|  |  |
| --- | --- |
| **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

[Declaration Sheet ii](#_Toc38893607)

[Contents iii](#_Toc38893608)

[List Of Figures iv](#_Toc38893609)

[1 Question A 5](#_Toc38893610)

[1.1 Functional and Data Requirements 5](#_Toc38893611)

[1.1.1 Functional Requirements 5](#_Toc38893612)

[1.1.2 Data Requirements 8](#_Toc38893613)

[1.2 Design 11](#_Toc38893614)

[1.2.1 ER Diagram 11](#_Toc38893615)

[1.2.2 Relational Schema 12](#_Toc38893616)

[1.2.3 Table Attributes 13](#_Toc38893617)

[1.3 Implementation of Back-End 15](#_Toc38893618)

[1.4 Implementation of Front-End and connection with Database 17](#_Toc38893619)

[1.5 Concluding Remarks 20](#_Toc38893620)

[1.5.1 Summary 20](#_Toc38893621)

[1.5.2 Limitations 22](#_Toc38893622)

[1.5.3 Improvements 23](#_Toc38893623)

[Bibliography 24](#_Toc38893624)

[Appendix A 25](#_Toc38893625)

[Appendix B 41](#_Toc38893626)

# List Of Figures

[Figure 1‑1 Project Exhibition ER Diagram 11](#_Toc38893627)

[Figure 1‑2 Login Page 20](#_Toc38893628)

[Figure 1‑3 Register Page 21](#_Toc38893629)

[Figure 1‑4 Student Home Page 21](#_Toc38893630)

[Figure 1‑5 Student Project Register Page 22](#_Toc38893631)

[Figure 1‑6 Room Allocation of Project 22](#_Toc38893632)

# Question A

Solution to Question A

## Functional and Data Requirements

### 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 |

### 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 |

## Design

### ER Diagram

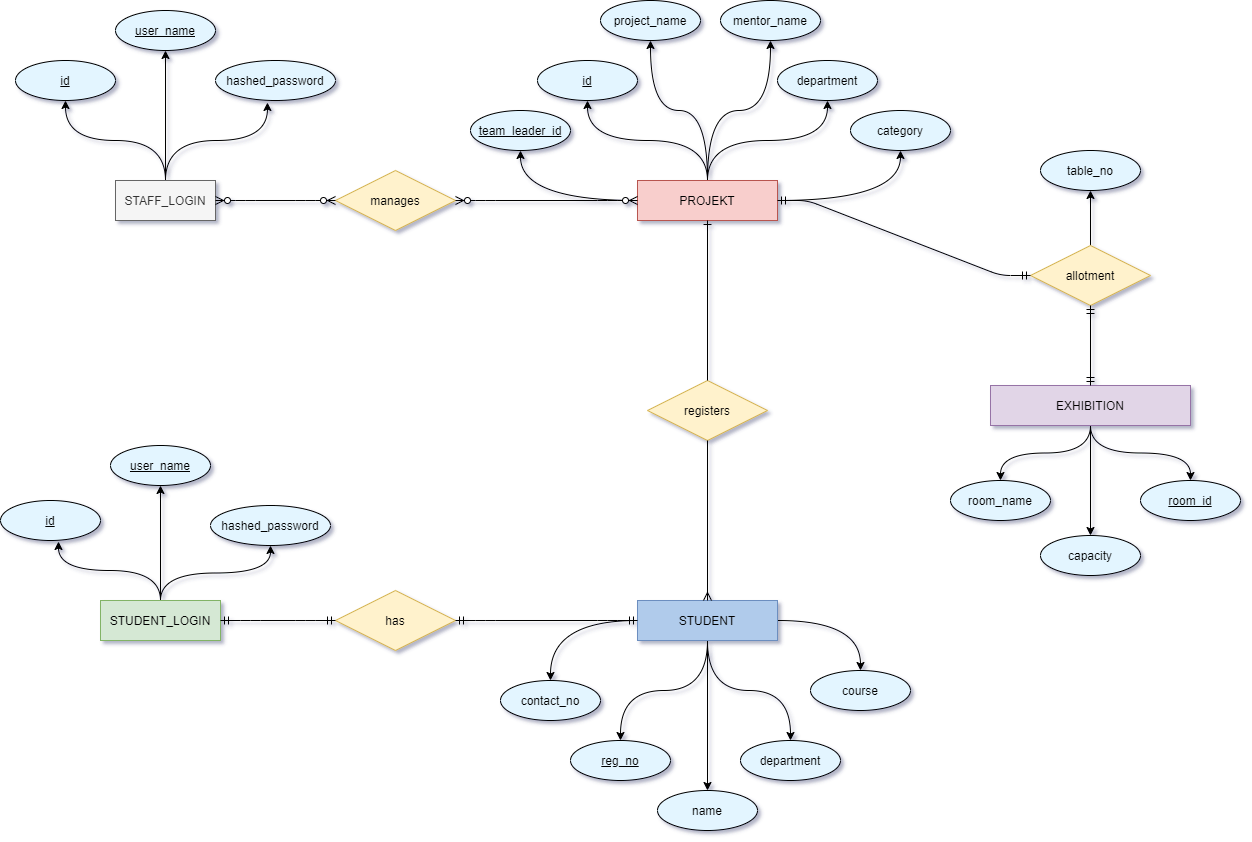


Figure ‑ Project Exhibition ER Diagram

### 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 |

### 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);

## 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**

1. const { user\_name, password } **=** req**.***body*;

**The mysql connection object is used to make a query**

1. *// fetch the hashed password from the db*
2. 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**

1. **if** (error) {
2. console**.**error(error);
3. res**.**status(500)**.**json({ message: error });
4. **return**;
5. } **else** **if** (results**.**length **==** 0) {
6. res**.**status(404)**.**json({ message: 'user not found' });
7. **return**;
8. }

**The results are obtained from the query performed**

1. 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**

1. *// check if the password is correct*
2. bcrypt**.**compare(password, user**.***hashed\_password*, (**err**, **result**) **=>** {
3. **if** (result **==** true) {
4. 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**

1. *// create an access token for this user*
2. const access\_token **=** jwt**.**sign({ username: user**.***user\_name*, role: 'student' }, jwtconfig**.**JWT\_ACCESS\_TOKEN);
3. *// fetch the user details from the db*
4. connection**.**query('SELECT \* FROM STUDENT WHERE id = ? LIMIT 1', user**.***id*, (**error**, **results**, **fields**) **=>** {
5. **if** (error) {
6. console**.**error(error);
7. res**.**status(500)**.**json({ message: error });
8. **return**;
9. } **else** **if** (results**.**length **==** 0) {
10. res**.**status(404)**.**json({ message: 'user not found' });
11. **return**;
12. }
13. *// return the user and the access token*
14. res**.**json({ user: results[0], token: access\_token });
15. });

**If the entered password is wrong then return STATUS 401 UNAUTHORISED**

1. } **else** {
2. res**.**status(401)**.**json({ message: 'incorrect password' });
3. **return**;
4. }
5. });
6. });
7. });

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

## 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**

1. const schema **=** yup**.**object({
2. username: yup**.**string()**.**required(),
3. password: yup**.**string()**.**required(),
4. });

**Login is a React Component**

1. class Login *extends* *Component* {
2. constructor(**props**) {
3. *super*(props);
4. }

**When this component is mounted on web browser**

1. componentDidMount() {

**check if the user is already logged in from the localStorage of the browser**

1. const user **=** JSON**.**parse(localStorage**.**getItem('user'));
2. **if** (user **!=** null) {
3. *this***.***props***.***history***.**replace('/home');
4. **return**;
5. }
6. }

**This is the part that is rendered to the web browser**

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

**We use Formik to create Forms easily in React**

1. <Formik
2. *validationSchema***=**{schema}
3. *onSubmit***=**{(**values**, { **setSubmitting**, **resetForm** }) **=>** {
4. 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**

1. axios**.**post(BASE\_URL **+** 'student/login', { user\_name: values**.***username*, password: values**.***password* })
2. **.**then((**res**) **=>** {
3. console**.**log(res);

**Store the credentials in the localStorage of the Browser**

1. localStorage**.**setItem('user', JSON**.**stringify(res**.***data***.***user*));
2. localStorage**.**setItem('token', res**.***data***.***token*);
3. resetForm();
4. *this***.***props***.***history***.**replace('/home');
5. })
6. **.**catch((**err**) **=>** {
7. console**.**error(err**.***response***.***data***.***message*);
8. });
9. setSubmitting(false);
10. }}
11. *initialValues***=**{{
12. username: '',
13. password: '',
14. }}
15. >
16. {({
17. handleSubmit,
18. handleChange,
19. handleBlur,
20. values,
21. touched,
22. isValid,
23. errors,
24. }) **=>** (

**React-Bootstrap Forms are used to create nice looking forms**

1. <Form *className***=**"my-2">
2. <Form.Group>
3. <Form.Label *className***=**"h6 mb-3" >Username</Form.Label>
4. <InputGroup>
5. <InputGroup.Prepend>
6. <InputGroup.Text *className***=**"bg-light text-dark" ><FontAwesomeIcon *icon***=**{faUser} /></InputGroup.Text>
7. </InputGroup.Prepend>
8. <Form.Control *type***=**"text" *name***=**"username" *value***=**{values**.***username*} *onChange***=**{handleChange} *isInvalid***=**{**!!**errors**.***username*} *placeholder***=**"17ETCS002159" />
9. <Form.Control.Feedback *type***=**"invalid">
10. {errors**.***username*}
11. </Form.Control.Feedback>
12. </InputGroup>
13. </Form.Group>
14. <Form.Group *controlId***=**"formBasicPassword">
15. <Form.Label *className***=**"h6 mb-3">Password</Form.Label>
16. <InputGroup>
17. <InputGroup.Prepend>
18. <InputGroup.Text *className***=**"bg-light text-dark" > <FontAwesomeIcon *icon***=**{faLock} />  </InputGroup.Text>
19. </InputGroup.Prepend>
20. <Form.Control *type***=**"password" *name***=**"password" *values***=**{values**.***password*} *onChange***=**{handleChange} *isInvalid***=**{**!!**errors**.***password*} *placeholder***=**"\*\*\*\*\*\*\*\*\*\*\*\*" />
21. <Form.Control.Feedback *type***=**"invalid">
22. {errors**.***password*}
23. </Form.Control.Feedback>
24. </InputGroup>
25. </Form.Group>
26. <Row *className***=**"mx-auto">
27. <Button *variant***=**"dark" *className***=**"px-5 mt-3 mb-0 mx-auto" *onClick***=**{handleSubmit}>Login</Button>
28. </Row>
29. </Form>
30. )}
31. </Formik>
32. <Button *variant***=**"link text-light ml-auto text-right mb-0"><Link *to***=**"/register">Register</Link> </Button>
33. </Card>
34. </Col>
35. </Row>
36. </Container>
37. );
38. }
39. }

## Concluding Remarks

### Summary

Deployed Website: [**https://satyajitghana.github.io/ruas-lms/login/**](https://satyajitghana.github.io/ruas-lms/login/)

Username: 17ETCS002159

Password: satyajit123

Screenshots of the WebApp using ReactJS + NodeJS + ExpressJS



Figure ‑ 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 ‑ Register Page

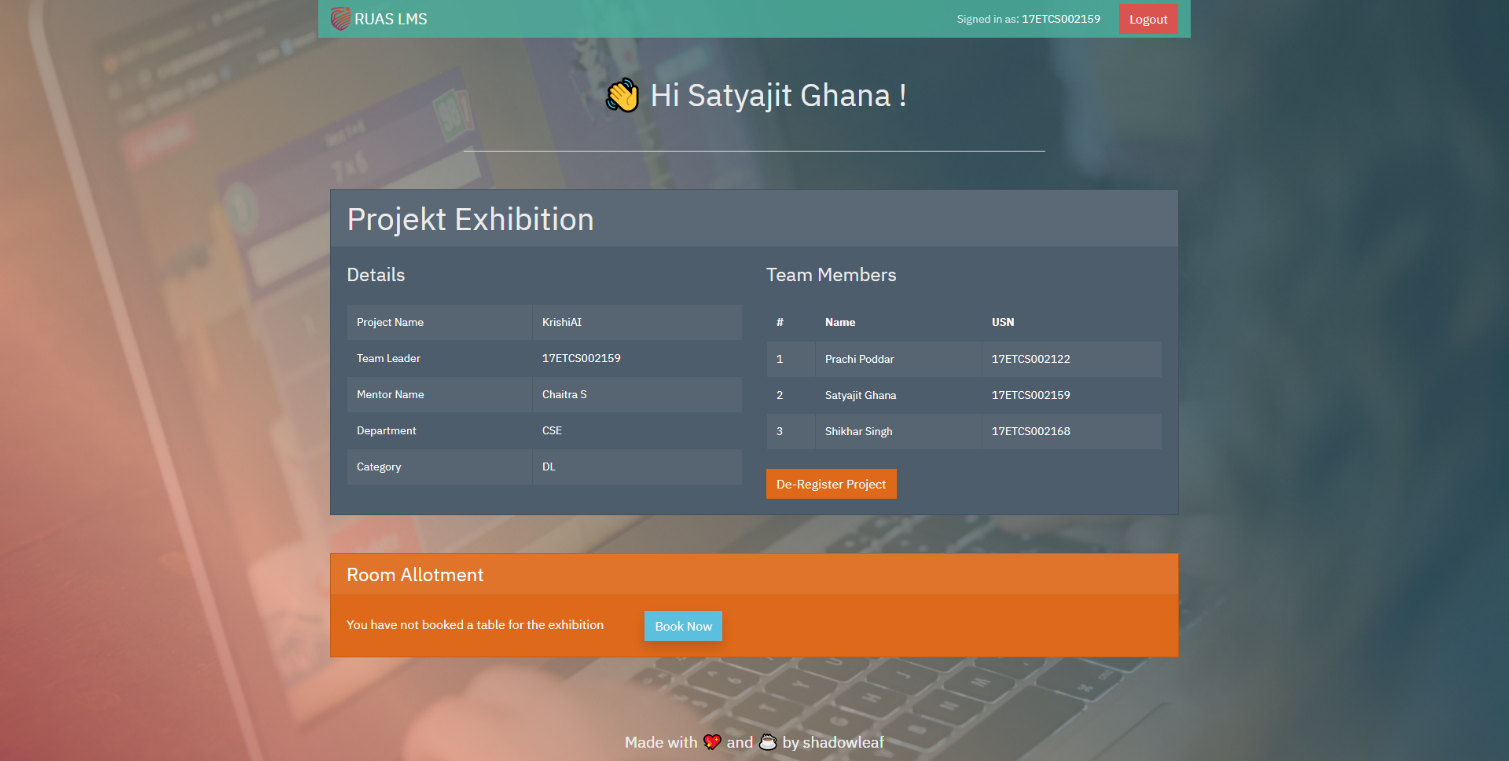


Figure ‑ Student Home Page

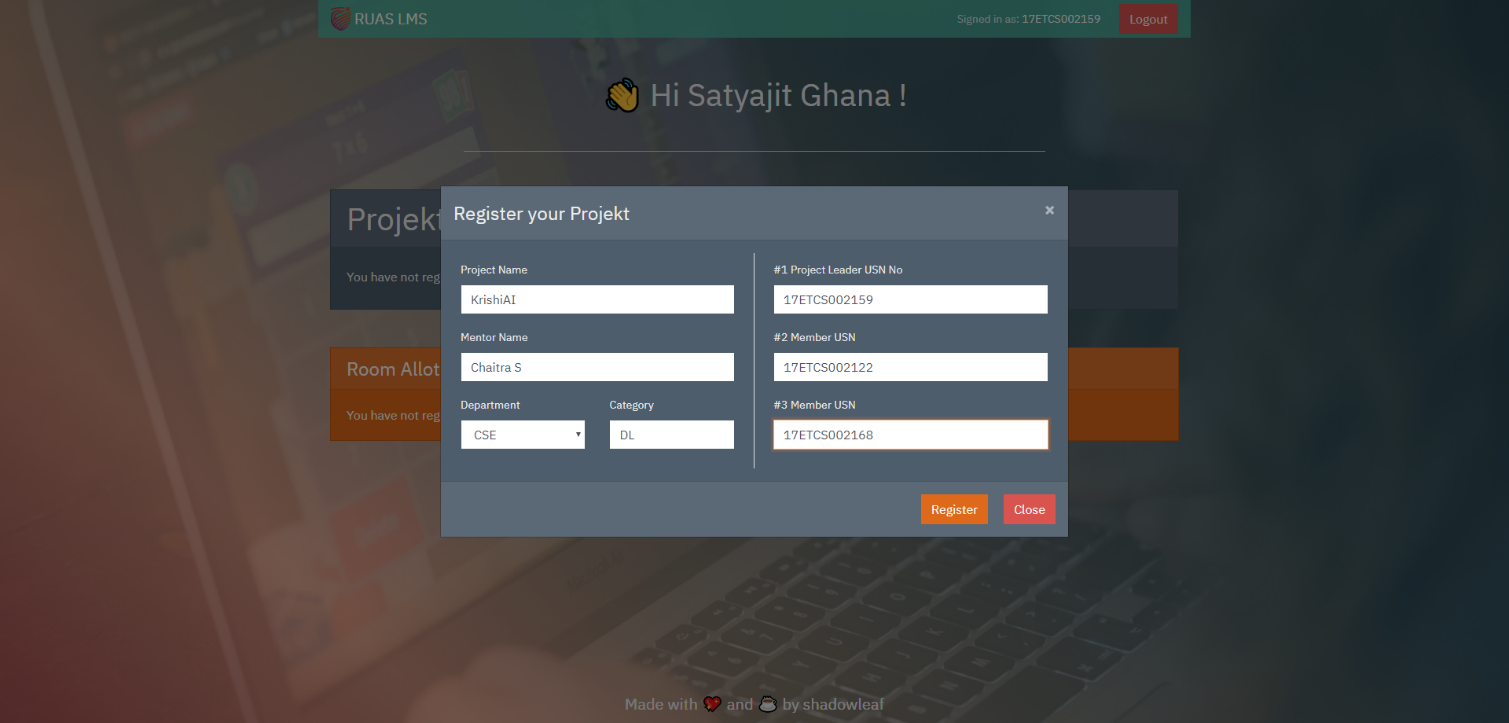


Figure ‑ Student Project Register Page



Figure ‑ Room Allocation of Project

### 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.

### 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**](https://github.com/satyajitghana/ruas-lms)

Deployed Website: [**https://satyajitghana.github.io/ruas-lms/login/**](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***=**{values**.***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***=**{values**.***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**.***reg\_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" *value***=**{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***=**{values**.***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***=**{values**.***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***=**{values**.***project\_name*} *onChange***=**{handleChange} *placeholder***=**"" />

                                            </Form.Group>

                                            <Form.Group>

                                                <Form.Label>Mentor Name</Form.Label>

                                                <Form.Control *type***=**"text" *name***=**"mentor\_name" *value***=**{values**.***mentor\_name*} *onChange***=**{handleChange} *placeholder***=**"" />

                                            </Form.Group>

                                            <Row>

                                                <Col>

                                                    <Form.Group>

                                                        <Form.Label>Department</Form.Label>

                                                        <Form.Control *as***=**"select" *name***=**"department" *value***=**{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" *value***=**{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**.***member1*} *onChange***=**{handleChange} *placeholder***=**"" />

                                            </Form.Group>

                                            <Form.Group>

                                                <Form.Label>#3 Member USN</Form.Label>

                                                <Form.Control *type***=**"text" *name***=**"member2" *value***=**{values**.***member2*} *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({ showRegisterModal: 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 <Button *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*}</strong> 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 *variant***=**"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**](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' }, jwtconfig**.**JWT\_ACCESS\_TOKEN);

*// fetch the user details from the db*

                connection**.**query('SELECT \* FROM STUDENT WHERE id = ? LIMIT 1', user**.***id*, (**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**;

                    }

*// 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, contact\_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', 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' });

**return**;

        }

        const user **=** results[0];

**if** (user**.***hashed\_password* **===** hashed\_password) {

            const access\_token **=** jwt**.**sign({ username: user**.***user\_name*, role: 'staff' }, jwtconfig**.**JWT\_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, department, category) VALUE (?)', [[project\_leader\_regno, project\_name, mentor\_name, department, category]], (**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) VALUES ?", [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, (**error**, **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 = ?', std\_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\_REGISTER p JOIN STUDENT c ON c.reg\_no = p.student\_reg\_no WHERE p.project\_id = ?', project**.***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: '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 EXHIBITION 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 FROM 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**.***params***.***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 FROM PROJECT\_EXHIBITION a WHERE (SELECT b.table\_no FROM PROJECT\_EXHIBITION b WHERE b.table\_no = a.table\_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*}, ${req**.***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**, **result**, **fields**) **=>** {

**if** (error) {

            console**.**error(error);

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

**return**;

        }

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

    });

});

module**.**exports **=** router;