', '$2b$10$uVRx4ogFBi0owMljpvEilONnd9wWOMtrpgVwqw2Mw8.aNmo6yEU1u'),
(2, '17ETCS002122', '$2b$10$ATp9qxsPWBsOUXDAB1YvK.yTLi4GK1mzpIHBCfSOCQwtxLU/52Pk2'),
(3, '17ETCS002168', '$2b$10$3cfBMD3yRi3YJk.fFGrNY.Yx1RRonj4z2cqgOe2fgZ78yNaqRxFkC');

INSERT INTO `STUDENT` (`id`, `reg_no`, `name`, `department`, `course`, `contact_no`) VALUES
(2, '17ETCS002122', 'Prachi Poddar', 'CSE', 'B.Tech', '9856523658'),
(1, '17ETCS002159', 'Satyajit Ghana', 'CSE', 'B.Tech', '7892137665'),
(3, '17ETCS002168', 'Shikhar Singh', 'CSE', 'B.Tech', '9852145896');

INSERT INTO `PROJEKT` (`id`, `project_leader_regno`, `project_name`, `mentor_name`, `department`, `
category`) VALUES
(2, '17ETCS002159', 'KrishiAI', 'Chaitra S', 'CSE', 'DL');


INSERT INTO `PROJECT_STUDENT_REGISTER` (`project_id`, `student_reg_no`) VALUES
(2, '17ETCS002159'),
(2, '17ETCS002122'),
(2, '17ETCS002168');


INSERT INTO `EXHIBITION` (`room_id`, `room_name`, `capacity`) VALUES
(1, 'A201', 60);
```

1.3 Implementation of Back-End

The backend was written in NodeJS using ExpressJS.

Refer to Appendix B for Complete Source Code, the essential components are illustrated in this section.

The Express Router uses POST model to serve as a server, the routes are defined as

```
app.get('/', (req, res) => {
    res.json({ message: 'Welcome to RUAS Server' })
});

app.use('/staff', stafflogin);
app.use('/student', student);
app.use('/projekt', projekt);
app.use('/exhibition', project_exhibitions);

app.use((err, req, res, next) => {
    res.status(err.status || 501);
    res.json({
        message: err.message
    });
});
```

mysql client for nodejs is used to establish a connection to the mysql server running at port 3306

```
const connection = mysql.createConnection({
    host: dbConfig.HOST,
    user: dbConfig.USER,
    password: dbConfig.PASSWORD,
    database: dbConfig.DB
});

Create a connection to the mysql server
connection.connect(error => {
    if (error) throw error;
    console.log('mysql successfully connected to ' + dbConfig.DB);
})
```

An Example of how student login works:

```
A Router End point is made here as POST type
1.  router.post('/login', (req, res) => {
2.  console.log(req.body);

The Body must contain a JSON that has user_name and password
3.  const { user_name, password } = req.body;
```

The mysql connection object is used to make a query

```
4.  // fetch the hashed password from the db
5.  connection.query('SELECT id, hashed_password FROM `STUDENT_LOGIN` WHERE `user_name` = ? LIMIT 1
    ', user_name, (error, results, fields) => {
```

If an error occurred in the query then return a STATUS 500 Server Error to Client

```
6.  if (error) {
7.  console.error(error);
8.  res.status(500).json({ message: error });
9.  return;
10. } else if (results.length == 0) {
11. res.status(404).json({ message: 'user not found' });
12. return;
13. }
```

The results are obtained from the query performed

```
14. const user = results[0];
```

BCrypt is a tool to check if the entered password matched the stored password, it uses a salt and a special hashing technique for better hashing, in case the database was stolen all the user passwords are still safe and the hashed passwords cannot be hash match attacked

```
15. // check if the password is correct
16. bcrypt.compare(password, user.hashed_password, (err, result) => {
17. if (result == true) {
18. console.info('user ' + user_name + ' is logging in');
```

We use JSON-Web-Token to generate a token so that every time the client can show its authentication identity to the server, this way we can have resource access to specific type of users

```
19. // create an access token for this user
20. const access_token = jwt.sign({ username: user.user_name, role: 'student' }, jwtconfig.JWT_ACCE
    SS_TOKEN);

21. // fetch the user details from the db
22. connection.query('SELECT * FROM STUDENT WHERE id = ? LIMIT 1', user.id, (error, results, field
    s) => {

23. if (error) {
24. console.error(error);
25. res.status(500).json({ message: error });
26. return;
27. } else if (results.length == 0) {
28. res.status(404).json({ message: 'user not found' });
29. return;
30. }

31. // return the user and the access token
32. res.json({ user: results[0], token: access_token });
33. });
```

If the entered password is wrong then return STATUS 401 UNAUTHORISED

```
34. } else {
35. res.status(401).json({ message: 'incorrect password' });
36. return;
```

```
37. }
38. });
39. });
40. });
```

The rest of the code has a similar working structure and can be referred in Appendix B.

1.4 Implementation of Front-End and connection with Database

Refer to Appendix A for the complete front-end source code.

The Front-End was chosen as ReactJS (JavaScript Web Framework)

Some essential front-end code is described in this section.

```
1.  const tranlucentbg = { backgroundColor: 'rgba(77, 94, 114, 0.5)' };

YUP is used to create an object schema for form validation
2.  const schema = yup.object({
3.  username: yup.string().required(),
4.  password: yup.string().required(),
5.  });

Login is a React Component
6.  class Login extends Component {

7.  constructor(props) {
8.  super(props);
9.  }

When this component is mounted on web browser
10. componentDidMount() {

check if the user is already logged in from the localStorage of the browser
11. const user = JSON.parse(localStorage.getItem('user'));

12. if (user != null) {
13. this.props.history.replace('/home');
14. return;
15. }
16. }

This is the part that is rendered to the web browser
17. render() {
18. return (
19. <Container className="login-page fill-window" fluid>
20. <Row className="m-auto" >
21. <Col sm={9} md={7} lg={6} className="mx-auto my-auto">
```

```
22. <Card className="login-card shadow shadow-lg my-5 p-5 text-
    white rounded" style={tranlucentbg} text="white">
23. <Row className="mb-3">
24. <Col sm={4} md={4} className="my-auto"><Card.Img variant="top" src={logo} /></Col>
25. <Col sm={8} md={8} lg={8} className="h1 my-auto font-weight-bold text-right" >RUAS LMS </Col>
26. </Row>
27. <Row className="h1 mx-auto"> Login </Row>

We use Formik to create Forms easily in React
28. <Formik
29. validationSchema={schema}
30. onSubmit={(values, { setSubmitting, resetForm }) => {

31. setSubmitting(true);

use Axios to make a POST request to our back-end and pass the user_name and password as JSON in
the body of the request
32. axios.post(BASE_URL + 'student/login', { user_name: values.username, password: values.password
    })
33. .then((res) => {
34. console.log(res);

Store the credentials in the localStorage of the Browser
35. localStorage.setItem('user', JSON.stringify(res.data.user));
36. localStorage.setItem('token', res.data.token);

37. resetForm();

38. this.props.history.replace('/home');

39. })
40. .catch((err) => {
41. console.error(err.response.data.message);
42. });

43. setSubmitting(false);
44. }}
45. initialValues={{
46. username: '',
47. password: '',
48. }}
49. >
50. {({
51. handleSubmit,
52. handleChange,
53. handleBlur,
54. values,
55. touched,
56. isValid,
57. errors,
58. }) => (

React-Bootstrap Forms are used to create nice looking forms
59. <Form className="my-2">
```

```
60. <Form.Group>
61. <Form.Label className="h6 mb-3" >Username</Form.Label>
62. <InputGroup>
63. <InputGroup.Prepend>
64. <InputGroup.Text className="bg-light text-
    dark" ><FontAwesomeIcon icon={faUser} /></InputGroup.Text>
65. </InputGroup.Prepend>
66. <Form.Control type="text" name="username" value={values.username} onChange={handleChange} isInv
    alid={!!errors.username} placeholder="17ETCS002159" />
67. <Form.Control.Feedback type="invalid">
68. {errors.username}
69. </Form.Control.Feedback>
70. </InputGroup>
71. </Form.Group>
72. <Form.Group controlId="formBasicPassword">
73. <Form.Label className="h6 mb-3">Password</Form.Label>
74. <InputGroup>
75. <InputGroup.Prepend>
76. <InputGroup.Text className="bg-light text-
    dark" > <FontAwesomeIcon icon={faLock} />  </InputGroup.Text>
77. </InputGroup.Prepend>
78. <Form.Control type="password" name="password" values={values.password} onChange={handleChange}
    isInvalid={!!errors.password} placeholder="************" />
79. <Form.Control.Feedback type="invalid">
80. {errors.password}
81. </Form.Control.Feedback>
82. </InputGroup>
83. </Form.Group>
84. <Row className="mx-auto">
85. <Button variant="dark" className="px-5 mt-3 mb-0 mx-auto" onClick={handleSubmit}>Login</Button>
86. </Row>
87. </Form>
88. )}
89. </Formik>
90. <Button variant="link text-light ml-auto text-right mb-
    0"><Link to="/register">Register</Link> </Button>
91. </Card>
92. </Col>
93. </Row>

94. </Container>
95. );
96. }
97. }
```

1.5 Concluding Remarks

1.5.1 Summary

Deployed Website: **`https://satyajitghana.github.io/ruas-lms/login/`**

`Username: 17ETCS002159`

`Password: satyajit123`

Screenshots of the WebApp using ReactJS + NodeJS + ExpressJS

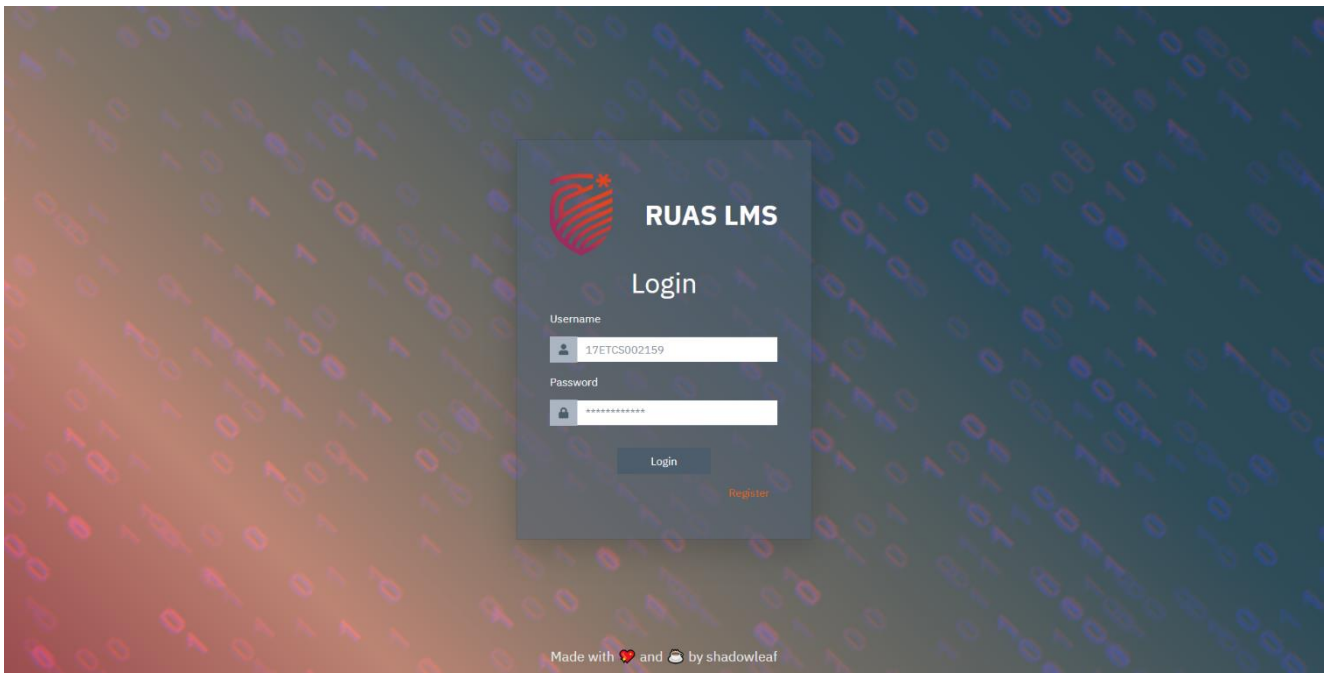

Figure 1-2 Login Page

Here the user is presented with a screen to enter their username and password, the form is taken by React and it creates a `HTTP POST` request to the server at the backend, the password is checked and a token is returned to the front-end, which is saved in the browser memory, now whenever a request is made, this token is checked for authenticity.

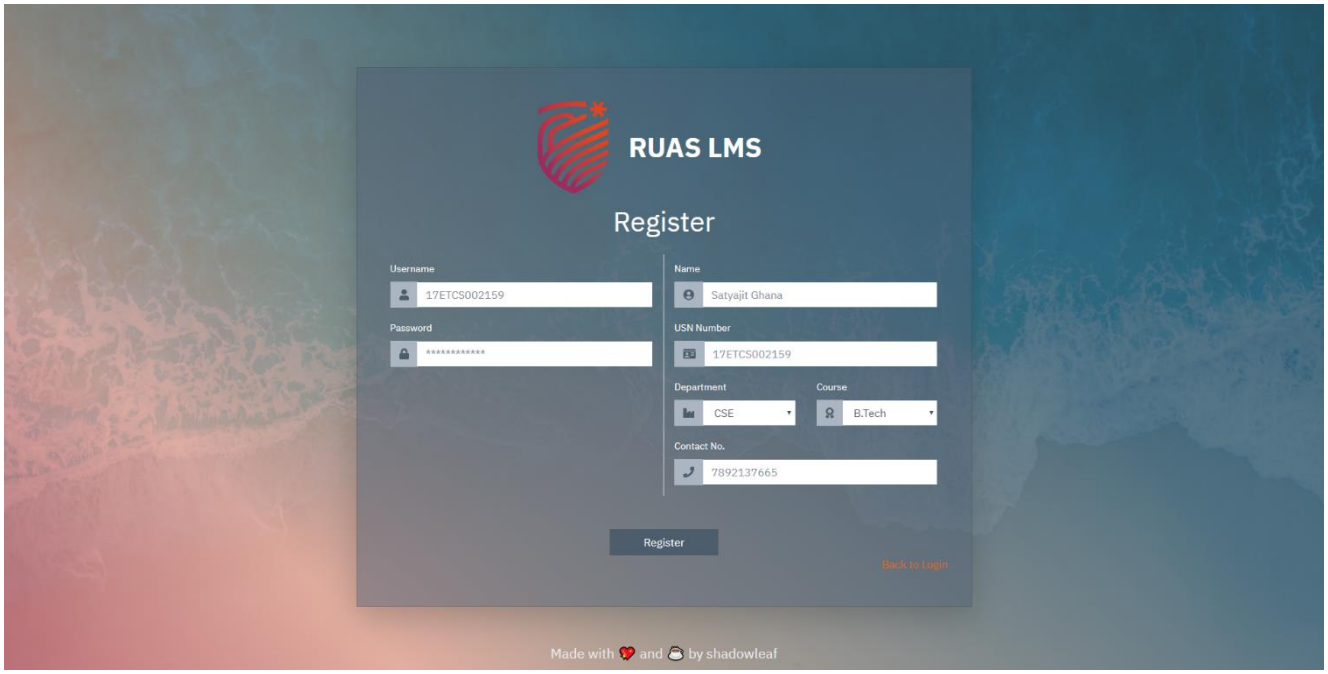

Figure 1-3 Register Page

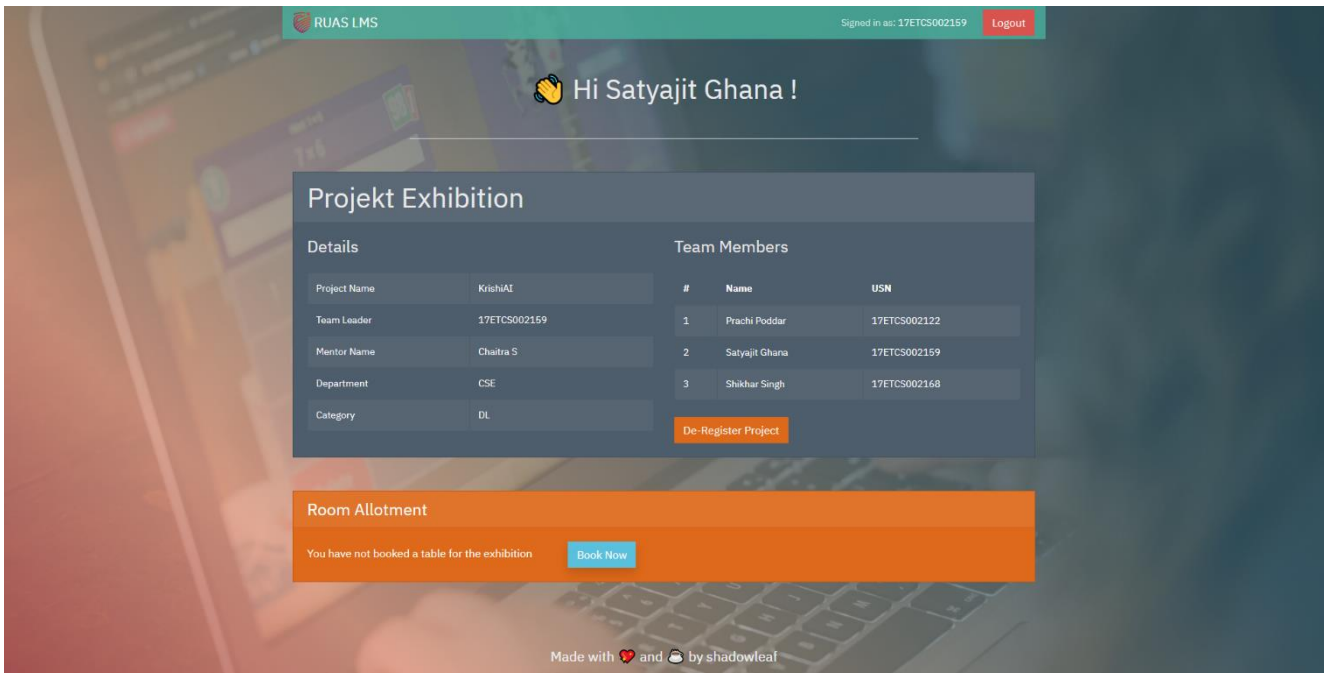

Figure 1-4 Student Home Page

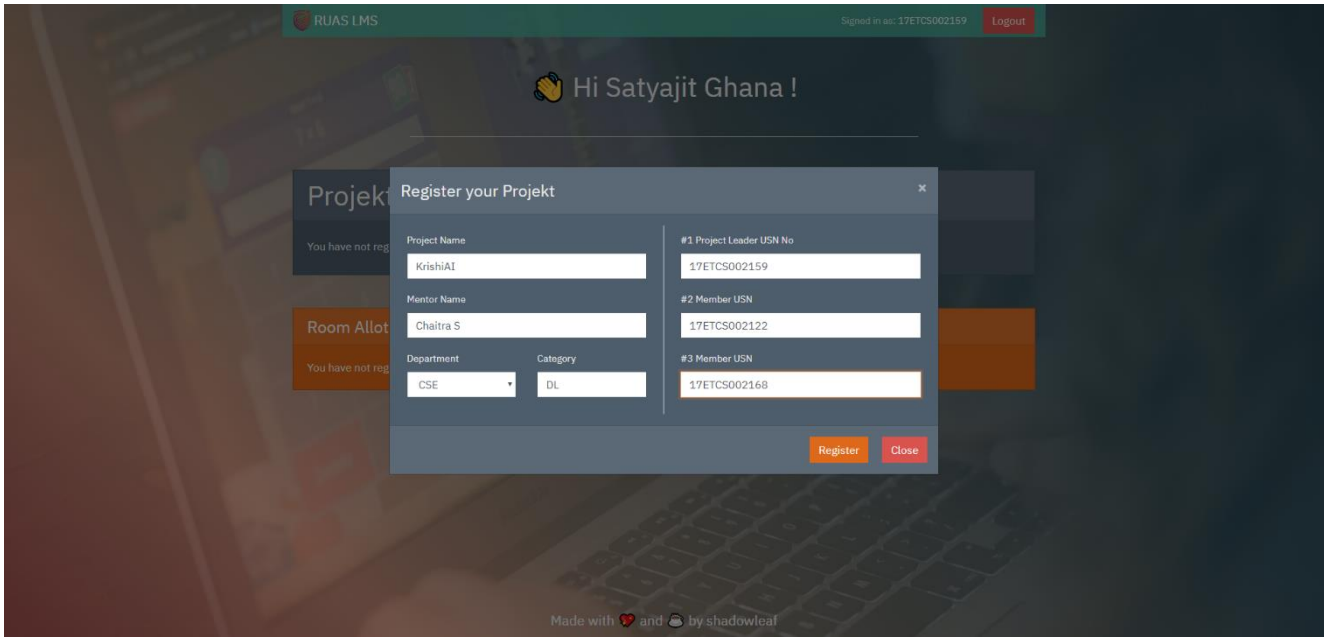

Figure 1-5 Student Project Register Page

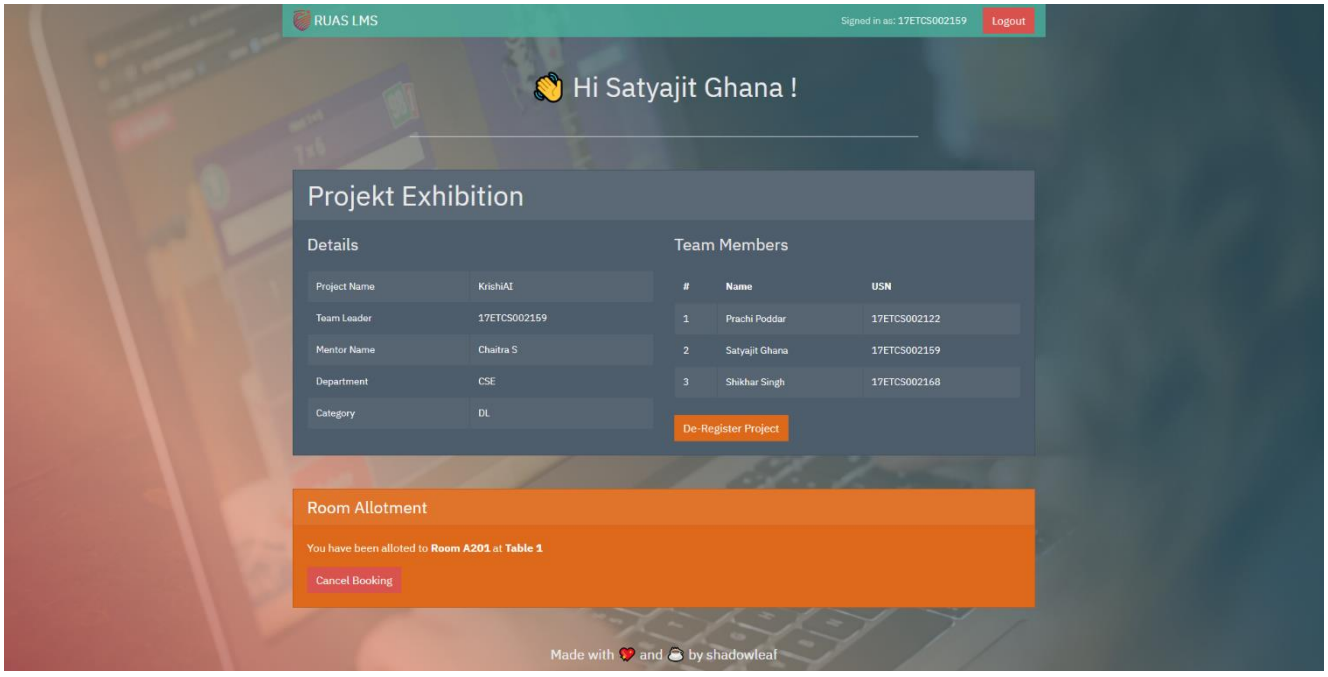

Figure 1-6 Room Allocation of Project

1.5.2 Limitations

- The libraries used by React and Node need to be maintained periodically, to ensure security vulnerabilities are fixed, to help with this, the project was published to GitHub and a Workflow was created, any package updates are notifies to the project, and any contributor can merge the PR to update the project libraries.

1.5.3 Improvements

- The API port exposed by the server is not secured under SSL, all the connections made are unsecure, hence this need to be fixed by adding TLS certificates to the backend.
- Requests made through the server are under CORS(Cross-origin resource sharing), this can be prevented by hosting the server in the same domain as the front end.

Bibliography

Appendix A

The Complete Source Code can be found at **https://github.com/satyajitghana/ruas-lms**

Deployed Website: **https://satyajitghana.github.io/ruas-lms/login/**

`Username: 17ETCS002159`

`Password: satyajit123`

FrontEnd Source Code

App.js

```
import React from 'react';
import './App.css';
import { BrowserRouter, Route, Redirect, Switch } from 'react-router-dom';
import Login from './Login'
import Register from './Register'
import StudentHome from './StudentHome'
import { Container, Navbar, Col, Row } from 'react-bootstrap';

function Footer() {
  return (

    <Container>
      <Navbar variant="dark" fixed="bottom">
        <Container>
          <Col lg={8} className="mx-auto">
            <Row>
              <h5 className="mx-
auto">Made with <span role="img">💔</span> and <span role="img">🍮</span> by shadowleaf</h5>
            </Row>
          </Col>
        </Container>

      </Navbar>
    </Container>
  )
}

function App() {
  return (
    <BrowserRouter basename={process.env.PUBLIC_URL}>
      <Switch>
        <Route path='/login' component={Login} />
        <Route path='/register' component={Register} />
        <Route path='/home' component={StudentHome} />
      </Switch>
      <Footer />
    </BrowserRouter>
```

```
  );
}

export default App;
```

Login.js

```
import React, { Component } from 'react';

import './Login.css';
import { Row, Container, Card, Form, Col, InputGroup, Button, Toast } from 'react-bootstrap';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'

import { faUser, faLock } from '@fortawesome/free-solid-svg-icons'
import * as yup from 'yup';
import logo from './msruas_logo_symbol.png'
import { Redirect, Link } from 'react-router-dom';
import { Formik } from 'formik';
import axios from 'axios';
import { BASE_URL } from './Api';

const tranlucentbg = { backgroundColor: 'rgba(77, 94, 114, 0.5)' };

const schema = yup.object({
    username: yup.string().required(),
    password: yup.string().required(),
});

class Login extends Component {

    constructor(props) {
        super(props);
    }

    componentDidMount() {
        const user = JSON.parse(localStorage.getItem('user'));

        if (user != null) {
            this.props.history.replace('/home');
            return;
        }
    }

    render() {
        return (
            <Container className="login-page fill-window" fluid>
                <Row className="m-auto" >
                    <Col sm={9} md={7} lg={6} className="mx-auto my-auto">
                        <Card className="login-card shadow shadow-lg my-5 p-5 text-
white rounded" style={tranlucentbg} text="white">
                            <Row className="mb-3">
                                <Col sm={4} md={4} className="my-
auto"><Card.Img variant="top" src={logo} /></Col>
                                <Col sm={8} md={8} lg={8} className="h1 my-auto font-weight-bold text-
right" >RUAS LMS </Col>
                            </Row>
                            <Row className="h1 mx-auto"> Login </Row>
```

```
                        <Formik
                            validationSchema={schema}
                            onSubmit={(values, { setSubmitting, resetForm }) => {

                                setSubmitting(true);

                                axios.post(BASE_URL + 'student/login', { user_name: values.username,
 password: values.password })
                                    .then((res) => {
                                        console.log(res);

                                        localStorage.setItem('user', JSON.stringify(res.data.user));
                                        localStorage.setItem('token', res.data.token);

                                        resetForm();

                                        this.props.history.replace('/home');

                                    })
                                    .catch((err) => {
                                        console.error(err.response.data.message);
                                    });

                                setSubmitting(false);
                            }}
                            initialValues={{
                                username: '',
                                password: '',
                            }}
                        >
                            {({
                                handleSubmit,
                                handleChange,
                                handleBlur,
                                values,
                                touched,
                                isValid,
                                errors,
                            }) => (
                                    <Form className="my-2">
                                        <Form.Group>
                                            <Form.Label className="h6 mb-3" >Username</Form.Label>
                                            <InputGroup>
                                                <InputGroup.Prepend>
                                                    <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faUser} /></InputGroup.Text>
                                                </InputGroup.Prepend>
                                                <Form.Control type="text" name="username" value={val
ues.username} onChange={handleChange} isInvalid={!!errors.username} placeholder="17ETCS002159" />
                                                <Form.Control.Feedback type="invalid">
                                                    {errors.username}
                                                </Form.Control.Feedback>
                                            </InputGroup>
                                        </Form.Group>
                                        <Form.Group controlId="formBasicPassword">
                                            <Form.Label className="h6 mb-3">Password</Form.Label>
```

```
                                                <InputGroup>
                                                    <InputGroup.Prepend>
                                                        <InputGroup.Text className="bg-light text-
dark" > <FontAwesomeIcon icon={faLock} />  </InputGroup.Text>
                                                    </InputGroup.Prepend>
                                                    <Form.Control type="password" name="password" values
={values.password} onChange={handleChange} isInvalid={!!errors.password} placeholder="***********" />
                                                    <Form.Control.Feedback type="invalid">
                                                        {errors.password}
                                                    </Form.Control.Feedback>
                                                </InputGroup>
                                            </Form.Group>
                                            <Row className="mx-auto">
                                                <Button variant="dark" className="px-5 mt-3 mb-0 mx-
auto" onClick={handleSubmit}>Login</Button>
                                            </Row>
                                        </Form>
                                    )}
                                </Formik>
                                <Button variant="link text-light ml-auto text-right mb-
0"><Link to="/register">Register</Link> </Button>
                            </Card>
                        </Col>
                    </Row>

                </Container>
            );
        }
}

export default Login;
```

Login.js

```
import React, { Component } from 'react';
import { Container, Row, Col, Card, Form, InputGroup, Button } from 'react-bootstrap';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { faUser, faLock, faUserCircle, faIdCard, faIndustry, faAward, faPhone } from '@fortawesome/free-
solid-svg-icons'
import logo from './msruas_logo_symbol.png'

import axios from 'axios';
import { BASE_URL } from './Api';
import { Formik } from 'formik';
import * as yup from 'yup';

import "./Register.css"
import { Redirect, Link } from 'react-router-dom';

const tranlucentbg = { backgroundColor: 'rgba(77, 94, 114, 0.5)' };

const registerSchema = yup.object({
    user_name: yup.string().required(),
    password: yup.string().required(),
    reg_no: yup.string().required(),
    name: yup.string().required(),
    department: yup.string().required(),
```

```
    course: yup.string().required(),
    contact_no: yup.string().required(),
})

class Register extends Component {

    registerUser(user) {
        axios.post(BASE_URL + 'student/register', user)
            .then((res) => {
                this.props.history.replace('/login');
            })
            .catch(console.error);
    }

    RegisterForm() {
        return (
            <Formik
                validationSchema={registerSchema}
                onSubmit={(values, { setSubmitting, resetForm }) => {
                    setSubmitting(true);
                    this.registerUser(values);
                    setSubmitting(false);
                }}
                initialValues={{
                    user_name: '',
                    password: '',
                    reg_no: '',
                    name: '',
                    department: 'CSE',
                    course: 'B.Tech',
                    contact_no: '',
                }}
            >
                {({
                    handleSubmit,
                    handleChange,
                    handleBlur,
                    values,
                    touched,
                    isValid,
                    errors,
                }) => (
                        <div>
                            <Row>
                                <Col lg={6} className="border-right">
                                    <Form className="my-2">
                                        <Form.Group>
                                            <Form.Label >Username</Form.Label>
                                            <InputGroup>
                                                <InputGroup.Prepend>
                                                    <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faUser} /></InputGroup.Text>
                                                </InputGroup.Prepend>
                                                <Form.Control type="text" name="user_name" value={values
.user_name} onChange={handleChange} placeholder="17ETCS002159" />
                                            </InputGroup>
                                        </Form.Group>
```

```
                                    <Form.Group>
                                        <Form.Label>Password</Form.Label>
                                        <InputGroup>
                                            <InputGroup.Prepend>
                                                <InputGroup.Text className="bg-light text-
dark" > <FontAwesomeIcon icon={faLock} />  </InputGroup.Text>
                                            </InputGroup.Prepend>
                                            <Form.Control type="password" name="password" value={val
ues.password} onChange={handleChange} placeholder="***********" />
                                        </InputGroup>
                                    </Form.Group>
                                </Form>
                            </Col>
                            <Col lg={6}>
                                <Form className="my-2">
                                    <Form.Group>
                                        <Form.Label>Name</Form.Label>
                                        <InputGroup>
                                            <InputGroup.Prepend>
                                                <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faUserCircle} /></InputGroup.Text>
                                            </InputGroup.Prepend>
                                            <Form.Control type="text" name="name" value={values.name
} onChange={handleChange} placeholder="Satyajit Ghana" />
                                        </InputGroup>
                                    </Form.Group>
                                    <Form.Group>
                                        <Form.Label>USN Number</Form.Label>
                                        <InputGroup>
                                            <InputGroup.Prepend>
                                                <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faIdCard} /></InputGroup.Text>
                                            </InputGroup.Prepend>
                                            <Form.Control type="text" name="reg_no" value={values.re
g_no} onChange={handleChange} placeholder="17ETCS002159" />
                                        </InputGroup>
                                    </Form.Group>
                                    <Row>
                                        <Col>
                                            <Form.Group>
                                                <Form.Label>Department</Form.Label>
                                                <InputGroup>
                                                    <InputGroup.Prepend>
                                                        <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faIndustry} /></InputGroup.Text>
                                                    </InputGroup.Prepend>
                                                    <Form.Control as="select" name="department" valu
e={values.department} onChange={handleChange} >
                                                        <option>CSE</option>
                                                        <option>EEE</option>
                                                        <option>ECE</option>
                                                        <option>CIVIL</option>
                                                    </Form.Control>
                                                </InputGroup>
                                            </Form.Group>
                                        </Col>
                                        <Col>
```

```
                                        <Form.Group>
                                            <Form.Label>Course</Form.Label>
                                            <InputGroup>
                                                <InputGroup.Prepend>
                                                    <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faAward} /></InputGroup.Text>
                                                </InputGroup.Prepend>
                                                <Form.Control as="select" name="course" value={v
alues.course} onChange={handleChange} >
                                                    <option>B.Tech</option>
                                                    <option>M.Tech</option>
                                                </Form.Control>
                                            </InputGroup>
                                        </Form.Group>
                                    </Col>
                                </Row>
                                <Form.Group>
                                    <Form.Label>Contact No.</Form.Label>
                                    <InputGroup>
                                        <InputGroup.Prepend>
                                            <InputGroup.Text className="bg-light text-
dark" ><FontAwesomeIcon icon={faPhone} /></InputGroup.Text>
                                        </InputGroup.Prepend>
                                        <Form.Control type="text" name="contact_no" value={value
s.contact_no} onChange={handleChange} placeholder="7892137665" />
                                    </InputGroup>
                                </Form.Group>
                            </Form>
                        </Col>
                    </Row>
                    <Row>
                        <Button variant="dark" className="mt-5 mb-0 mx-auto px-
5" onClick={handleSubmit} >Register</Button>
                    </Row>
                </div>
            )}
        </Formik>
    );
}

render() {
    return (
        <Container className="register-page fill-window" fluid>
            <Container>
                <Row className="m-auto" >

                    <Col sm={8} md={8} lg={10} className="mx-auto my-auto">
                        <Card className="shadow shadow-lg my-5 p-5 text-
white rounded" style={tranlucentbg} text="white">
                            <Row className="mb-3 mx-auto">
                                <Col sm={2} md={2} lg={2} className="ml-
auto"><Card.Img variant="top" src={logo} /></Col>
                                <Col sm={6} md={6} lg={4} className="h1 my-auto font-weight-bold mr-
auto" >RUAS LMS </Col>
                            </Row>
                            <Row className="h1 mx-auto mb-4">Register</Row>
                            {this.RegisterForm()}
```

```
                                <Row>
                                    <Button variant="link text-light m-0 p-0 ml-auto text-right mb-
0"> <Link to="/login">Back to Login </Link></Button>
                                </Row>
                            </Card>
                        </Col>
                    </Row>
                </Container>
            </Container>
        )
    }
}

export default Register;
```

StudentHome.js

```
import React, { Component } from 'react';
import { Container, Row, Button, Col, Table, Card, Spinner, Modal, Form, InputGroup } from 'react-
bootstrap';
import { Navbar } from 'react-bootstrap';
import axios from 'axios';
import { BASE_URL } from './Api';
import { Formik } from 'formik';
import * as yup from 'yup';

import logo from './msruas_logo_symbol.png'

import './StudentHome.css'

const tranlucentbg = { backgroundColor: 'rgba(74, 177, 157, 0.8)' };

const projectRegisterSchema = yup.object({
    project_leader_regno: yup.string().required(),
    project_name: yup.string().required(),
    mentor_name: yup.string().required(),
    department: yup.string().required(),
    category: yup.string().required(),
    member1: yup.string().required(),
    member2: yup.string().required(),
});

class StudentHome extends Component {

    constructor(props) {
        super(props);

        this.state = {
            user: null,
            isLoading: true,
            isRoomDetailsLoading: true,
            roomDetails: null,
            projekt: null,
            error: null,
            showRegisterModal: false
        }
    }
```

```
fetchProjektDetails() {

    const user = JSON.parse(localStorage.getItem('user'));

    if (user != null) {
        axios.get(BASE_URL + 'projekt/from-reg-no/' + user.reg_no)
            .then((res) => {
                this.setState({
                    projekt: res.data,
                    isLoading: false,
                });

                this.getRoomDetails();
            })
            .catch((err) => {
                this.setState({
                    error: err.response.data.message,
                    isLoading: false,
                })
            });
    }
}

deregisterProject() {
    axios.delete(BASE_URL + 'projekt/' + this.state.projekt.id)
        .then((res) => {
            this.setState({
                projekt: null
            });
        });
}

componentDidMount() {

    const user = JSON.parse(localStorage.getItem('user'));

    if (user == null) {
        this.props.history.replace('/login');
        return;
    }

    this.setState({
        user: user,
    });

    this.fetchProjektDetails();

}

registerProjectModal() {
    return (

        <Formik
            validationSchema={projectRegisterSchema}
            onSubmit={(values, { setSubmitting, resetForm }) => {
                setSubmitting(true);
```

```
                    const payload = {
                        project_leader_regno: values.project_leader_regno,
                        project_name: values.project_name,
                        mentor_name: values.mentor_name,
                        department: values.department,
                        category: values.category,
                        students: [values.project_leader_regno, values.member1, values.member2]
                    }

                    axios.post(BASE_URL + 'projekt/register', payload)
                        .then((res) => {
                            // fetch the project details now
                            this.fetchProjektDetails();
                            this.setState({ showRegisterModal: false });
                        })
                        .catch((err) => {
                            console.error(err);
                        });

                    setSubmitting(false);
                }}
                initialValues={{
                    project_leader_regno: '',
                    project_name: '',
                    mentor_name: '',
                    department: 'CSE',
                    category: '',
                    member1: '',
                    member2: '',
                }}
            >
                {({
                    handleSubmit,
                    handleChange,
                    handleBlur,
                    values,
                    touched,
                    isValid,
                    errors,
                }) => (

                        <Modal
                            show={this.state.showRegisterModal}
                            onHide={() => this.setState({ showRegisterModal: false })}
                            size="lg"
                            aria-labelledby="contained-modal-title-vcenter"
                            centered
                        >
                            <Modal.Header closeButton>
                                <Modal.Title id="contained-modal-title-
vcenter">Register your Projekt</Modal.Title>
                            </Modal.Header>
                            <Modal.Body>
                                <Row>
                                    <Col lg={6} className="border-right px-4">
                                        <Form className="my-2">
```

```
                                        <Form.Group>
                                            <Form.Label>Project Name</Form.Label>
                                            <Form.Control type="text" name="project_name" value={val
ues.project_name} onChange={handleChange} placeholder="" />
                                        </Form.Group>
                                        <Form.Group>
                                            <Form.Label>Mentor Name</Form.Label>
                                            <Form.Control type="text" name="mentor_name" value={valu
es.mentor_name} onChange={handleChange} placeholder="" />
                                        </Form.Group>
                                        <Row>
                                            <Col>
                                                <Form.Group>
                                                    <Form.Label>Department</Form.Label>

                                                    <Form.Control as="select" name="department" valu
e={values.department} onChange={handleChange} >
                                                        <option>CSE</option>
                                                        <option>EEE</option>
                                                        <option>ECE</option>
                                                        <option>CIVIL</option>
                                                    </Form.Control>
                                                </Form.Group>
                                            </Col>
                                            <Col>
                                                <Form.Group>
                                                    <Form.Label>Category</Form.Label>
                                                    <Form.Control type="text" name="category" value=
{values.category} onChange={handleChange} placeholder="" />
                                                </Form.Group>
                                            </Col>
                                        </Row>

                                    </Form>
                                </Col>
                                <Col lg={6} className="px-4">
                                    <Form className="my-2">
                                        <Form.Group>
                                            <Form.Label >#1 Project Leader USN No</Form.Label>
                                            <Form.Control type="text" name="project_leader_regno" va
lue={values.project_leader_regno} onChange={handleChange} placeholder="" />
                                        </Form.Group>
                                        <Form.Group>
                                            <Form.Label>#2 Member USN</Form.Label>
                                            <Form.Control type="text" name="member1" value={values.m
ember1} onChange={handleChange} placeholder="" />
                                        </Form.Group>
                                        <Form.Group>
                                            <Form.Label>#3 Member USN</Form.Label>
                                            <Form.Control type="text" name="member2" value={values.m
ember2} onChange={handleChange} placeholder="" />
                                        </Form.Group>
                                    </Form>
                                </Col>
                            </Row>
                        </Modal.Body>
                        <Modal.Footer>
```

```
                                <Button variant="primary" className="mr-
3" onClick={handleSubmit}>Register</Button>
                                <Button variant="danger" onClick={() => this.setState({ showRegisterModa
l: false })}>Close</Button>
                            </Modal.Footer>
                        </Modal>
                    )}

            </Formik>

        );
    }

    getRoomDetails() {
        if (this.state.user == null)
            return;

        if (this.state.projekt == null)
            this.setState({ isRoomDetailsLoading: false, roomDetails: null })

        axios.get(BASE_URL + 'exhibition/' + this.state.projekt.id)
            .then((res) => {
                console.log('Hi');
                this.setState({
                    isRoomDetailsLoading: false,
                    roomDetails: res.data,
                });
            })
            .catch((err) => {
                this.setState({
                    isRoomDetailsLoading: false,
                    roomDetails: null,
                })
            });
    }

    bookTable() {
        if (this.state.projekt == null)
            return;

        axios.post(BASE_URL + 'exhibition/register/' + this.state.projekt.id)
            .then((res) => {
                this.getRoomDetails();
            });
    }

    RoomDetails() {
        if (this.state.isRoomDetailsLoading) {

            return (
                <Row className="mx-auto">
                    <Col className="mx-auto">
                        <Spinner animation="grow" /> Loading . . .
                                    </Col>
                </Row>
            )
        }
```

```
        if (this.state.projekt == null) {

            return (
                <Row>
                    <Col md={6} xl={6} className="">You have not register a project yet </Col>
                </Row>
            )
        }

        if (this.state.roomDetails == null) {
            return (
                <Row>
                    <Col md={6} xl={6} className="">You have not booked a table for the exhibition <Butt
on variant="info" className="ml-5 shadow" onClick={() => { this.bookTable(); }} >Book Now</Button></Col>
                </Row>
            )
        }

        return (
            <div>
                <Card.Text>
                    You have been alloted to <strong>{' '} Room {this.state.roomDetails.room_name}</stro
ng> at <strong>{' '} Table {this.state.roomDetails.table_no}</strong>
                </Card.Text>
                <Button variant="danger" onClick={() => this.deregisterTable()} >Cancel Booking</Button>
            </div>
        );
    }

    deregisterTable() {
        axios.delete(BASE_URL + 'exhibition/deregister/' + this.state.projekt.id)
            .then((res) => {
                this.setState({
                    roomDetails: null,
                });
                this.getRoomDetails();
            });
    }

    ProjektDetails() {

        if (this.state.isLoading)

            return (
                <Row className="mx-auto">
                    <Col className="mx-auto">
                        <Spinner animation="grow" /> Loading . . .
                                    </Col>
                </Row>
            )

        if (this.state.projekt == null) {
            return (
                <Row>
```

```
                    <Col md={6} xl={6} className="">You have not registered your group project <Button v
ariant="info" className="ml-
5 shadow" onClick={() => this.setState({ showRegisterModal: true })} >Register Now</Button></Col>
                </Row>
            )
        }

        return (
            <Row>
                <Col md={6} xl={6} className="">

                    <h4 className="mb-4" >Details</h4>
                    <Table striped bordered variant="dark">
                        <tbody>
                            <tr>
                                <td>Project Name</td>
                                <td>{this.state.projekt.project_name}</td>
                            </tr>
                            <tr>
                                <td>Team Leader</td>
                                <td>{this.state.projekt.project_leader_regno}</td>
                            </tr>
                            <tr>
                                <td>Mentor Name</td>
                                <td>{this.state.projekt.mentor_name}</td>
                            </tr>
                            <tr>
                                <td>Department</td>
                                <td>{this.state.projekt.department}</td>
                            </tr>
                            <tr>
                                <td>Category</td>
                                <td>{this.state.projekt.category}</td>
                            </tr>
                        </tbody>
                    </Table>
                </Col>
                <Col md={6} xl={6} className="">
                    <h4 className="mb-4" >Team Members</h4>
                    <Table striped bordered variant="dark">
                        <thead>
                            <tr>
                                <th>#</th>
                                <th>Name</th>
                                <th>USN</th>
                            </tr>
                        </thead>
                        <tbody>

                            {this.state.projekt.members.map((e, i) => {
                                return (
                                    <tr key={i}>
                                        <td>{i + 1}</td>
                                        <td>{e.name}</td>
                                        <td>{e.reg_no}</td>
                                    </tr>
                                );
```

```
                            })}

                        </tbody>
                    </Table>
                    <Button className="mt-2 warning" onClick={() => this.deregisterProject()} >De-
Register Project</Button>
                </Col>
            </Row>
        )
    }

    checkAuthentication() {

    }

    render() {

        const user = JSON.parse(localStorage.getItem('user'));

        if (user == null) {
            this.props.history.replace('/login');

            return (
                <h1>401: UNAUTHORIZED</h1>
            )
        }

        return (
            <div className="home-page meow">
                {this.registerProjectModal()}
                <Container>
                    <Navbar variant="dark" expand="lg" sticky="top" style={tranlucentbg}>
                        <Navbar.Brand href="#home">
                            <img
                                alt=""
                                src={logo}
                                width="25"
                                height="30"
                                className="align-top font-weight-bold"
                            />
                            {' '} RUAS LMS
                        </Navbar.Brand>
                        <Navbar.Toggle />
                        <Navbar.Collapse className="justify-content-end">
                            <Navbar.Text>
                                Signed in as: <a href="#login">{user.reg_no}</a>
                            </Navbar.Text>

                            <Button variant="danger" className="ml-4" onClick={() => {
                                localStorage.removeItem('token');
                                localStorage.removeItem('user');

                                this.props.history.replace('/login');
                            }} >Logout</Button>
                        </Navbar.Collapse>
                    </Navbar>
                </Container>
```

```
                <Container className="">
                    <Col lg={8} className="mx-auto mb-5">
                        <Row className="border-bottom">
                            <h1 className="mx-auto my-5">👋 Hi {user.name} !</h1>
                        </Row>
                    </Col>
                    <Col className="mb-5">
                        <Card>
                            <Card.Header as="h1">Projekt Exhibition</Card.Header>
                            <Card.Body>
                                {this.ProjektDetails()}
                            </Card.Body>
                        </Card>
                    </Col>

                    <Col>
                        <Card bg='primary' text='white'>
                            <Card.Header as="h4">Room Allotment</Card.Header>
                            <Card.Body>
                                {this.RoomDetails()}
                            </Card.Body>
                        </Card>
                    </Col>
                </Container>

                <Container>
                    <Navbar variant="dark" fixed="bottom">
                        <Container>
                            <Col lg={8} className="mx-auto">
                                <Row>
                                    <h5 className="mx-auto">Made with 💔 and ☕ by shadowleaf</h5>
                                </Row>
                            </Col>
                        </Container>

                    </Navbar>
                </Container>
            </div>
        )
    }
}

export default StudentHome;
```

Appendix B

The Complete Source Code can be found at **https://github.com/satyajitghana/ruas-lms**

BackEnd Source Code

server.js

```
import app from './app';

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {
    console.log('RUAS Server is running on port ' + PORT)
});
```

app.js

```
import express from 'express';
import bodyparser from 'body-parser';
import stafflogin from './routes/staff';
import student from './routes/student';
import projekt from './routes/projekt';
import project_exhibitions from './routes/project-exhibitions';
import cors from 'cors';

const app = express();

app.use(bodyparser.json());
app.use(bodyparser.urlencoded({ extended: true }));
app.use(cors());

app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", '*');
    res.header("Access-Control-Allow-Credentials", true);
    res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS');
    res.header("Access-Control-Allow-Headers", 'Origin,X-Requested-With,Content-
Type,Accept,content-type,application/json');
    next();
});

app.get('/', (req, res) => {
    res.json({ message: 'Welcome to RUAS Server' })
});

app.use('/staff', stafflogin);
app.use('/student', student);
app.use('/projekt', projekt);
app.use('/exhibition', project_exhibitions);

app.use((err, req, res, next) => {
```

```
    res.status(err.status || 501);
    res.json({
        message: err.message
    });
});

module.exports = app;
```

db.js

```
const mysql = require('mysql');
const dbConfig = require('../config/db.config');

const connection = mysql.createConnection({
    host: dbConfig.HOST,
    user: dbConfig.USER,
    password: dbConfig.PASSWORD,
    database: dbConfig.DB
});

connection.connect(error => {
    if (error) throw error;
    console.log('mysql successfully connected to ' + dbConfig.DB);
})

module.exports = connection;
```

student.js

```
import connection from './db'
import jwtconfig from '../config/jwt.config';
import bcrypt from 'bcrypt';

import jwt from 'jsonwebtoken';
import express from 'express';
const router = express.Router();

// user login using user_name and password
// returns user details and token
// test POST request ✅ PASS
// {
//  "user_name": "17ETCS002159",
//  "password": "satyajit123"
// }
router.post('/login', (req, res) => {
    console.log(req.body);

    const { user_name, password } = req.body;

    // fetch the hashed password from the db
    connection.query('SELECT id, hashed_password FROM `STUDENT_LOGIN` WHERE `user_name` = ? LIMIT 1
', user_name, (error, results, fields) => {
        if (error) {
            console.error(error);
            res.status(500).json({ message: error });
            return;
```

```
        } else if (results.length == 0) {
            res.status(404).json({ message: 'user not found' });
            return;
        }

        const user = results[0];

        // check if the password is correct
        bcrypt.compare(password, user.hashed_password, (err, result) => {
            if (result == true) {
                console.info('user ' + user_name + ' is logging in');

                // create an access token for this user
                const access_token = jwt.sign({ username: user.user_name, role: 'student' }, jwtcon
fig.JWT_ACCESS_TOKEN);

                // fetch the user details from the db
                connection.query('SELECT * FROM STUDENT WHERE id = ? LIMIT 1', user.id, (error, re
sults, fields) => {

                    if (error) {
                        console.error(error);
                        res.status(500).json({ message: error });
                        return;
                    } else if (results.length == 0) {
                        res.status(404).json({ message: 'user not found' });
                        return;
                    }

                    // return the user and the access token
                    res.json({ user: results[0], token: access_token });

                });
            } else {
                res.status(401).json({ message: 'incorrect password' });
                return;
            }
        });
    });
});

// register student account with user_name and password
// test POST request ✅ PASS
// {
//   "user_name": "17ETCS002159",
//   "password": "satyajit123",
//   "reg_no": "17ETCS002159",
//   "name": "Satyajit Ghana",
//   "department": "CSE",
//   "course": "B.Tech",
//   "contact_no": "9898989898"
// }
router.post('/register', (req, res) => {
```

```
    const { user_name, password } = req.body;
    const { reg_no, name, department, course, contact_no } = req.body;

    const saltRounds = 10;

    // create a hashed password with salt
    bcrypt.hash(password, saltRounds, (err, hash) => {

        if (err) {
            console.error(err);
            res.status(500).json({ message: err });
            return;
        }

        connection.beginTransaction((err) => {
            if (err) {
                console.error(err);
                res.status(500).json({ message: err });
                return;
            }

            // create the login credentials row
            connection.query('INSERT INTO STUDENT_LOGIN( user_name, hashed_password ) VALUE (?)', [
[user_name, hash]], (errors, results, fields) => {
                if (errors) {
                    console.error(errors);
                    connection.rollback(() => {
                        res.status(500).json({ message: errors });
                    });
                    return;
                }

                const id = results.insertId;

                // add the students details to the table
                connection.query('INSERT INTO STUDENT( id, reg_no, name, department, course, contac
t_no ) VALUES (?)', [[id, reg_no, name, department, course, contact_no]], (errors, results, fields
) => {
                    if (errors) {
                        console.error(errors);
                        connection.rollback(() => {
                            res.status(500).json({ message: errors });
                        });
                        return;
                    }

                    // commit the transaction
                    connection.commit((err) => {
                        if (err) {
                            console.error(err);
                            connection.rollback(() => {
                                res.status(500).json({ message: err });
                            });
                            return;
```

```
                        }

                        res.json({ message: 'user added successfully' });
                    });
                });
            });
        });
    });
});

module.exports = router;
```

staff.js

```
import connection from './db'
import jwtconfig from '../config/jwt.config';

import jwt from 'jsonwebtoken';
import express from 'express';
const router = express.Router();

router.post('/login', (req, res) => {

    const { user_name, hashed_password } = req.body;

    connection.query('SELECT `hashed_password` FROM `STAFF_LOGIN` WHERE `user_name` = ? LIMIT 1', u
ser_name, (error, results, fields) => {
        if (error) {
            console.log(error);
            res.status(500).json({ message: error });
            throw error;
        } else if (results.length == 0) {
            res.status(404).json({ message: 'user not found' });
            return;
        }

        const user = results[0];

        if (user.hashed_password === hashed_password) {
            const access_token = jwt.sign({ username: user.user_name, role: 'staff' }, jwtconfig.J
WT_ACCESS_TOKEN);
            res.json({ access_token });
        } else {
            res.status(401).json({ message: 'incorrect password' });
        }

    });
});

// these are debug routes, dont use in production
router.get('/', (req, res, next) => {
    connection.query('SELECT * FROM STAFF_LOGIN', (error, results, fields) => {
        if (error) {
            console.log(error);
```

```
            res.status(500).json({ message: error });
            throw error;
        }
        res.json(results);
    });
});

router.get('/:user_name', (req, res) => {
    connection.query('SELECT * FROM `STAFF_LOGIN` WHERE `user_name` = ? LIMIT 1', [req.params.user_
name], (error, results, fields) => {
        if (error) {
            console.log(error);
            res.status(500).json({ message: error });
            throw error;
        } else if (results.length == 0) {
            res.status(404).json({ message: 'user not found' });
        }
        res.status(200).json(results[0]);
    });
});

module.exports = router;
```

projekt.js

```
import connection from './db'
import express from 'express';
const router = express.Router();

/**
 * registers the group project into the database
 *
 * @param req.body = {
 * project_leader_regno, project_name, mentor_name, department, category
 * }
 *
 * POST request tested works ✅
 {
    "project_leader_regno": "17ETCS002159",
    "project_name": "KrishiAI",
    "mentor_name": "Chaitra S",
    "department": "CSE",
    "category": "DL",
    "students": ["17ETCS002159"]
}
 *
 *
 * @author shadowleaf
 */
router.post('/register', (req, res) => {
    const { project_leader_regno, project_name, mentor_name, department, category } = req.body;

    // begin a transaction
    connection.beginTransaction((err) => {
```

```
        if (err) {
            console.error(err);
            res.status(500).json({ message: err });
            return;
        }

        // insert the project details into PROJEKT table
        connection.query('INSERT INTO PROJEKT(project_leader_regno, project_name, mentor_name, depa
rtment, category) VALUE (?)', [[project_leader_regno, project_name, mentor_name, department, catego
ry]], (errors, results, fields) => {

            if (errors) {
                console.error(errors);
                connection.rollback(() => {
                    res.status(500).json({ message: errors });
                });
                return;
            }

            const project_id = results.insertId;

            // students to be registered for this project
            const { students } = req.body;
            const student_insert = students.map(obj => [project_id, obj]);

            // insert the student details
            connection.query("INSERT INTO PROJECT_STUDENT_REGISTER(project_id, student_reg_no) VALU
ES ?", [student_insert], (errors, results, fields) => {

                if (errors) {
                    console.error(errors);
                    connection.rollback(() => {
                        res.status(500).json({ message: errors });
                    });
                    return;
                }

                connection.commit((err) => {
                    if (err) {
                        console.error(err);
                        connection.rollback(() => {
                            res.status(500).json({ message: errors });
                        });
                        return;
                    }

                    res.json({ message: 'students registered for project successfully' });
                });
            });
        });
    });
});

// deletes a project from the table given its id
```

```
router.delete('/:id', (req, res) => {
    const proj_id = req.params.id;

    connection.beginTransaction((err) => {
        if (err) {
            console.error(err);
            res.status(500).json({ message: err });
            return;
        }

        connection.query('DELETE FROM PROJECT_STUDENT_REGISTER WHERE project_id = ?', proj_id, (er
ror, result, fields) => {
            if (error) {
                console.error(error);
                connection.rollback(() => {
                    res.status(500).json({ message: error });
                });
                return;
            }

            connection.query('DELETE FROM PROJEKT WHERE id = ?', proj_id, (error, results, fields)
 => {
                if (error) {
                    console.error(error);
                    connection.rollback(() => {
                        res.status(500).json({ message: error });
                    });
                    return;
                }

                connection.commit((err) => {
                    if (err) {
                        console.error(err);
                        connection.rollback(() => {
                            res.status(500).json({ message: errors });
                        });
                        return;
                    }

                    res.json({ message: 'project registration deleted successfully' });
                });
            });
        });
    });
});

// fetched a project given a group member reg_no
router.get('/from-reg-no/:student_reg_no', (req, res) => {
    const std_reg_no = req.params.student_reg_no;

    connection.query('SELECT project_id FROM PROJECT_STUDENT_REGISTER where student_reg_no = ?', st
d_reg_no, (error, result, fields) => {
        if (error) {
            console.error(error);
```

```
            res.status(404).json({ message: error });
            return;
        }

        if (result.length === 0) {
            res.status(404).json({ message: 'project not found' });
            return;
        }

        const proj_id = result[0].project_id;

        connection.query('SELECT * FROM PROJEKT WHERE id = ?', proj_id, (error, result, fields) =>
 {

            if (error) {
                console.error(error);
                res.status(404).json({ message: error });
                return;
            }

            if (result.length === 0) {
                res.status(404).json({ message: 'project not found' });
                return;
            }

            const project = result[0];

            connection.query('select p.student_reg_no as reg_no, c.name from PROJECT_STUDENT_REGIST
ER p JOIN STUDENT c ON c.reg_no = p.student_reg_no WHERE p.project_id = ?', project.id, (error, re
sult, fields) => {

                if (error) {
                    console.error(error);
                    res.status(404).json({ message: error });
                    return;
                }

                if (result.length === 0) {
                    res.status(404).json({ message: 'students registered not found' });
                    return;
                }

                project.members = result;

                res.json(project);
            });

        });
    });
});

module.exports = router;
```

project-exhibition.js

```
import connection from './db'
import express from 'express';
const router = express.Router();

// gets the room allocation details for a given project_id
router.get('/:project_id', (req, res) => {

    connection.query('SELECT a.room_id, b.room_name, a.table_no FROM PROJECT_EXHIBITION a join EXHI
BITION b ON a.room_id = b.room_id WHERE a.project_id = ? LIMIT 1', req.params.project_id, (error,
results, fields) => {

        if (error) {
            console.error(error);
            res.status(500).json({ message: error });
            return;
        }

        if (results.length === 0) {
            res.status(500).json({ message: 'table has not been booked for this project' });
        }

        res.json(results[0]);

    });
});

/**
 * registers for a project, given its project id
 * @author shadowleaf
 */
router.post('/register/:project_id', (req, res) => {

    // find a room that has space
    connection.query('SELECT t1.room_id, t1.room_name, t1.capacity, IFNULL(t2.count, 0) AS count FR
OM EXHIBITION t1 LEFT JOIN (select room_id, count(table_no) as count from PROJECT_EXHIBITION group
by room_id) as t2 on t1.room_id = t2.room_id AND t2.count <= t1.capacity LIMIT 1', (error, results
, field) => {
        if (error) {
            console.error(error);
            res.status(500).json({ message: error });
            return;
        }

        if (results[0].capacity === results[0].count || results.length === 0) {
            res.status(500).json({ message: 'no empty rooms found' });
            return;
        }

        // check if the room is completely empty
        if (results[0].count === 0) {
            // fill in table 1
            connection.query(`INSERT INTO PROJECT_EXHIBITION VALUES (${results[0].room_id}, ${req.p
arams.project_id}, 1)`, (error, results, fields) => {
```

```
                if (error) {
                    console.error(error);
                    res.status(500).json({ message: error });
                    return;
                }

                res.json({ message: 'project registered successfully', id: results.insertId });
            });
        } else {
            // find the next empty table
            connection.query(`SELECT a.room_id, a.table_no AS beforegap, a.table_no + 1 AS avail FR
OM PROJECT_EXHIBITION a WHERE (SELECT b.table_no FROM PROJECT_EXHIBITION b WHERE b.table_no = a.tab
le_no + 1) IS NULL AND a.table_no + 1 <= (SELECT capacity FROM EXHIBITION where room_id = a.room_id
) AND a.room_id = ${results[0].room_id} LIMIT 1`, (error, results, fields) => {
                if (error) {
                    console.error(error);
                    res.status(500).json({ message: error });
                    return;
                }

                if (results.length === 0) {
                    res.status(500).json({ message: 'no available tables found in rooms' });
                    return;
                }

                const next_available_table = results[0].avail;

                // fill in the next available table
                connection.query(`INSERT INTO PROJECT_EXHIBITION VALUES (${results[0].room_id}, ${r
eq.params.project_id}, ${next_available_table})`, (error, results, fields) => {
                    if (error) {
                        console.error(error);
                        res.status(500).json({ message: error });
                        return;
                    }

                    res.json({ message: 'project registered successfully', id: results.insertId });
                });
            });
        }
    });

});

// de-register a project given its proj_id
router.delete('/deregister/:proj_id', (req, res) => {

    const proj_id = req.params.proj_id;

    connection.query('DELETE FROM PROJECT_EXHIBITION WHERE project_id = ?', proj_id, (error, resul
t, fields) => {

        if (error) {
            console.error(error);
```

```
            res.status(500).json({ message: error });
            return;
        }

        res.json({ message: 'project deregistered successfully' });
    });

});

module.exports = router;
```