Sri Lankan Institute of Information Technology

Application Framework - SE3040

Year 3, Semester 1 (2020) - Final Examination

Category – "Education"

IT18078510 – S.P. Vidhanaarachchi

Table of Contents

Introduction

The web application named as “iLearn” is a self-learning online video sharing platform which allows students to download and watch lecture videos online. Through this e-learning website student can register and go through video tutorials which are uploaded by the lecturer. With the current pandemic of COVID-19 people tend to stay at home. The COVID-19 epidemic has effected the education system all around the world. Students are unable to attend the lectures because all the education institutes are closed. iLearn helps the students to learn and engage in studies.

According to the course modules, a lecturer can log into the system and add the lecture tutorials with the corresponding module, lecture number, a description, name of the lecturer in charge and the video source. Those added lectures can be viewed and downloaded by the students. Lecturers can upload the most relevant videos so that the students have less difficulties in self-learning.

This system has three main functionalities which are student management, course (lecture) management and sign in-sing out. The students can either view the video tutorials or download it. But, in order to view the video tutorials and their relevant lecture descriptions, a student needs to be a registered user. For that students needs to sign up for a new account on the system. After creating a new account, the user can use his/her credentials to log into the system and access the privileges. Students only have the privilege to access the video tutorials in the course web. A lecturer has the authority of uploading the lectures, edit its details and delete if necessary. Lecturers can also view the registered students to the web application. If in case, they can use the administrator privileges to remove a student.

This web application was developed under the following technologies:

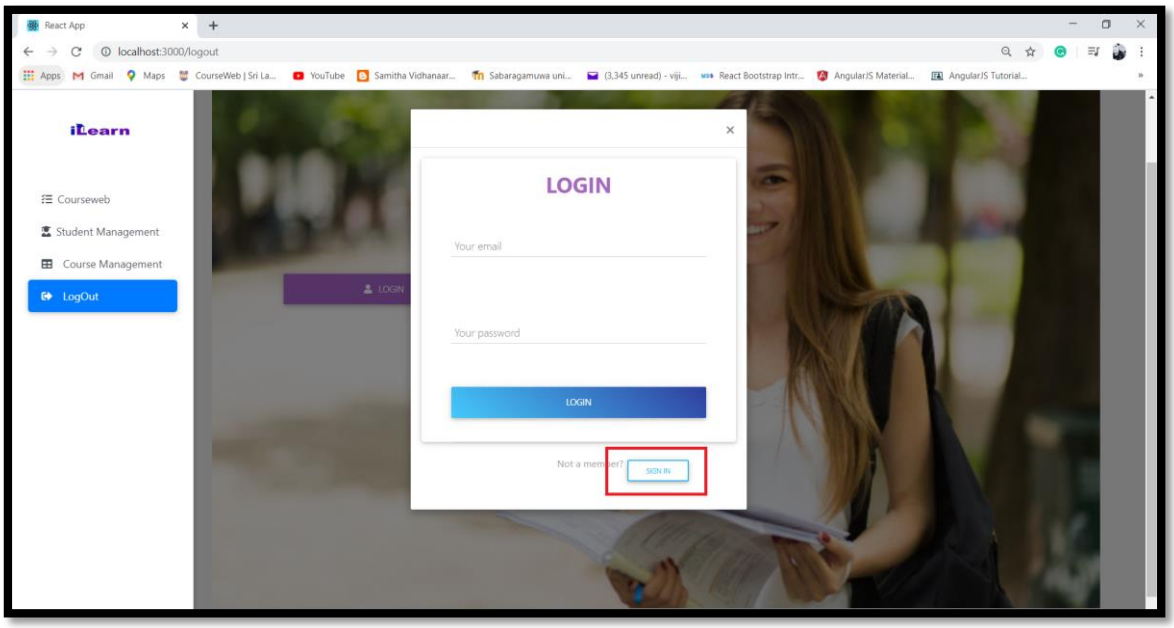
- ReactJs
- NodeJs
- ExpressJs
- MongoDB
- RESTful Web Services

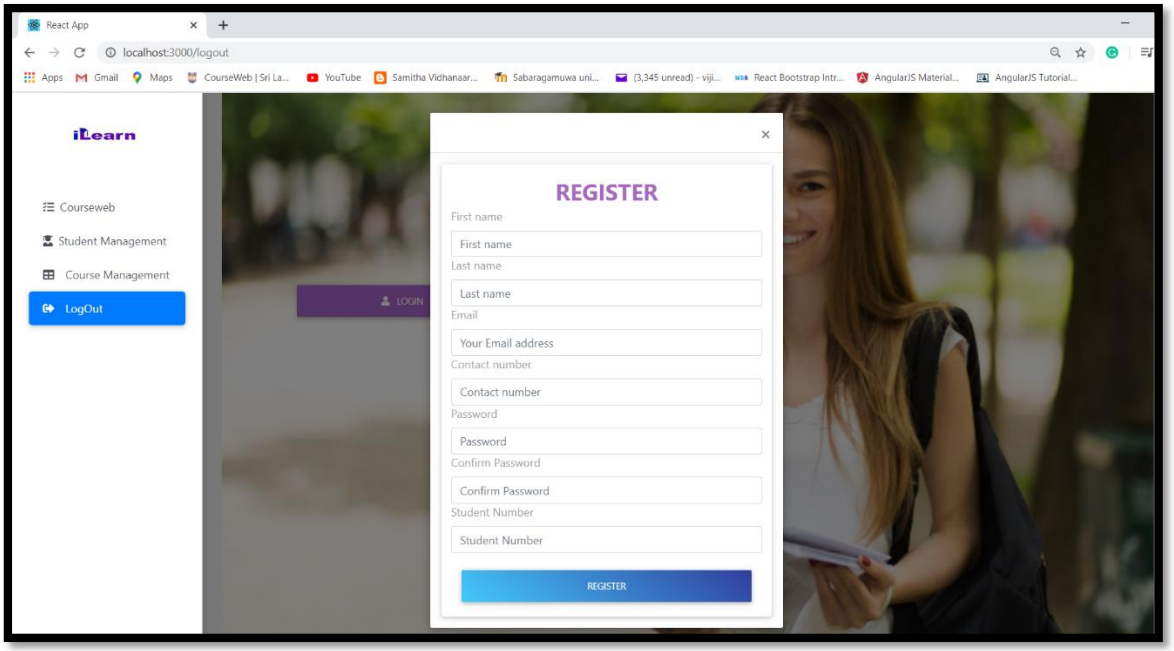
The database used was MongoDB, the frontend was implemented using ReactJs and the backend was implemented using NodeJs, ExpressJs with the use of RESTful Web Services

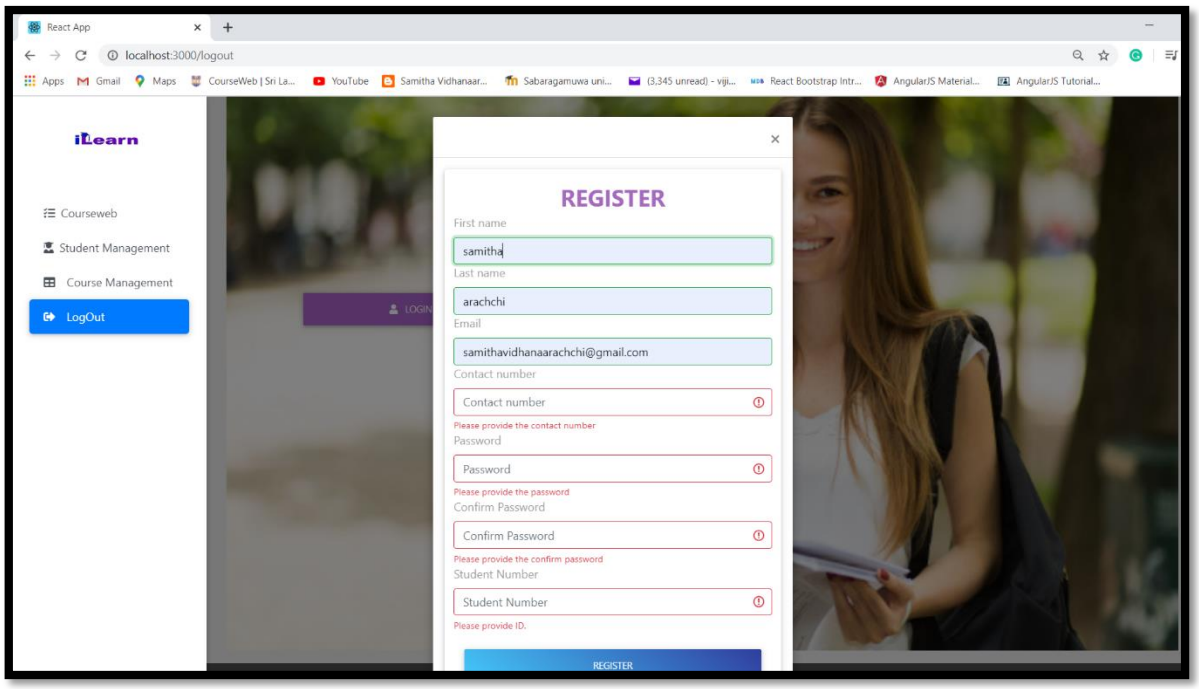
Functionalities

Student Management

Students can register and create an account. Students can register by clicking the “sign up” button. The register form will be displayed as a model making the web application more user friendly and attractive.

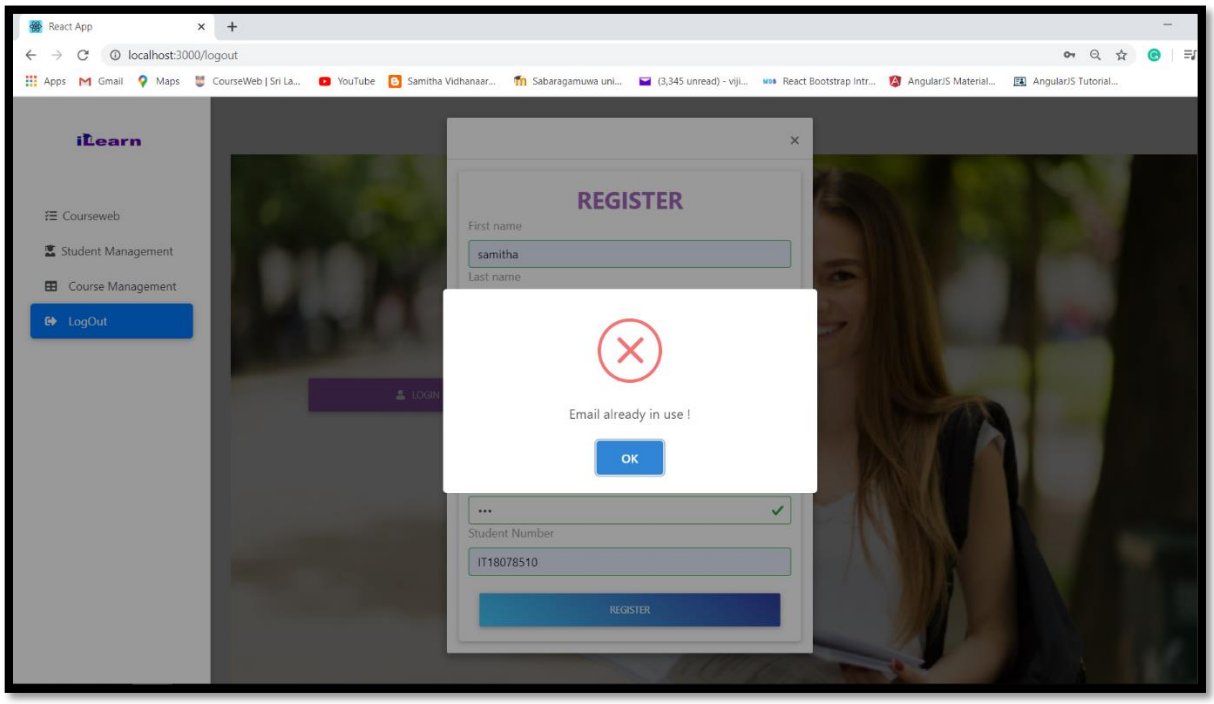




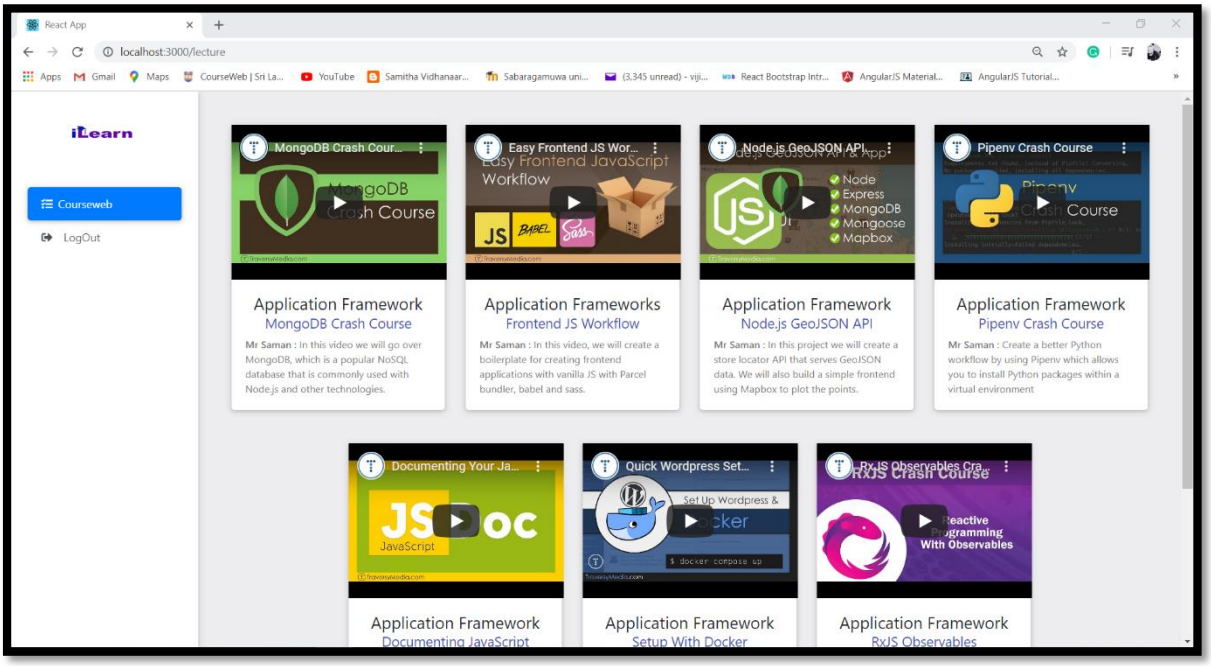


All the fields are validated; therefore, a student has to enter all the necessary details in order to register him/herself.

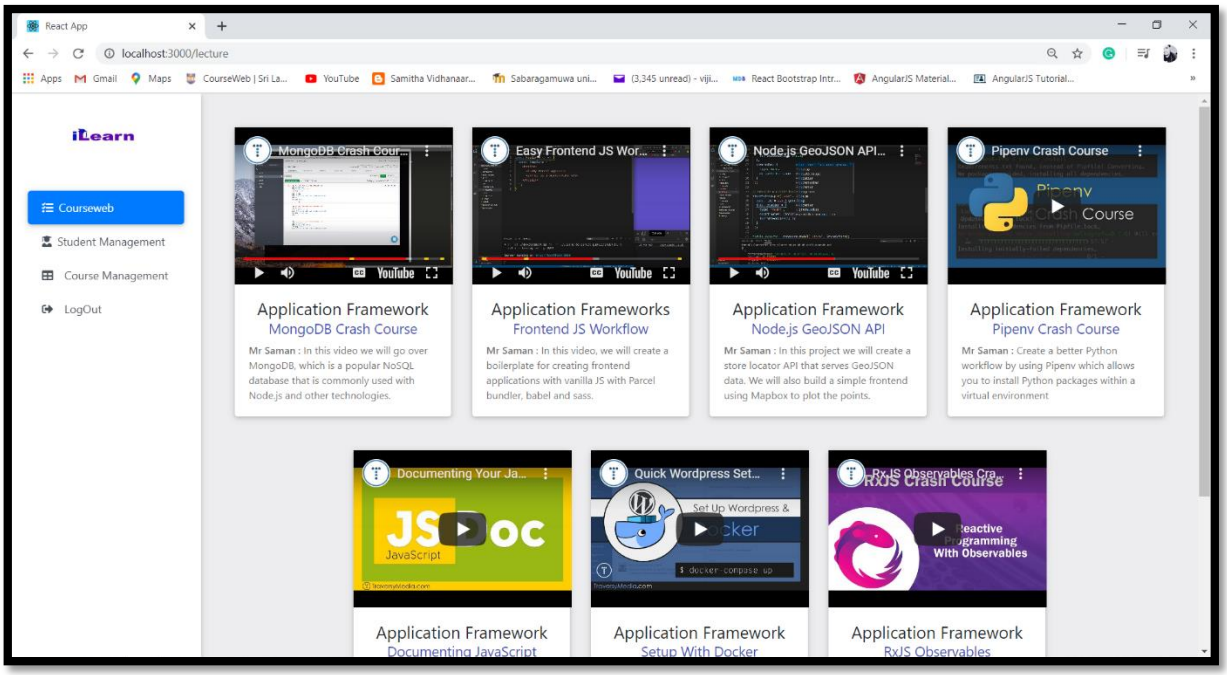
Also a student cannot register twice since the system is validated to check whether the student is already a registered user.

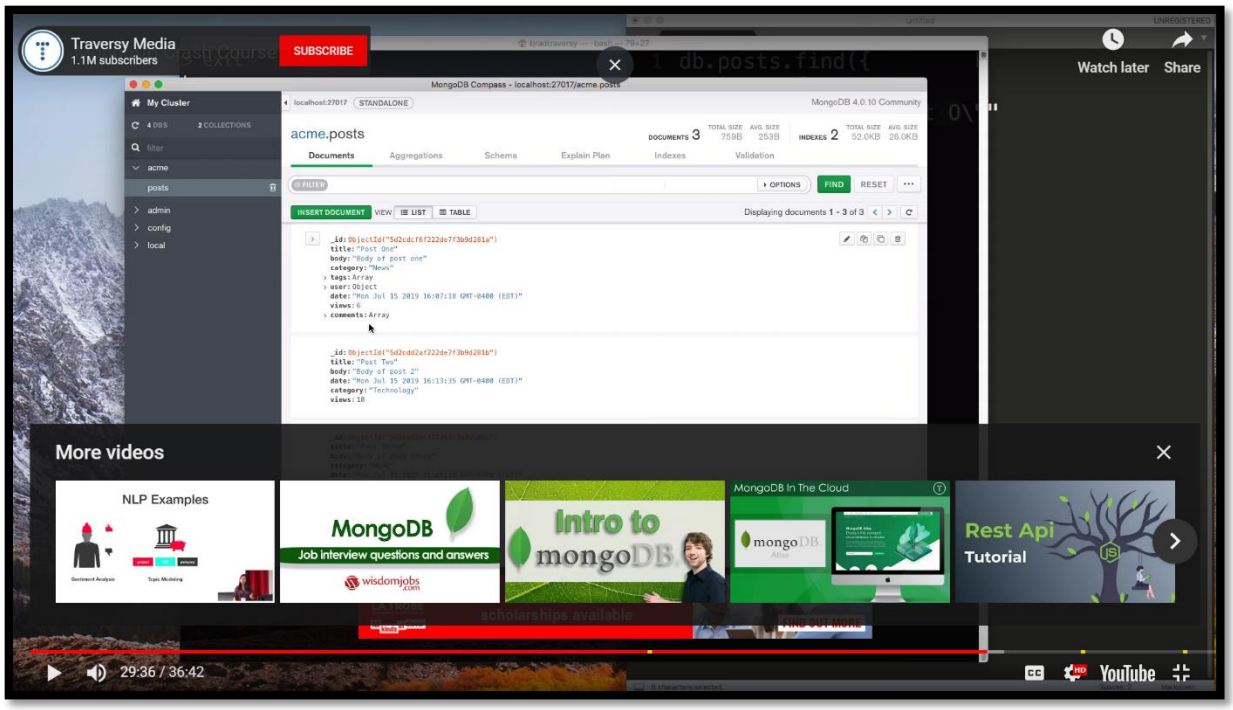


The student has the privilege to view the uploaded lecture tutorials in the course web. The admin accessible pages are restricted to students hence only the course web page is visible.

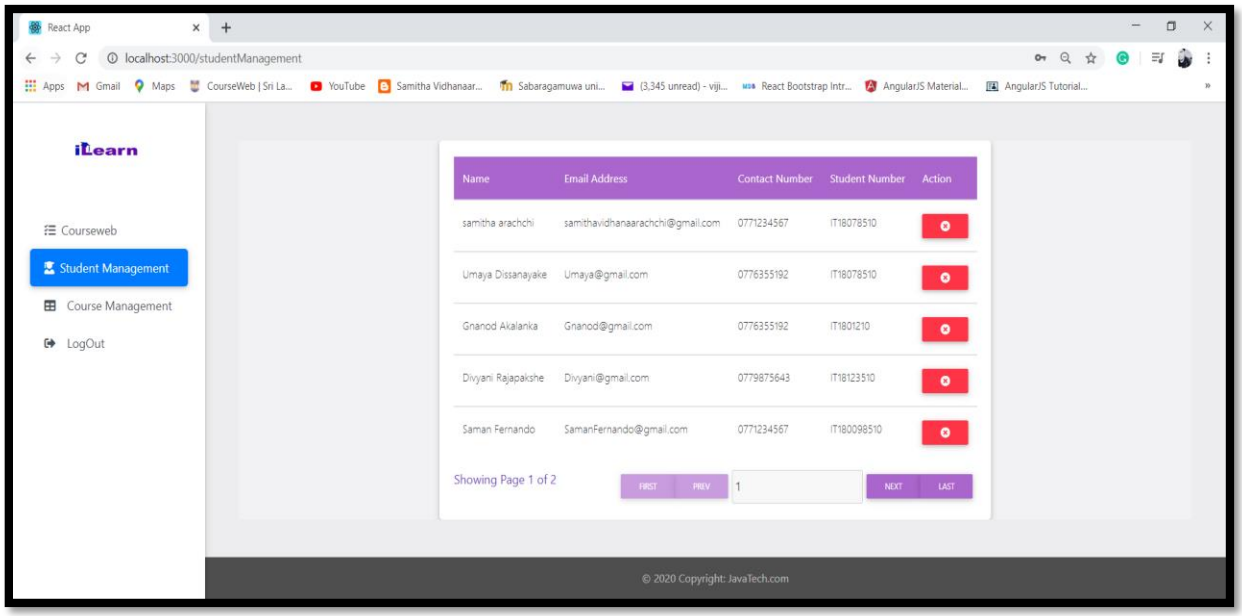


The videos can be viewed in the page itself. Any amount of videos can be played at once. If the student needs he/she can view it as full screen as well.

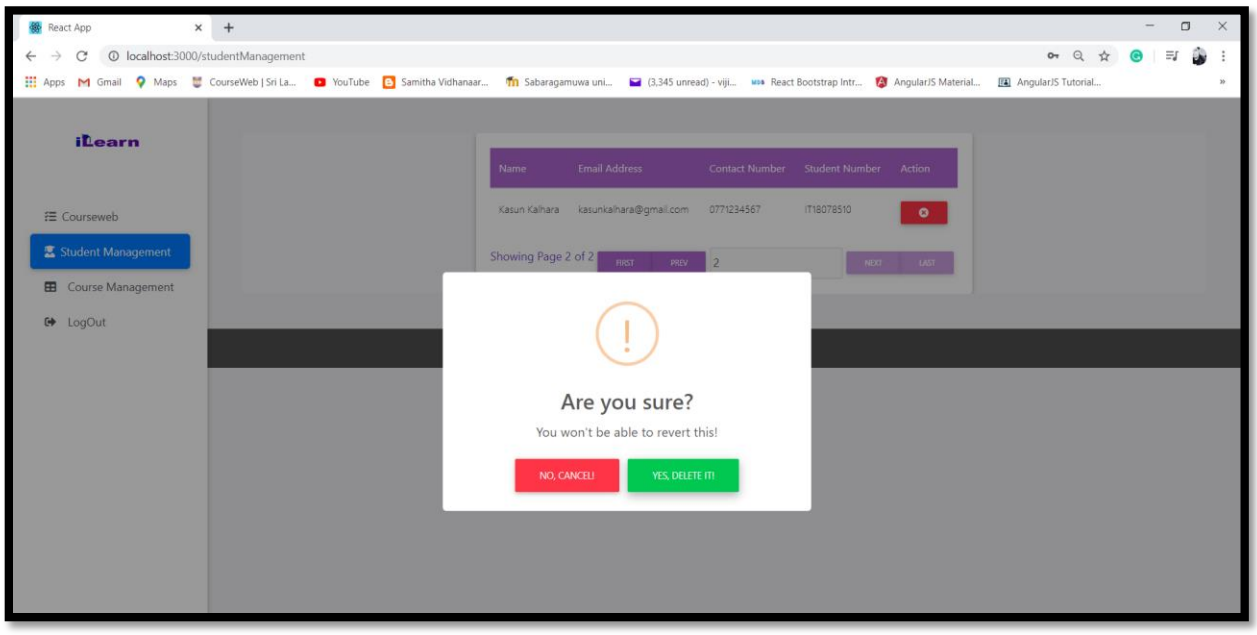




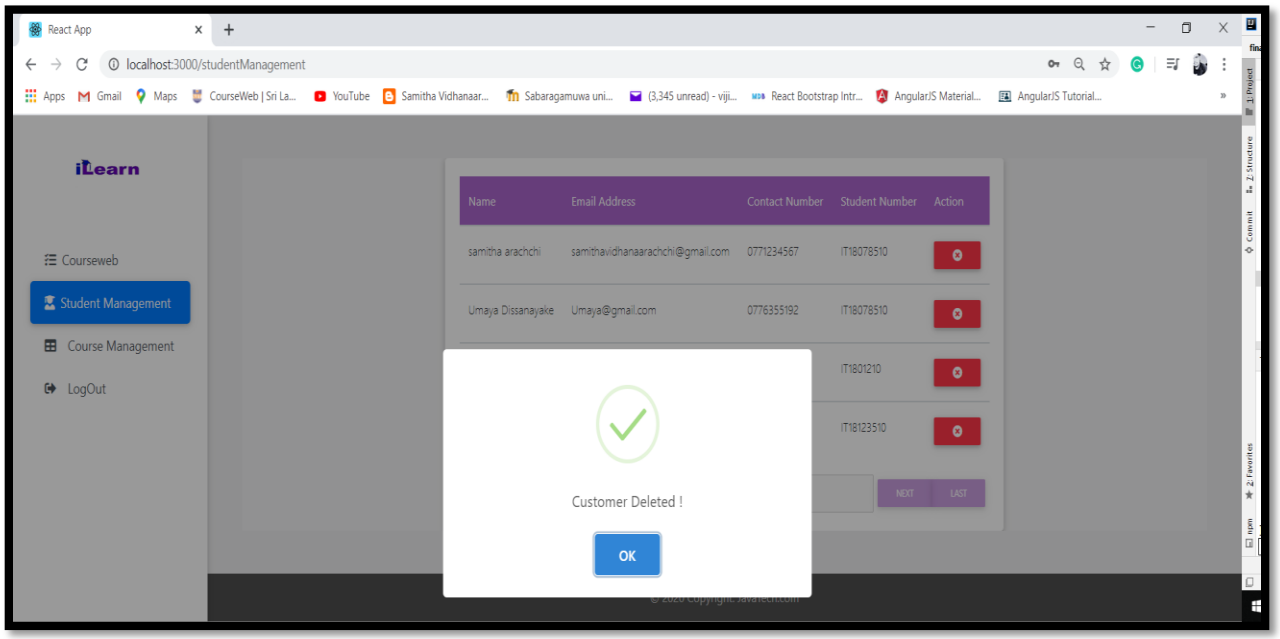
The lecturers have the privilege to view the registered students. Their name, email address, contact numbers and student ID is displayed The table content is organized with a pagination in order to organize lots of records as well as to increase user friendliness.



The lecturers can delete students if they want to. A confirmation message is initially displayed asking to confirm the decision.

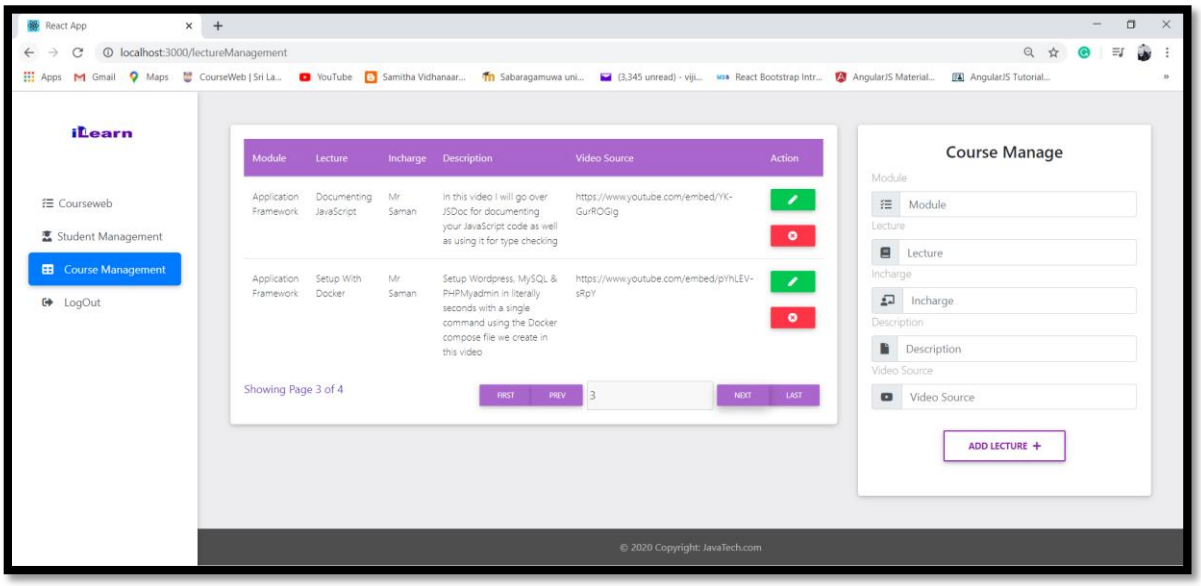


Upon confirming the student account will be deleted.



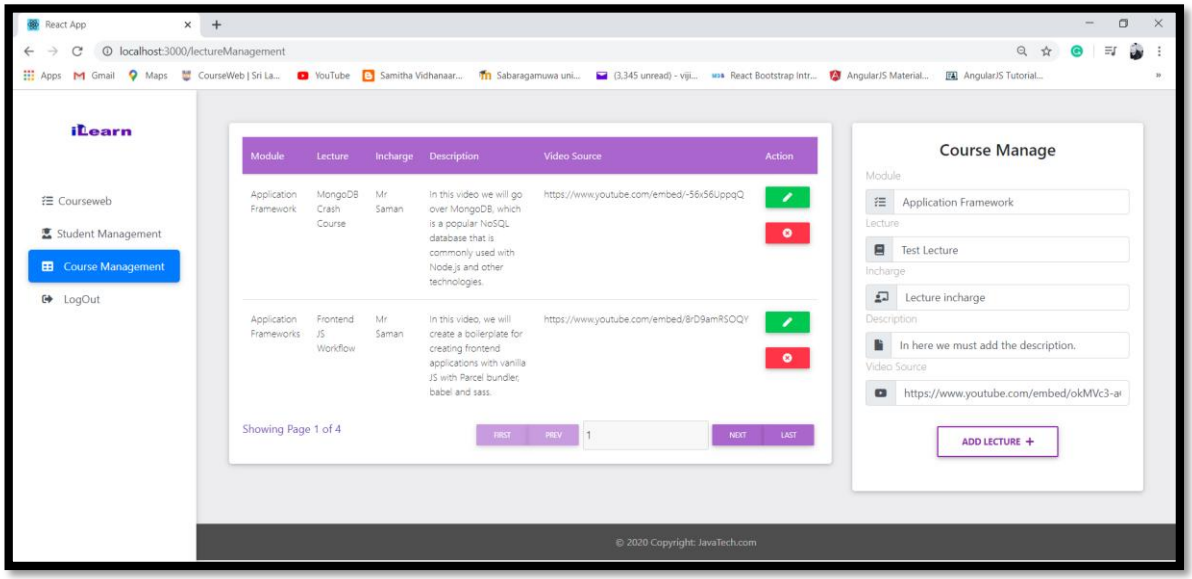
Course/Lecture Management

Only lecturers have the privilege to manage the courses. The navigation bars for user management and course management is only visible for the lecturers. In the course management window, a lecturer can add, edit delete lecture tutorials for the students to see.

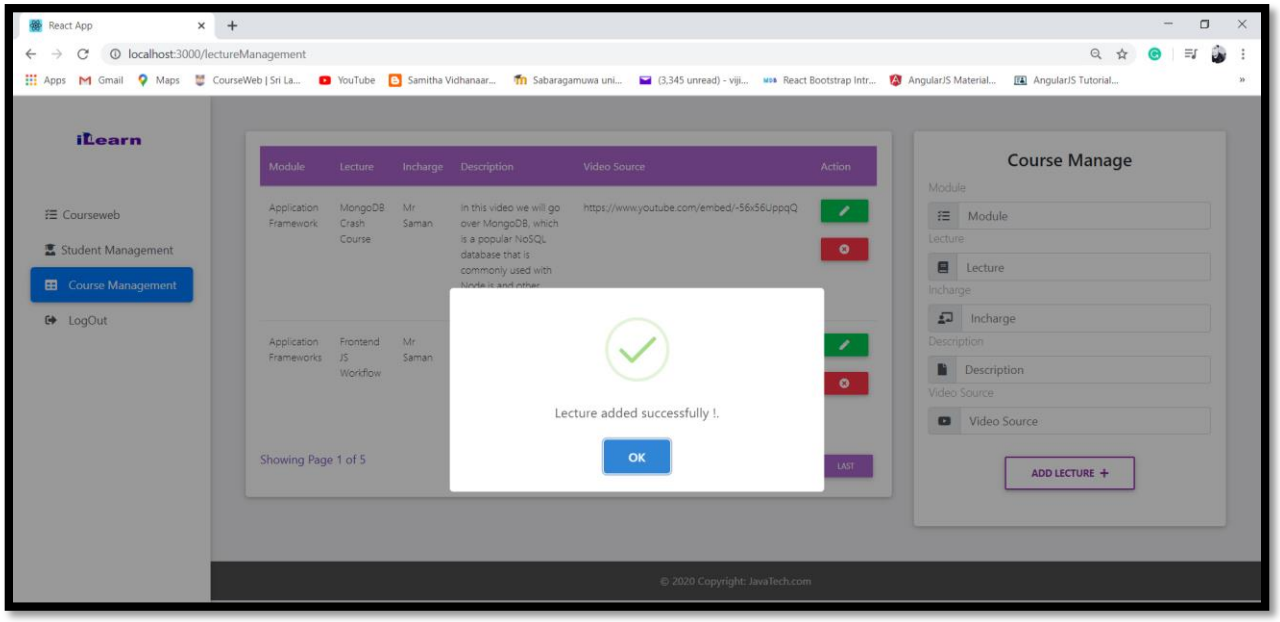


Besides in the home page, the lecturers can view all the lecture tutorials uploaded in this window as a table form which is organized by a pagination.

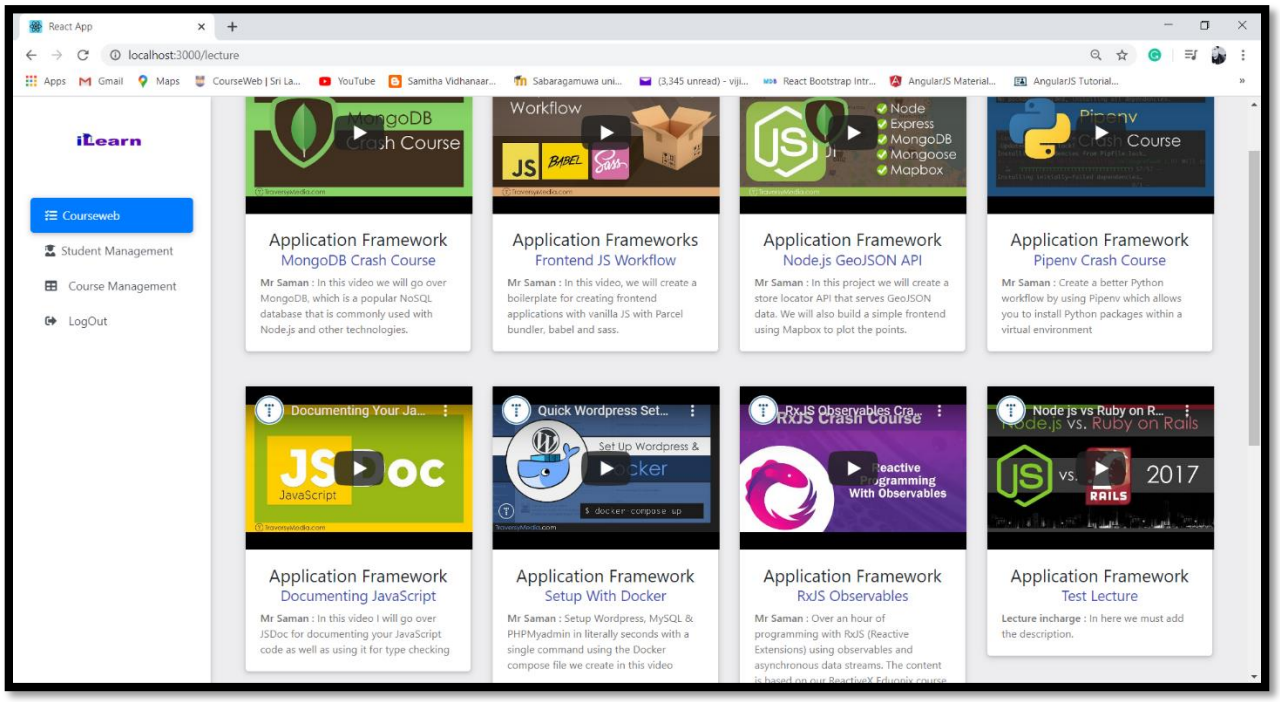
Lecturers can add the tutorials with its module name, lecture name/ number, description, lecture in charge and the video source from the form. All the fields have been validated.



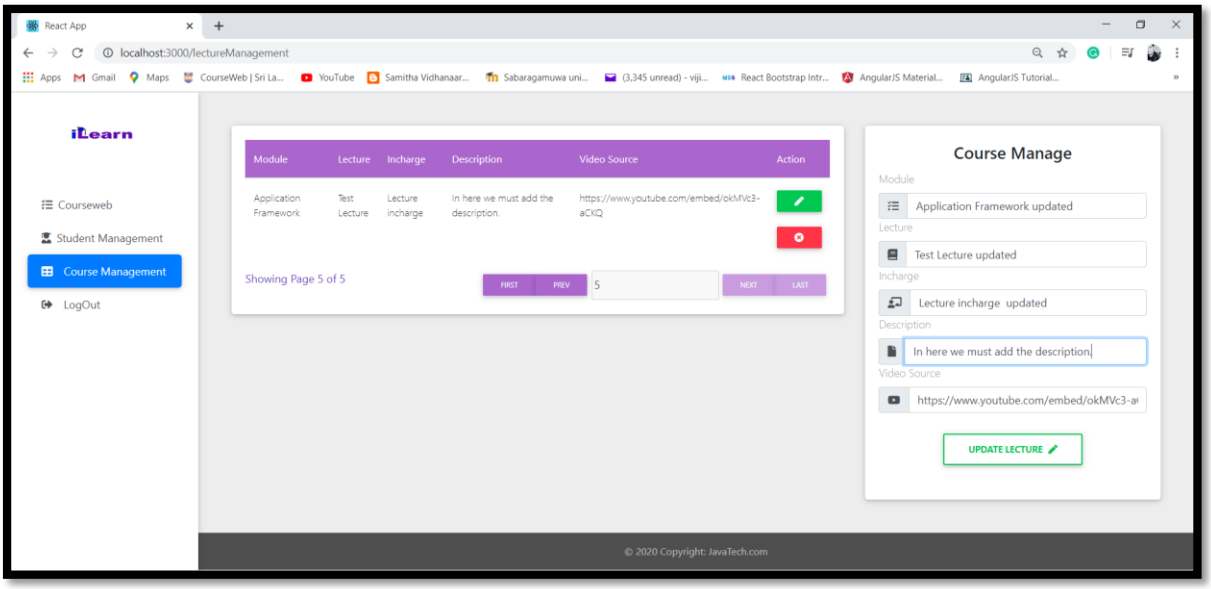
After adding a tutorial a success message is displayed



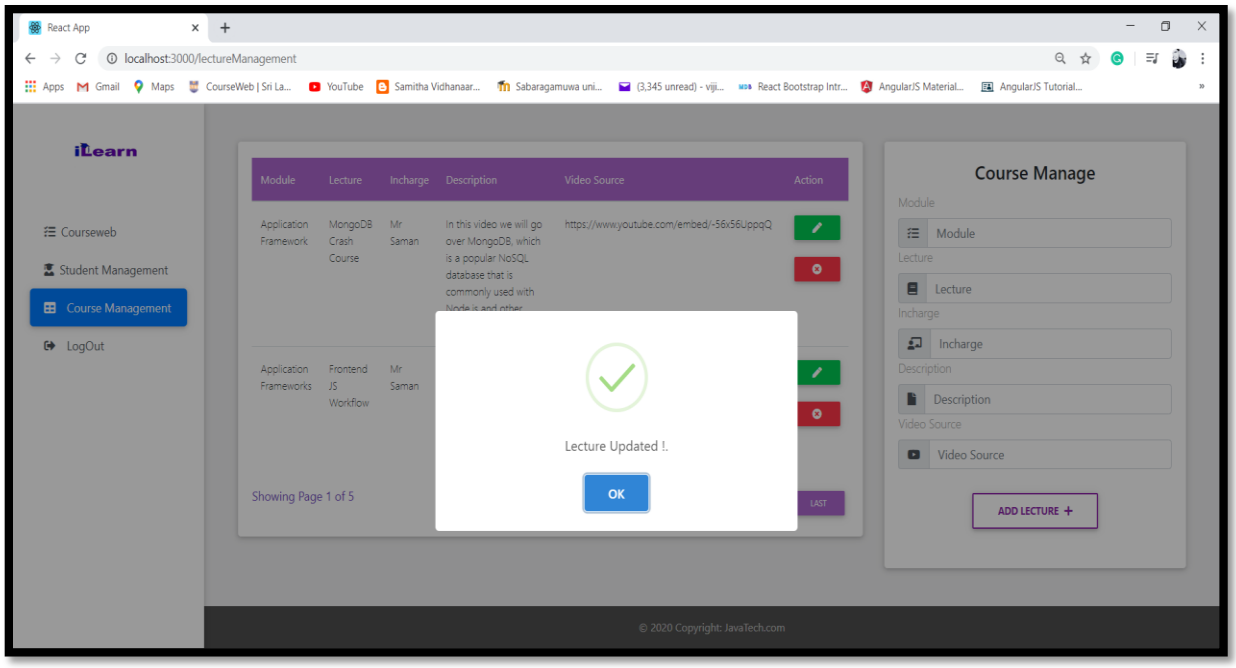
The added Tutorial is now visible in the course web. As shown, the Application Framework Test lecture can be now seen in the homepage.



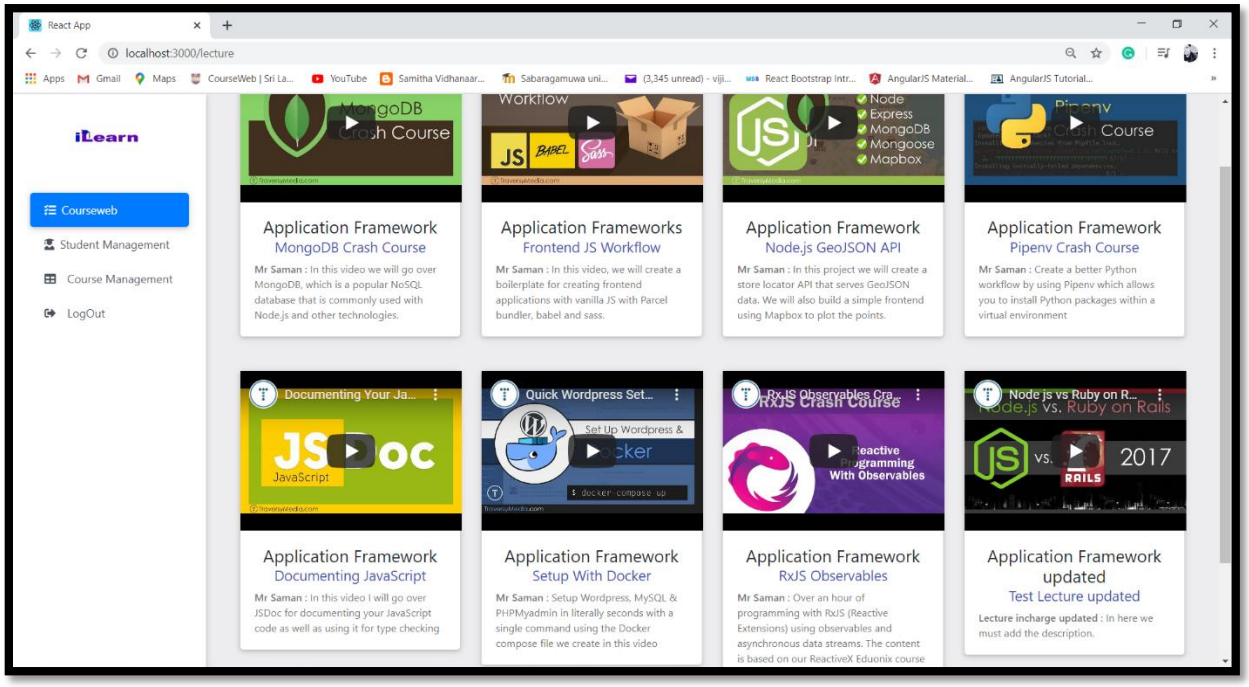
The lecturers can also update the details of the existing tutorials if they need to. For that they have to click the edit button which is available for each record in the table as shown below. By clicking the edit button the details will get loaded to the form and the “add” button will be changed to “Update”.



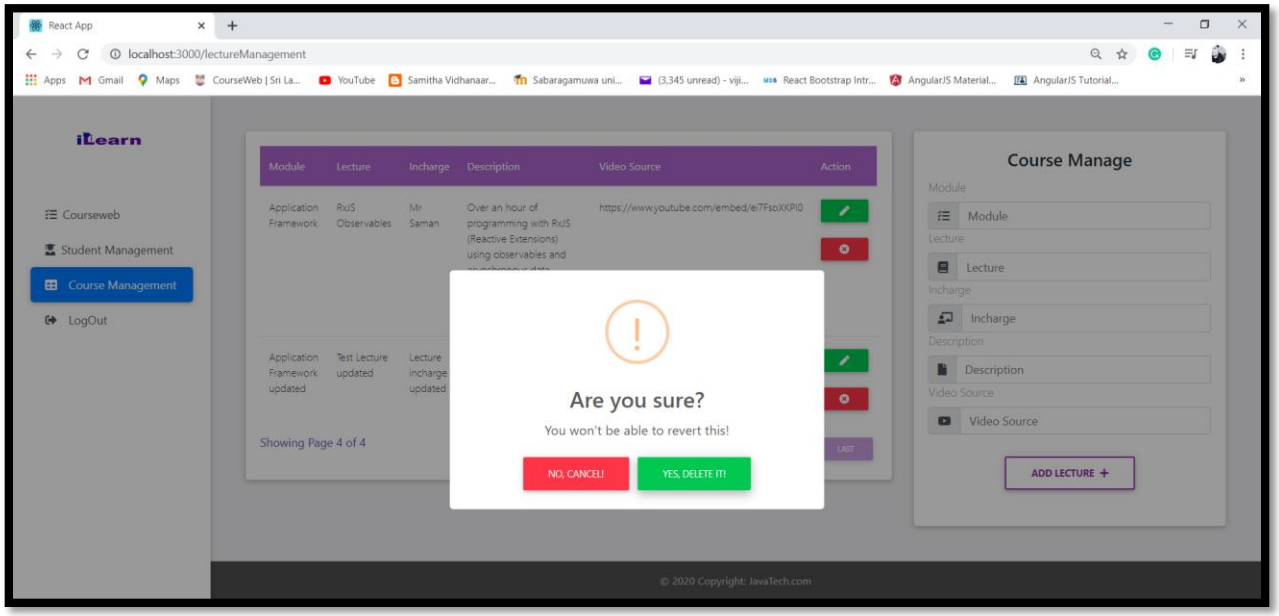
After editing the necessary details, the lecture can update the record. After updating a success message is displayed.



The updated tutorial can be now seen in the homepage with the changed details.

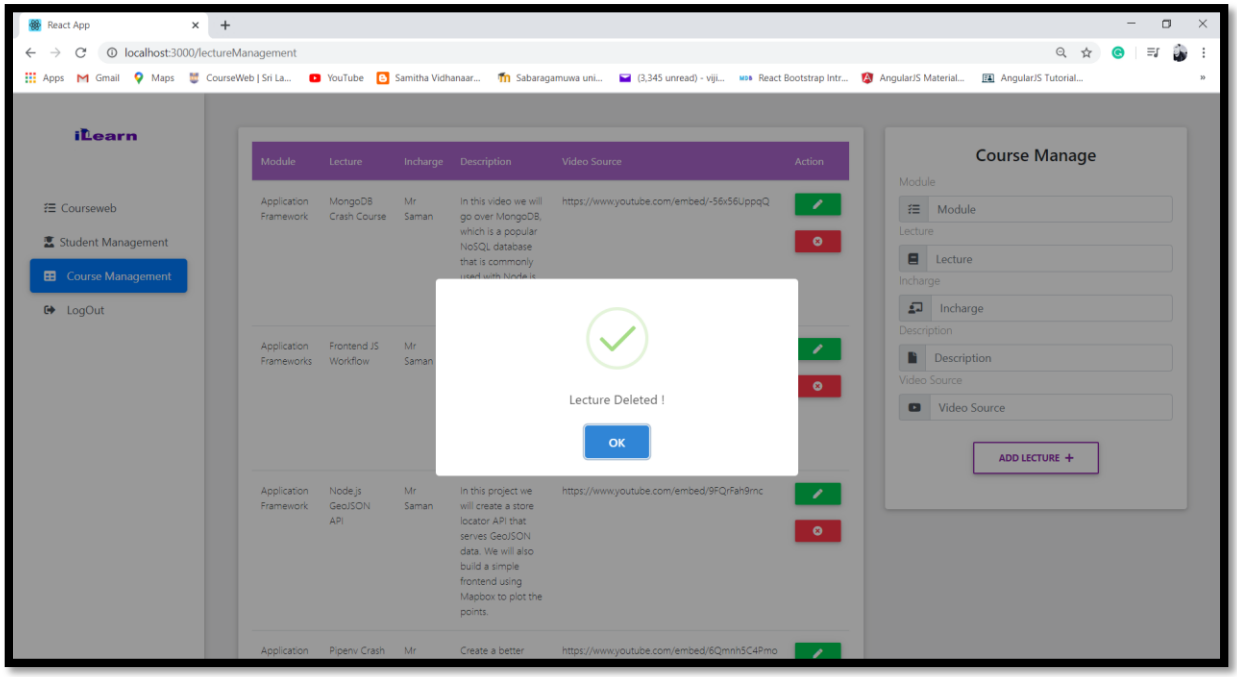


Finally, the lecturers can also delete the tutorials if they need to. For that they have to click the delete button of the relative record.

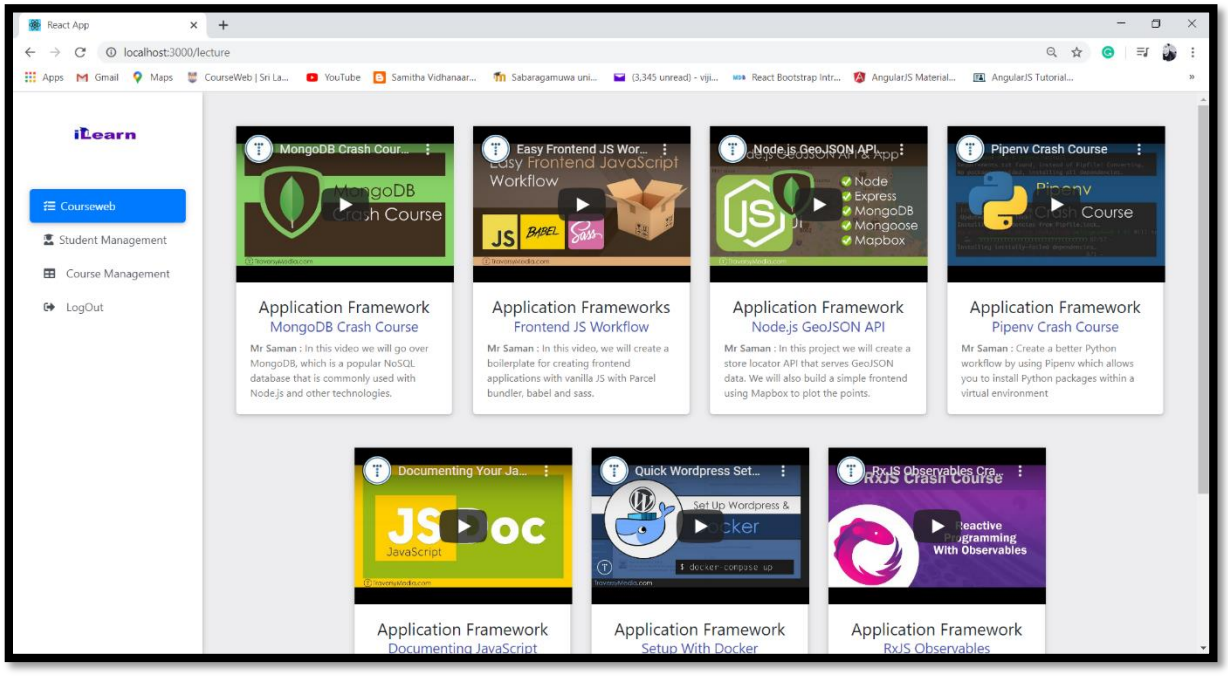


A confirmation message is display to verify the decision.

Upon verifying the record will be deleted and a success message is displayed.



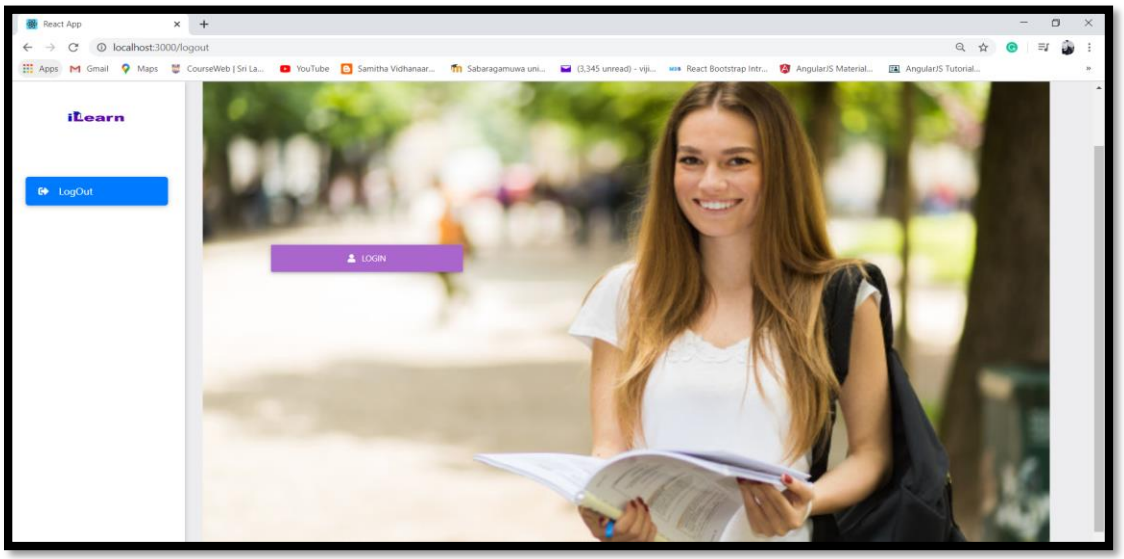
The deleted tutorial is no longer available in the homepage as shown below.



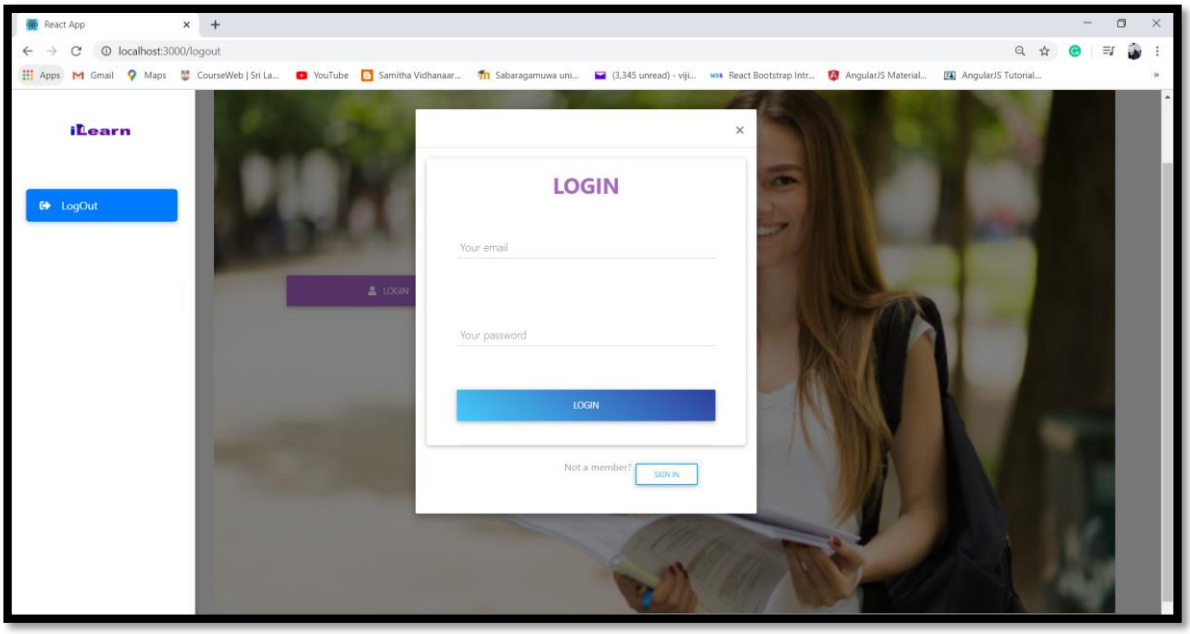
sign in-sing out

As stated above a new user to the system can register for a new account by filling out the user details given in the registration form. Once the user details are filled up and validated, the user is provided with a user account.

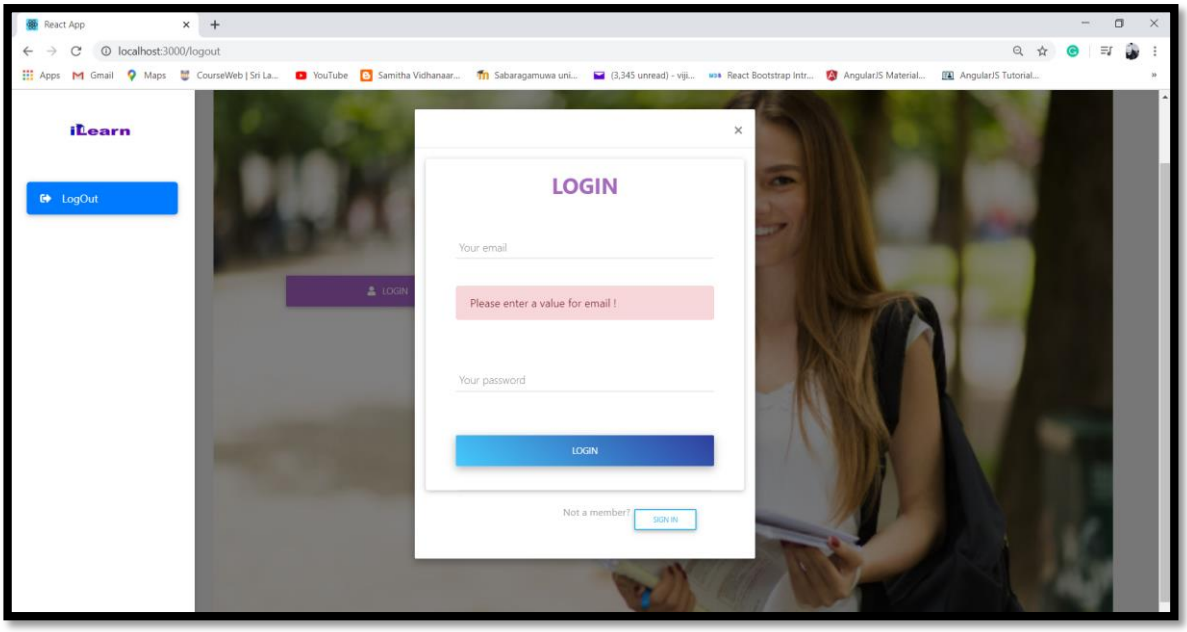
After entering the user credentials in the login form; username and password, the user can log into the system.



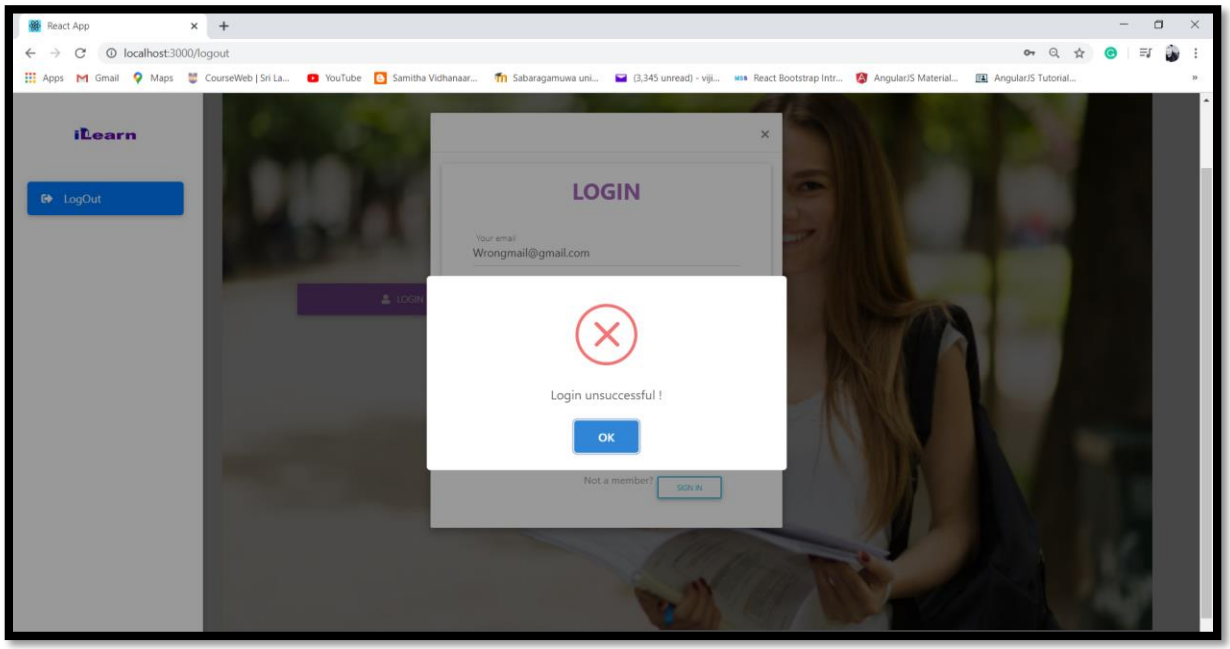
The forms are displayed as models where they appear from above making the web application more attractive



The fields are correctly validated.

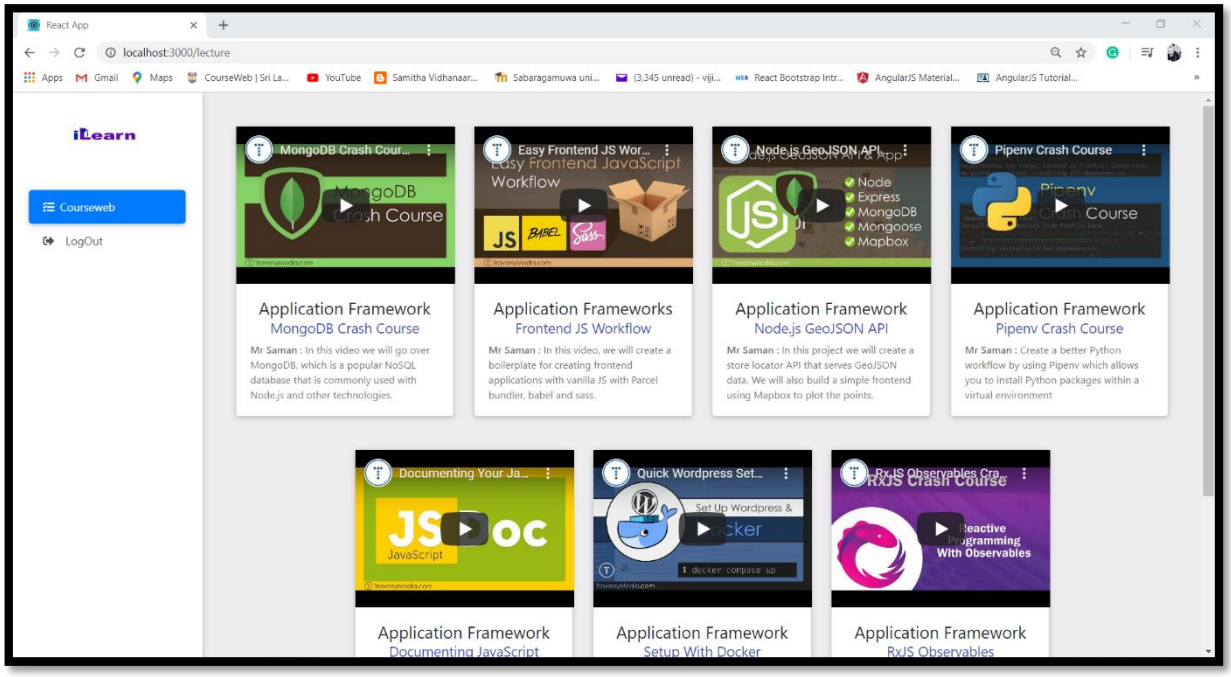


If the user enters wrong credentials, an error message is displayed.

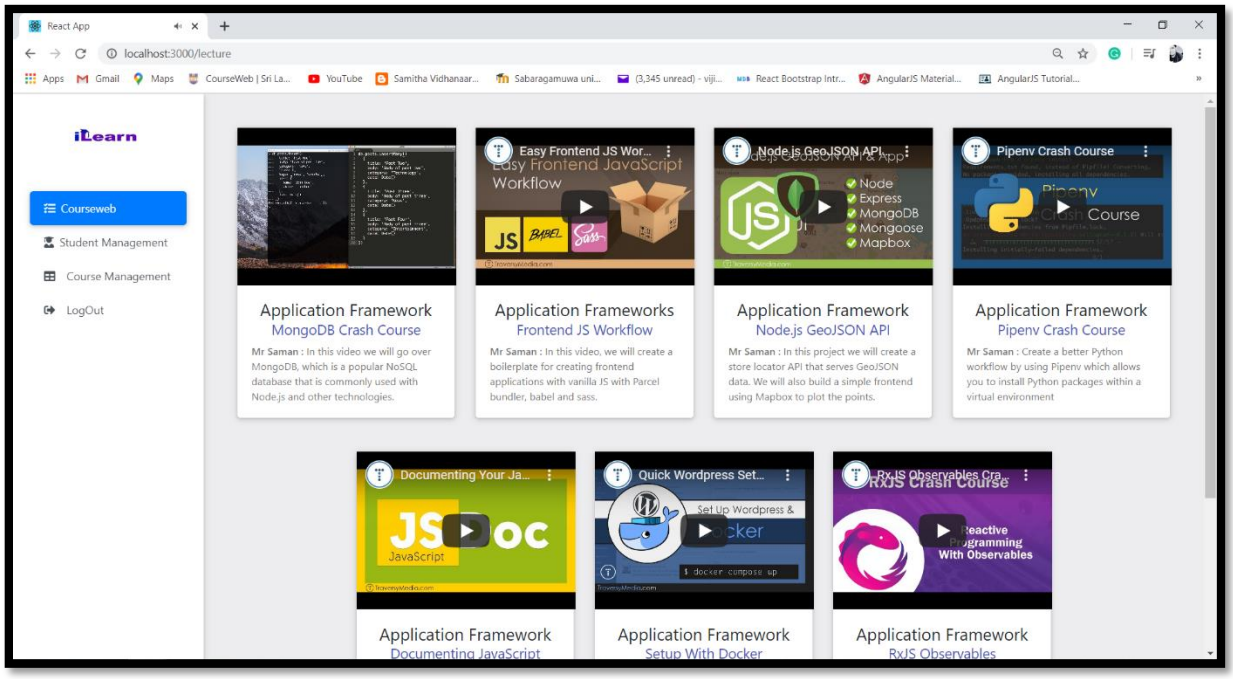


If the user enters valid credentials they are directed to their relative pages. Students are directed to a portal where only the video tutorials can be seen while lectures have the privilege to access both student management and course management windows. Finally, a logged user can logout after visiting the iLearn web application

- For a student



- For a lecturer



RESTful Web Service Implementation

1. Adding a New User – POST Method

```
router.route('/add').post(function (req,res) {
    let userDetail = new userdetails(req.body);
    console.log("-----------------------------------");
    console.log(req.body);
    console.log("-----------------------------------");
    userdetails.findOne({ email: req.body.email  },)
        .exec()
        .then(userValid =>{
            if( userValid ){
                res.status(200).json({'userDetail': "userAvailable"});
            }else{
                console.log("**************success");

                userDetail.save().then(users => {
                    if(users){
                        res.status(200).json({'userDetail':'successful'});
                    }else{
                        res.status(400).send( data: 'failed');
                    }
                }).catch(err=>{
                    res.status(500).json(err);
                })
                res.status(200).json({'userDetail':'successful'});
            }
        }).catch(err=>{
        res.status(500).json(err);
    })
```

2. . View all User Details by User– GET Method

```
router.route('/getAllusers').get(function (req,res) {
    console.log("getDetails --- usermanage")
    userdetails.find().exec().then(item => {
        console.log(item)
        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});
```

3. Validate User Details by- GET Method

```
router.get("/validateUser/:email/:password",function (req,res) {
    let email = req.params.email;
    let password = req.params.password;
    userdetails.findOne({ email: email, password: password },)
        .exec()
        .then(userValid =>{
            console.log("User Valid");
            console.log(userValid);
            console.log("User Valid");
            if( userValid ){
                res.status(200).json({"Message": userValid});
            }else{
                res.status(200).json({"Message": "unsuccessful"});
            }
        }).catch(err=>{
        res.status(500).json(err);
    })
});
```

4. View single user details by – GET Method

```
//
router.route('/getDetailuser/:id').get(function (req,res) {
    console.log("getDetailsof the user-----------------------------------------------");
    let id = req.params.id;

    userdetails.find({ _id : id }).exec().then(item => {
        console.log(item.email)
        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});
```

5. Delete user Details by – GET Method

```
//
router.route('/deleteUser/:id').get(function (req, res) {
    let id=req.params.id;
    console.log("Delete Called!");
    userdetails.deleteOne({_id:id}).then(sup=>{
        console.log("successful");
        res.status(200).json({'userDelete':'successful'});
    }).catch(err=>{
        console.log("fail");
        res.status(400).send( data: 'fail');
    });
});
```

6. Add course details by –POST Method

```
router.route('/add').post(function (req,res) {
    let adminDetail = new AdminDetail(req.body);
    adminDetail.save()
        .then(sup=>{
            res.status(200).json({'AdminDetail':'successful'});
        }).catch(err=>{
        res.status(400).send( data: 'fail');
    });
});
```

7. View all course details by – GET Method

```
router.route('/getAlldetail').get(function (req,res) {

    AdminDetail.find().exec().then(item => {

        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});
```

8. Delete Course details by – GET Method

```
router.route('/deleteAdmin/:id').get(function (req, res) {
    let id=req.params.id;
    console.log("Delete Admin Called!");
    AdminDetail.deleteOne({_id:id}).then(sup=>{
        console.log("successful");
        res.status(200).json({'adminDelete':'successful'});
    }).catch(err=>{
        console.log("fail");
        res.status(400).send( data: 'fail');
    });
});
```

9. Update Course details by – GET Method

```
router.route('/updateDetail/:id/:name/:email/:position/:password/:video').get(function (req, res) {
    console.log("update function called")
    let id = req.params.id;
    let name=req.params.name;
    let email=req.params.email;
    let position=req.params.position;
    let password=req.params.password;
    let video=req.params.video;

    AdminDetail.updateOne({_id : id},{$set: {Module:name, Lecture: email, Incharge: position, Description: password, Video: video}}).then(sup=>{
        console.log(" successfully edited");
        console.log(sup);
        res.status(200).json({'adminUpdate':'successful'});
    }).catch(err=>{
        console.log("update fail");
        res.status(400).send( data: 'fail');
    });
});

module.exports = router;
```

Mongo Query for Implemented Collection

1. Query: UpdateOne – Updating the details of a user

```
    AdminDetail.updateOne({_id : id},{$set: {Module:name, Lecture: email, Incharge: position, Description: password, Video: video}}).then(sup=>{
        console.log(" successfully edited");
        console.log(sup);
        res.status(200).json({'adminUpdate':'successful'});
    }).catch(err=>{
        console.log("update fail");
        res.status(400).send( data: 'fail');
    });
});
```

2. . Query: findOne – Finding a user by user email

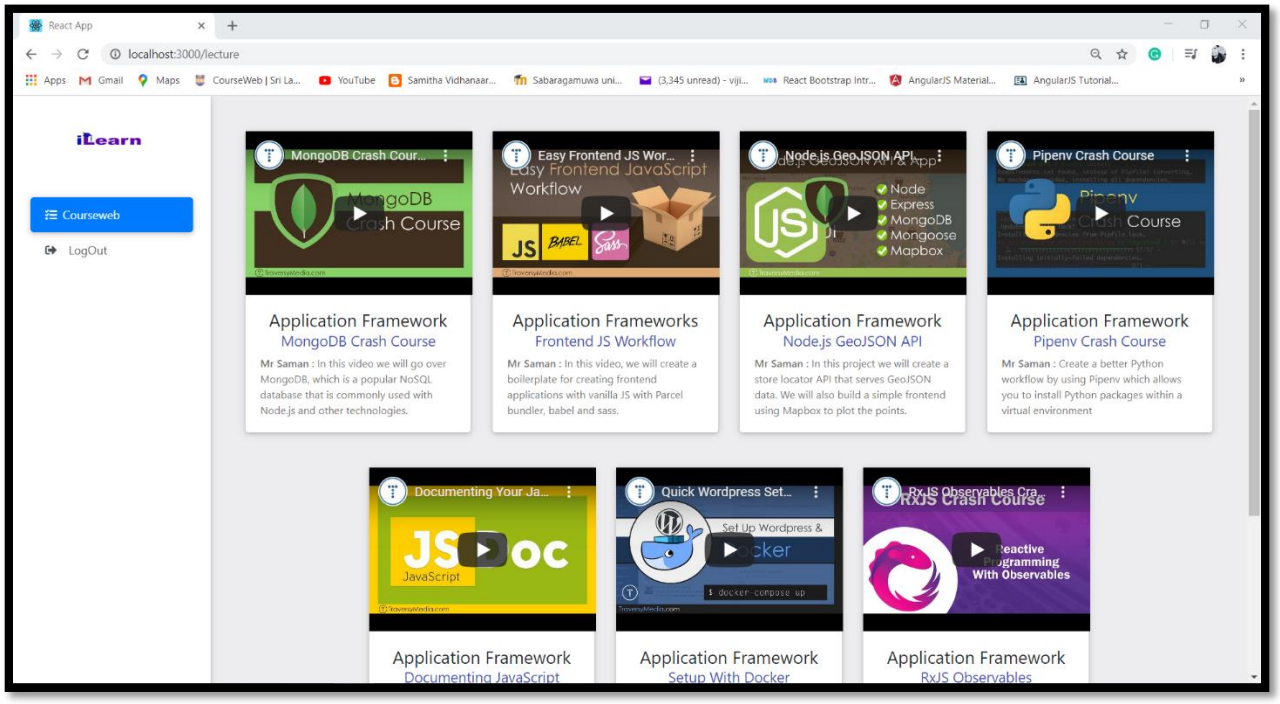
```
userdetails.findOne({ email: req.body.email  },)
    .exec()
    .then(userValid =>{
        if( userValid ){
            res.status(200).json({'userDetail': "userAvailable"});
        }else{
            console.log("**************success");

            userDetail.save().then(users => {
                if(users){
                    res.status(200).json({'userDetail':'successful'});
                }else{
                    res.status(400).send( data: 'failed');
                }
            }).catch(err=>{
                res.status(500).json(err);
            })
            res.status(200).json({'userDetail':'successful'});
        }
    }).catch(err=>{
    res.status(500).json(err);
})
```

3. Query: find – Finding details

```
    userdetails.find().exec().then(item => {
        console.log(item)
        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});
```

Screenshot of the Home Page running on localhost



Implementation of Codes

Front End

Footer.js

```
import React from 'react';
import { MDBFooter, MDBBtn, MDBIcon } from 'mdbreact';

const Footer = () => {
    return (
        <MDBFooter color="grey" className="text-center font-small darken-2">


            <p className="footer-copyright mb-0 py-3 text-center">
                &copy; {new Date().getFullYear()} Copyright: JavaTech.com
            </p>
        </MDBFooter>
    );
}

export default Footer;
```

sideNavigation.js

```
 import React from 'react';
import logo from "../assets/logo.png";
import { MDBListGroup, MDBListGroupItem, MDBIcon } from 'mdbreact';
import { NavLink } from 'react-router-dom';

const TopNavigation = () => {
    return (
        <div className="sidebar-fixed position-fixed">
            <a href="#!" className="logo-wrapper waves-effect">
                <img alt="MDB React Logo" className="img-fluid" src={logo}/>
            </a>
            <MDBListGroup className="list-group-flush">
                {/*<NavLink exact={true} to="/dashboard"
activeClassName="activeClass">*/}
                {/*    <MDBListGroupItem>*/}
                {/*        <MDBIcon icon="chart-pie" className="mr-2"/>*/}
                {/*        Dashboard*/}
                {/*    </MDBListGroupItem>*/}
                {/*</NavLink>*/}
                {/*<NavLink to="/profile" activeClassName="activeClass">*/}
                {/*    <MDBListGroupItem>*/}
                {/*        <MDBIcon icon="user" className="mr-2"/>*/}
                {/*        User Profile*/}
                {/*    </MDBListGroupItem>*/}
```

```
                {/*</NavLink>*/}

                <NavLink to="/lecture" activeClassName="activeClass" >
                    <MDBListGroupItem >
                        <MDBIcon icon="tasks" className="mr-2"  />
                        Courseweb
                    </MDBListGroupItem>
                </NavLink>
                <NavLink to="/studentManagement"
activeClassName="activeClass">
                    <MDBListGroupItem>
                        <MDBIcon icon="user-graduate" className="mr-2" />
                        Student Management
                    </MDBListGroupItem>
                </NavLink>

                <NavLink to="/lectureManagement"
activeClassName="activeClass">
                    <MDBListGroupItem>
                        <MDBIcon icon="table" className="mr-3"/>
                        Course Management
                    </MDBListGroupItem>
                </NavLink>

                <NavLink to="/logout" activeClassName="activeClass">
                    <MDBListGroupItem>
                        <MDBIcon icon="sign-out-alt" className="mr-3"/>
                        LogOut
                    </MDBListGroupItem>
                </NavLink>

            </MDBListGroup>
        </div>
    );
}

export default TopNavigation;
```

topNavigation.js

```
import React, { Component } from 'react';
import { MDBNavbar, MDBNavbarBrand, MDBNavbarNav, MDBNavbarToggler,
MDBCollapse, MDBNavItem, MDBNavLink, MDBIcon } from 'mdbreact';

class TopNavigation extends Component {
    state = {
        collapse: false
    }

    onClick = () => {
        this.setState({
            collapse: !this.state.collapse,
        });
    }
```

```
    toggle = () => {
        this.setState({
            dropdownOpen: !this.state.dropdownOpen
        });
    }

    render() {
        return (
            <MDBNavbar className="flexible-navbar" light expand="md"
scrolling>
                <MDBNavbarBrand href="/">
                    <strong>MDB</strong>
                </MDBNavbarBrand>
                <MDBNavbarToggler onClick = { this.onClick } />
                <MDBCollapse isOpen = { this.state.collapse } navbar>
                    <MDBNavbarNav left>
                        <MDBNavItem active>
                            <MDBNavLink to="#">Home</MDBNavLink>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a rel="noopener noreferrer" className="nav-link
Ripple-parent" href="https://mdbootstrap.com/docs/react/"
target="_blank">About MDB</a>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a rel="noopener noreferrer" className="nav-link
Ripple-parent" href="https://mdbootstrap.com/docs/react/getting-
started/download/" target="_blank">Free download</a>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a rel="noopener noreferrer"  className="nav-link
Ripple-parent" href="https://mdbootstrap.com/bootstrap-tutorial/"
target="_blank">Free tutorials</a>
                        </MDBNavItem>
                    </MDBNavbarNav>
                    <MDBNavbarNav right>
                        <MDBNavItem>
                            <a className="nav-link navbar-link" rel="noopener
noreferrer" target="_blank" href="https://pl-
pl.facebook.com/mdbootstrap/"><MDBIcon fab icon="facebook" /></a>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a className="nav-link navbar-link" rel="noopener
noreferrer" target="_blank" href="https://twitter.com/mdbootstrap"><MDBIcon
fab icon="twitter" /></a>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a className="border border-light rounded mr-1
nav-link Ripple-parent" rel="noopener noreferrer"
href="https://github.com/mdbootstrap/React-Bootstrap-with-Material-Design"
target="_blank"><MDBIcon fab icon="github" className="mr-2"/>MDB GitHub</a>
                        </MDBNavItem>
                        <MDBNavItem>
                            <a className="border border-light rounded mr-1
nav-link Ripple-parent" rel="noopener noreferrer"
href="https://mdbootstrap.com/products/react-ui-kit/"
```

```
target="_blank"><MDBIcon fab icon="github" className="mr-2"/>Go Pro</a>
                        </MDBNavItem>
                    </MDBNavbarNav>
                </MDBCollapse>
            </MDBNavbar>
        );
    }
}

export default TopNavigation;
```

Router.js

```
import React from 'react';
import { Route, Switch } from 'react-router-dom';
import DashboardPage from './pages/DashboardPage';
import ProfilePage from './pages/ProfilePage';
import TablesPage from './pages/TablesPage';
import MapsPage from './pages/MapsPage';
import NotFoundPage from './pages/NotFoundPage';
import UserDetails from "./views/UserManagement/UserManage";
import Courses from "./views/CourseManagement/CourseManagement"
import CourseAdd from "./views/CourseManagement/CourseAdd"
import Login from "./views/Login/Login"

class Routes extends React.Component {
  render() {
    return (
      <Switch>
        <Route path='/' exact component={UserDetails} />
        {/*<Route path='/profile' component={ProfilePage} />*/}
          <Route exact path='/studentManagement' component={UserDetails}/>
          <Route exact path='/lecture' component={Courses}/>
          <Route exact path='/lectureManagement' component={CourseAdd}/>
          <Route exact path='/logout' component={Login}/>
        <Route path='/404' component={NotFoundPage} />
      </Switch>
    );
  }
}

export default Routes;
```

CourseAdd.js

```
 import React, {Component} from "react";
import {BrowserRouter as Router, NavLink} from 'react-router-dom';
import {
    MDBMask,
    MDBRow,
    MDBCol,
    MDBView,
```

```
    MDBContainer, MDBPagination, MDBPageItem, MDBPageNav,
    MDBNavbar,
    MDBIcon, MDBDataTable, MDBFormInline,
    MDBCard,
    MDBCardBody,
    MDBCardTitle,
    MDBInput, MDBBtn, MDBTableHead, MDBTableBody, MDBTable, MDBAlert,
MDBBtnGroup
} from 'mdbreact';

import './CourseAdd.css';
import 'sweetalert2/src/sweetalert2.scss';
import Swal from 'sweetalert2/dist/sweetalert2.js'
import constants from "../../../constants/constants";
import axios from "axios";
import {InputGroup} from "react-bootstrap";



export default class CourseAdd extends Component {
    constructor(props) {
        super(props);


        this.state = {
            AdminName:'',
            AdminNameValidation: false,
            AdminEmail: '',
            AdminEmailValidation: false,
            AdminPosition: '',
            AdminPositionValidation: false,
            AdminPassword: '',
            AdminPasswordValidation: false,
            AdminCPass: '',
            AdminCPassValidation: false,
            currentPage: 1,
            userPerPage: 2,
            adminList:[],
            selectedId : '',
            selectedName : '',
            selectedEmail : '',
            selectedPosition : '',
            selectedPassword : '',
            selectedConfirm : '',
            edited: false,


        }

        this.sweetalertfunction = this.sweetalertfunction.bind(this);
        this.submitAdmin = this.submitAdmin.bind(this);
        this.getDetails = this.getDetails.bind(this);
        this.onChangeName = this.onChangeName.bind(this);
        this.onChangeEmail = this.onChangeEmail.bind(this);
        this.onChangePosition = this.onChangePosition.bind(this);
        this.onChangePassword = this.onChangePassword.bind(this);
        this.onChangeConfirmPass = this.onChangeConfirmPass.bind(this);
```

```
        this.editAdmin = this.editAdmin.bind(this);
        this.editAdminUpdate = this.editAdminUpdate.bind(this);

        this.firstPage = this.firstPage.bind(this);
        this.prevPage = this.prevPage.bind(this);
        this.nextPage = this.nextPage.bind(this);
        this.lastPage = this.lastPage.bind(this);

        this.onChangeName2 = this.onChangeName2.bind(this);
        this.onChangeEmail2 = this.onChangeEmail2.bind(this);
        this.onChangePosition2 = this.onChangePosition2.bind(this);
        this.onChangePassword2 = this.onChangePassword2.bind(this);
        this.onChangeConfirmPass2 = this.onChangeConfirmPass2.bind(this);

    }

    componentDidMount() {
        this.getDetails();
        // if(localStorage.getItem("Position") !=="Admin"){
        //     this.props.history.push('/');
        // }
    }

    getDetails(){
        console.log("get Admin Details");
        axios.get(constants.backend_url +
'api/adminDetail/getAlldetail').then(response => {
            console.log(response.data);
            this.setState({adminList:response.data})
            // this.countgender();
        }).catch(function (error) {
            console.log(error);
        })
    }


    onChangeName(e)
    {
        this.setState({
            AdminName: e.target.value,
            AdminNameValidation: false
        });
    }

    onChangeEmail(e)
    {
        this.setState({
            AdminEmail: e.target.value,
            AdminEmailValidation: false
        });

    }

    onChangePosition(e)
    {
        this.setState({
            AdminPosition:  e.target.value,
```

```
            AdminPositionValidation: false
        });

    }

    onChangePassword(e)
    {
        this.setState({
            AdminPassword: e.target.value,
            AdminPasswordValidation: false
        });

    }

    onChangeConfirmPass(e)
    {
        this.setState({
            AdminCPass: e.target.value,
            AdminCPassValidation: false
        });

    }



    onChangeName2(e)
    {
        this.setState({
            selectedName: e.target.value,
            AdminNameValidation: false
        });
    }

    onChangeEmail2(e)
    {
        this.setState({
            selectedEmail: e.target.value,
            AdminEmailValidation: false
        });

    }

    onChangePosition2(e)
    {

        this.setState({
            selectedPosition: e.target.value,
            AdminPositionValidation: false
        });


    }

    onChangePassword2(e)
    {
        this.setState({
            selectedPassword: e.target.value,
```

```
            AdminPasswordValidation: false
        });

    }

    onChangeConfirmPass2(e)
    {
        this.setState({
            selectedConfirm: e.target.value,
            AdminCPassValidation: false
        });

    }


    firstPage(){
        if(this.state.currentPage > 1){
            this.setState({
                currentPage: 1
            })
        }
    }

    prevPage(){
        if(this.state.currentPage > 1){
            this.setState({
                currentPage: this.state.currentPage - 1
            })
        }

    }

    nextPage(){

        if(this.state.currentPage < Math.ceil(this.state.adminList.length /
this.state.userPerPage)){
            this.setState({
                currentPage: this.state.currentPage + 1
            })
        }

    }

    lastPage(){

        if(this.state.currentPage < Math.ceil(this.state.adminList.length /
this.state.userPerPage)){
            this.setState({
                currentPage: Math.ceil(this.state.adminList.length /
this.state.userPerPage)
            })
        }

    }

    editAdmin(id, module, lecture, incharge, description, video){
        console.log(id)
```

```
        this.setState({
            selectedId:id,
            selectedName : module,
            selectedEmail : lecture,
            selectedPosition : incharge,
            selectedPassword : description,
            selectedConfirm : video,
            edited : true

        })

    };

    editAdmindel(id, module, lecture, incharge, description, video){
        console.log(id)
        this.setState({
            selectedId:id,
            selectedName : module,
            selectedEmail : lecture,
            selectedPosition : incharge,
            selectedPassword : description,
            selectedConfirm : video,
            edited : false

        })

    };

    editAdminUpdate(id, module, lecture, incharge, description, video){
            if(this.state.selectedName !== ''){
                if(this.state.selectedEmail !== ''){
                    if(this.state.selectedPosition !== ''){
                        if(this.state.selectedPassword !== ''){
                            if(this.state.selectedConfirm !== '' ){

                                    axios.get(constants.backend_url +
'api/adminDetail/updateDetail/'+ id+'/'+ module+'/'+ lecture+'/'+
incharge+'/'+ description +'/'+ video).then(response => {
                                        if (response.data.adminUpdate ===
'successful') {

                                            Swal.fire(
                                                '',
                                                'Lecture Updated !.',
                                                'success'
                                            )

                                            this.setState({
                                                selectedId:'',
                                                selectedName : '',
                                                selectedEmail : '',
                                                selectedPosition : '',
                                                selectedPassword : '',
                                                selectedConfirm : '',
                                                edited : false})
                                        }else {
                                            Swal.fire(
                                                '',
```

```
                                                'Update Failed !',
                                                'error'
                                            )}


                                        });

                                    }else{console.log("cpass empty");
                                        this.setState({
                                            AdminCPassValidation: true
                                        })
                                    }
                                }else{console.log("pass empty");
                                    this.setState({
                                        AdminPasswordValidation: true
                                    })
                                }
                            }else{console.log("position empty");
                                this.setState({
                                    AdminPositionValidation: true
                                })
                            }
                        }else{console.log("Email empty");
                            this.setState({
                                AdminEmailValidation: true
                            })
                        }
                    }else{console.log("Name empty''");
                        this.setState({
                            AdminNameValidation: true
                        })
                    }




        this.getDetails();
    }

    sweetalertfunction(id){
        console.log("button clicks");
        const swalWithBootstrapButtons = Swal.mixin({
            customClass: {
                confirmButton: 'btn btn-success',
                cancelButton: 'btn btn-danger'
            },
            buttonsStyling: false
        })

        swalWithBootstrapButtons.fire({
            title: 'Are you sure?',
            text: "You won't be able to revert this!",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonText: 'Yes, delete it!',
```

```
            cancelButtonText: 'No, cancel!',
            reverseButtons: true
        }).then((result) => {
            if (result.value) {
                axios.get(constants.backend_url +
'api/adminDetail/deleteAdmin/'+ id).then(response => {
                    if (response.data.adminDelete === 'success') {
                        swalWithBootstrapButtons.fire(
                            '',
                            'Delete Failed !.',
                            'error'
                        )


                    }else {
                        Swal.fire(
                            '',
                            'Lecture Deleted !',
                            'success'
                        )
                        this.setState({
                            AdminName:'',
                            AdminNameValidation: false,
                            AdminEmail: '',
                            AdminEmailValidation: false,
                            AdminPosition: '',
                            AdminPositionValidation: false,
                            AdminPassword: '',
                            AdminPasswordValidation: false,
                            AdminCPass: '',
                            AdminCPassValidation: false,
                            currentPage: 1,
                            userPerPage: 5,
                            adminList:[],
                            selectedId : '',
                            selectedName : '',
                            selectedEmail : '',
                            selectedPosition : '',
                            selectedPassword : '',
                            selectedConfirm : '',
                            edited: false
                        })
                        this.editAdmindel(this.state.selectedId,
this.state.selectedName, this.state.selectedEmail,
this.state.selectedPosition, this.state.selectedPassword ,
this.state.selectedConfirm);
                        this.getDetails();
                    }
                })

            } else if (
                /* Read more about handling dismissals below */
                result.dismiss === Swal.DismissReason.cancel
            ) {
                swalWithBootstrapButtons.fire(
                    'Cancelled',
                    'Lecture not deleted',
```

```
                    'error'
                )
            }
        })
    }



    submitAdmin(event) {
        event.preventDefault();
        // event.target.className += " was-validated";


            if(this.state.AdminName !== ''){
                if(this.state.AdminEmail !== ''){
                    if(this.state.AdminPosition !== ''){
                        if(this.state.AdminPassword !== ''){
                            if(this.state.AdminCPass !== '' ){
                                    console.log("Validation complete");
                                    const newadminDetail = {
                                        Module: this.state.AdminName,
                                        Lecture: this.state.AdminEmail,
                                        Incharge: this.state.AdminPosition,
                                        Description:
this.state.AdminPassword,
                                        Video:this.state.AdminCPass

                                    }
                                    console.log("newAdminDetail")
                                    console.log(newadminDetail)
                                    console.log("newAdminDetail")
                                    axios.post(constants.backend_url +
'api/adminDetail/add', newadminDetail)
                                        .then(res => {
                                                console.log(res)
                                                console.log(newadminDetail);
                                                if (res.data.AdminDetail ===
'successful') {

console.log("************************************************");
                                                    Swal.fire(
                                                        '',
                                                        'Lecture added
successfully !.',
                                                        'success'
                                                    );
                                                    this.setState({
                                                        AdminName:'',
                                                        AdminNameValidation:
false,
                                                        AdminEmail: '',
                                                        AdminEmailValidation:
false,
                                                        AdminPosition: '',

AdminPositionValidation: false,
```

```
                                                    AdminPassword: '',
AdminPasswordValidation: false,
                                                    AdminCPass: '',
                                                    AdminCPassValidation:
false
                                                })
                                                this.getDetails();

                                            } else {
                                                Swal.fire(
                                                    '',
                                                    'Lecture not added
!',
                                                    'error'
                                                )
                                            }
                                        }
                                    );

                                }else{console.log("cpass empty");
                                    this.setState({
                                        AdminCPassValidation: true
                                    })
                                }
                            }else{console.log("pass empty");
                                this.setState({
                                    AdminPasswordValidation: true
                                })
                            }
                        }else{console.log("position empty");
                            this.setState({
                                AdminPositionValidation: true
                            })
                        }
                    }else{console.log("Email empty");
                        this.setState({
                            AdminEmailValidation: true
                        })
                    }
                }else{console.log("Name empty'");
                    this.setState({
                        AdminNameValidation: true
                    })
                }
    };



    render() {
        const{adminList, currentPage, userPerPage} = this.state;
        const lastIndex = currentPage * userPerPage;
        const firstIndex = lastIndex - userPerPage;
        const currentUsers = adminList.slice(firstIndex, lastIndex );
        const totalPages = Math.ceil(adminList.length / userPerPage) ;
        return (
```

```
            <div id='parallaxintro'>

                <MDBRow>
                    <MDBCol size="8">
                        <MDBCard>
                            <MDBCardBody>

                                <form>
                                    <MDBTable responsive>
                                        <MDBTableHead color="secondary-color"
textWhite>
                                            <tr>
                                                <th>Module</th>
                                                <th>Lecture</th>
                                                <th>Incharge</th>
                                                <th>Description</th>
                                                <th>Video Source</th>
                                                <th>Action</th>
                                            </tr>
                                        </MDBTableHead>
                                        <MDBTableBody>

                                            {
                                                currentUsers.length === 0 ?
                                                    <tr >
                                                        <td colSpan="12"
style={{textAlign : "center", fontWeight: "bold"}}>
                                                            <MDBAlert
color="danger" >

                                                                No Courses
Added

                                                            </MDBAlert>
                                                        </td>
                                                    </tr> :
                                                    currentUsers.map(item =>
{

                                                        return(



                                                            <tr>

<td>{item.Module}</td>

<td>{item.Lecture}</td>

<td>{item.Incharge}</td>

<td>{item.Description}</td>

<td>{item.Video}</td>

                                                                <td>
                                                                    <MDBBtn
tag="a" size="sm" color="success" onClick={()=>this.editAdmin(item._id,
item.Module, item.Lecture, item.Incharge, item.Description, item.Video )} >

<MDBIcon size="lg" icon="pen" />
```

```
</MDBBtn>   

                                                                    <MDBBtn
tag="a" size="sm" color="danger"  onClick={() =>
this.sweetalertfunction(item._id)}>

<MDBIcon size="lg" icon="times-circle" />
                                                                    </MDBBtn>
                                                                </td>
                                                            </tr>

                                                        )
                                                    })}
                                                </MDBTableBody>
                                            </MDBTable>
                                            <div style={{"float" : "left" , "color" :
"#6f42c1"}}> Showing Page {currentPage} of {totalPages} </div>
                                            <div style={{"float" : "right" }}>
                                                <InputGroup>

<InputGroup.Prepend></InputGroup.Prepend>
                                                    <MDBBtnGroup>
                                                        <MDBBtn color="secondary"
size="sm" disabled={currentPage === 1 ? true : false}
onClick={this.firstPage}>First</MDBBtn>
                                                        <MDBBtn color="secondary"
size="sm" disabled={currentPage === 1 ? true : false} onClick={this.prevPage}
>Prev</MDBBtn>
                                                    </MDBBtnGroup>

                                                    <input type="text"
className="pageNumCss" name="currentPage" value={currentPage}
onChange={this.changePage} disabled/>
                                                    <InputGroup.Append>
                                                        <MDBBtnGroup>
                                                            <MDBBtn color="secondary"
size="sm" disabled={currentPage === totalPages ? true : false}
onClick={this.nextPage}>Next</MDBBtn>
                                                            <MDBBtn color="secondary"
size="sm" disabled={currentPage === totalPages ? true : false}
onClick={this.lastPage}>Last</MDBBtn>
                                                        </MDBBtnGroup>
                                                    </InputGroup.Append>
                                                </InputGroup>
                                            </div>

                                        </form>
                                    </MDBCardBody>
                                </MDBCard>
                            </MDBCol>

                            <MDBCol md="4">
                                <MDBCard>
                                    <MDBCardBody>

                                        {
```

```
                                    this.state.edited ?

                                        <form >
                                            <p className="h4 text-center py-
1">Course Manage</p>
                                            <label
htmlFor="defaultFormCardNameEx1"
                                                   className="grey-text font-
weight-light">Module</label>
                                            <div className="input-group">
                                                <div className="input-group-
prepend">
                                            <span className="input-group-
text" id="basic-addon">
                                              <i className="fa fa-tasks
prefix"></i>
                                            </span>
                                                </div>
                                                <input
                                                    type="text"
                                                    className="form-control"
                                                    placeholder="Module"
                                                    aria-label="Module"
                                                    aria-describedby="basic-
addon"
                                            value={this.state.selectedName}
                                            onChange={this.onChangeName2}
                                                />
                                                {
this.state.AdminNameValidation ? <MDBAlert color="danger">
                                                        Module Field Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>


                                            <label
htmlFor="defaultFormCardEmailEx2"
                                                   className="grey-text font-
weight-light">Lecture</label>

                                            <div className="input-group">
                                                <div className="input-group-
prepend">
                                            <span className="input-group-
text" id="basic-addon">
                                            <MDBIcon icon="book" />
                                            </span>
                                                </div>
                                                <input
                                                    type="text"
                                                    className="form-control"
                                                    placeholder="Lecture"
                                                    aria-label="Lecture"
```

```
                                                    aria-describedby="basic-
addon"

value={this.state.selectedEmail}

onChange={this.onChangeEmail2}
                                                />
                                                {

this.state.AdminEmailValidation ? <MDBAlert color="danger">
                                                        Lecture Field Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>


                                            <label
htmlFor="defaultFormCardEmailEx3"
                                                   className="grey-text font-
weight-light">Incharge</label>

                                            <div className="input-group">
                                                <div className="input-group-
prepend">
                                            <span className="input-group-
text" id="basic-addon">
                                            <MDBIcon icon="chalkboard-
teacher" />
                                            </span>
                                                </div>
                                                <input
                                                    type="text"
                                                    className="form-control"
                                                    placeholder="Incharge"
                                                    aria-label="Incharge"
                                                    aria-describedby="basic-
addon"

value={this.state.selectedPosition}

onChange={this.onChangePosition2}
                                                />
                                                {

this.state.AdminPositionValidation ? <MDBAlert color="danger">
                                                        Incharge Field Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>


                                            <label
htmlFor="defaultFormCardEmailEx4"
                                                   className="grey-text font-
weight-light">Description</label>

                                            <div className="input-group">
```

```
                                        <div className="input-group-
prepend">
                                    <span className="input-group-
text" id="basic-addon">
                                    <MDBIcon icon="file" />
                                    </span>
                                        </div>
                                        <input
                                            type="text"
                                            className="form-control"
                                            placeholder="Description"
                                            aria-label="Description"
                                            aria-describedby="basic-
addon"
value={this.state.selectedPassword}
onChange={this.onChangePassword2}
                                        />
                                        {
this.state.AdminPasswordValidation ? <MDBAlert color="danger">
                                                Description Field
Empty
                                            </MDBAlert> : ''
                                        }
                                    </div>

                                    <label
htmlFor="defaultFormCardEmailEx5"
                                       className="grey-text font-
weight-light">Video Source</label>

                                    <div className="input-group">
                                        <div className="input-group-
prepend">
                                    <span className="input-group-
text" id="basic-addon">
                                    <MDBIcon fab icon="youtube" />
                                    </span>
                                        </div>
                                        <input
                                            type="text"
                                            className="form-control"
                                            placeholder="Video
Source"
                                            aria-label="VideoSource"
                                            aria-describedby="basic-
addon"
value={this.state.selectedConfirm}
onChange={this.onChangeConfirmPass2}
                                        />
                                        {
this.state.AdminCPassValidation ?
```

```
                                                    <MDBAlert
color="danger">
                                                        Video Source Fild
Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>

                                            <div className="text-center py-4
mt-0">
                                                <MDBBtn outline
color="success" type="button"
onClick={()=>this.editAdminUpdate(this.state.selectedId,this.state.selectedNa
me,this.state.selectedEmail,this.state.selectedPosition,this.state.selectedPa
ssword,this.state.selectedConfirm)}>
                                                    <b>Update Lecture</b>
                                                    <MDBIcon icon="pen"
className="ml-2"/>
                                                </MDBBtn>
                                            </div>
                                        </form>
                                        :

                                        <form onSubmit={this.submitAdmin}>
                                            <p className="h4 text-center py-
1">Course Manage</p>

                                            <label
htmlFor="defaultFormCardNameEx1"
                                                   className="grey-text font-
weight-light">Module</label>

                                            <div className="input-group">
                                                <div className="input-group-
prepend">

                                            <span className="input-group-
text" id="basic-addon">

                                              <i className="fa fa-tasks
prefix"></i>

                                            </span>
                                                </div>
                                                <input
                                                    type="text"
                                                    className="form-control"
                                                    placeholder="Module"
                                                    aria-label="Module"
                                                    aria-describedby="basic-
addon"

value={this.state.AdminName}

onChange={this.onChangeName}
                                                />
                                                {

this.state.AdminNameValidation ? <MDBAlert color="danger">
                                                        Module Field Empty
                                                    </MDBAlert> : ''
                                                }
```

```
                                        </div>


                                        <label
htmlFor="defaultFormCardEmailEx2"
                                               className="grey-text font-
weight-light">Lecture</label>

                                        <div className="input-group">
                                            <div className="input-group-
prepend">
                                        <span className="input-group-
text" id="basic-addon">
                                         <MDBIcon icon="book" />
                                        </span>
                                            </div>
                                            <input
                                                type="text"
                                                className="form-control"
                                                placeholder="Lecture"
                                                aria-label="Lecture"
                                                aria-describedby="basic-
addon"
                                        value={this.state.AdminEmail}
                                        onChange={this.onChangeEmail}
                                            />
                                            {
this.state.AdminEmailValidation ? <MDBAlert color="danger">
                                                    Lecture Field Empty
                                                </MDBAlert> : ''
                                            }
                                        </div>


                                        <label
htmlFor="defaultFormCardEmailEx3"
                                               className="grey-text font-
weight-light">Incharge</label>

                                        <div className="input-group">
                                            <div className="input-group-
prepend">
                                        <span className="input-group-
text" id="basic-addon">
                                        <MDBIcon icon="chalkboard-
teacher" />
                                        </span>
                                            </div>
                                            <input
                                                type="text"
                                                className="form-control"
                                                placeholder="Incharge"
                                                aria-label="Incharge"
                                                aria-describedby="basic-
```

```
addon"
value={this.state.AdminPosition}
onChange={this.onChangePosition}
                                                />
                                                {
this.state.AdminPositionValidation ? <MDBAlert color="danger">
                                                        Incharge Field Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>

                                            <label
htmlFor="defaultFormCardEmailEx4"
                                                   className="grey-text font-
weight-light">Description</label>
                                            <div className="input-group">
                                                <div className="input-group-
prepend">
                                            <span className="input-group-
text" id="basic-addon">
                                             <MDBIcon icon="file" />
                                            </span>
                                                </div>
                                                <input
                                                    type="text"
                                                    className="form-control"
                                                    placeholder="Description"
                                                    aria-label="Description"
                                                    aria-describedby="basic-
addon"
value={this.state.AdminPassword}
onChange={this.onChangePassword}
                                                />
                                                {
this.state.AdminPasswordValidation ? <MDBAlert color="danger">
                                                        Description Field
Empty
                                                    </MDBAlert> : ''
                                                }
                                            </div>

                                            <label
htmlFor="defaultFormCardEmailEx5"
                                                   className="grey-text font-
weight-light">Video Source</label>
                                            <div className="input-group">
                                                <div className="input-group-
```

```
prepend">                                                    <span className="input-group-

text" id="basic-addon">                                      <MDBIcon fab icon="youtube" />
                                                             </span>
                                                                 </div>
                                                                 <input
                                                                     type="text"
                                                                     className="form-control"
                                                                     placeholder="Video

Source"

                                                                     aria-label="VideoSource"
                                                                     aria-describedby="basic-

addon"

value={this.state.AdminCPass}

onChange={this.onChangeConfirmPass}
                                                                 />
                                                                 {

this.state.AdminCPassValidation ?
                                                                         <MDBAlert

color="danger">
                                                                             Video Source Fild
Emplty
                                                                         </MDBAlert> : ''
                                                                 }
                                                             </div>

                                                             <div className="text-center py-4

mt-0">
                                                                 <MDBBtn className="btn btn-

outline-purple" type="submit">
                                                                     <b>Add Lecture</b>
                                                                     <MDBIcon icon="plus"

className="ml-2"/>
                                                                 </MDBBtn>
                                                             </div>
                                                         </form>
                                    }


                                </MDBCardBody>
                            </MDBCard>
                        </MDBCol>
                    </MDBRow>


                </div>
            );
        }
    }
```

CourseManagement.js

```
import React, {Component} from "react";
import {
    MDBCard,
    MDBCol,
    MDBRow,
    MDBView,
    MDBMask,
    MDBCardImage,
    MDBCardBody,
    MDBCardTitle,
    MDBCardText,
    MDBCardFooter,
    MDBBtn,
    MDBIcon,
    MDBAlert
} from 'mdbreact';
import src1 from '../../../assets/img-1.jpg';
import 'sweetalert2/src/sweetalert2.scss';
import constants from "../../../constants/constants";
import axios from "axios";
import {InputGroup} from "react-bootstrap";

export default class Courses extends Component {
    constructor(props) {
        super(props);


        this.state = {
            lectureList:[]
        }

        this.getDetails = this.getDetails.bind(this);
    }


    componentDidMount() {
        this.getDetails();
        // if(localStorage.getItem("Position") !=="Admin"){
        //     this.props.history.push('/');
        // }
    }
    getDetails(){
        console.log("get Lecture Details");
        axios.get(constants.backend_url +
'api/adminDetail/getAlldetail').then(response => {
            console.log(response.data);
            this.setState({lectureList:response.data})

        }).catch(function (error) {
            console.log(error);
        })
    }

    render() {
        return (
```

```
            <MDBRow className="justify-content-center">

                {
                    this.state.lectureList.length === 0 ?
                        <tr >
                            <td colSpan="12" style={{textAlign : "center",
fontWeight: "bold"}}>
                                <MDBAlert color="danger" >
                                    No Courses Added
                                </MDBAlert>
                            </td>
                        </tr> :

                        this.state.lectureList.map(item => {
                            return(
                    <MDBCol sm="12" md="6" lg="3" className="mb-5">
                        <MDBCard>
                            <iframe width="223" height="223"
                                    src={item.Video}
                                    frameBorder="0"
                                    allow="accelerometer; autoplay;
encrypted-media; gyroscope; picture-in-picture"
                                    allowFullScreen>
                            </iframe>
                            <MDBCardBody>
                                <MDBCardTitle className="text-center mb-2
font-bold">{item.Module}</MDBCardTitle>
                                <MDBCardTitle sub
                                              className="text-center indigo-
text mb-2 font-bold">{item.Lecture}</MDBCardTitle>
                                <MDBCardText>
                                    <strong className="mb-2">{item.Incharge}
: </strong>
                                    {item.Description}
                                </MDBCardText>
                            </MDBCardBody>
                        </MDBCard>
                    </MDBCol>
                            )
                        })}

            </MDBRow>

        );
    }
}
```

Login.js

```
import React, { Component } from 'react';
import {
    MDBContainer,
    MDBBtn,
    MDBModal,MDBView,
    MDBModalBody,
```

```
    MDBModalHeader,
    MDBModalFooter,
    MDBCard,
    MDBCardBody, MDBInput, MDBAlert, MDBRow
} from 'mdbreact';
import axios from "axios";
import constants from "../../../constants/constants";
import 'sweetalert2/src/sweetalert2.scss';
import Swal from 'sweetalert2/dist/sweetalert2.js'
import './Login.css'


export default class Login extends Component{
    constructor(props) {
        super(props);
        this.state = {
            collapse: false,
            isWideEnough: false,
            modal: false,
            model2 : false,
            fname: '',
            lname: '',
            email: '',
            gender: '',
            password: '',
            phone: '',
            confirmpass: '',
            dob:'',
            MaleCount:0,
            FemaleCount:0,
            loginEmail:'',
            loginPass: '',
            loginEmailV: false,
            loginPassV: false
        };
        this.onClick = this.onClick.bind(this);
        this.submitUser = this.submitUser.bind(this);
        this.validateUser = this.validateUser.bind(this);
        this.onChangeFname = this.onChangeFname.bind(this);
        this.onChangeLname = this.onChangeLname.bind(this);
        this.onChangeEmail = this.onChangeEmail.bind(this);
        this.onChangePhone = this.onChangePhone.bind(this);
        this.onChangeGender = this.onChangeGender.bind(this);
        this.onChangeDOB = this.onChangeDOB.bind(this);
        this.onChangePassword = this.onChangePassword.bind(this);
        this.onChangeConfirmPassword =
this.onChangeConfirmPassword.bind(this);
        this.onChangeEmailV = this.onChangeEmailV.bind(this);
        this.onChangePassV = this.onChangePassV.bind(this);

    }

    onClick() {
        this.setState({
            collapse: !this.state.collapse,
        });
    }
```

```
    toggle = () => {
        this.setState({
            modal: !this.state.modal
        });
    }

    toggle2 = () => {
        this.setState({
            modal: !this.state.modal,
            modal2: !this.state.model2
        });
    }

    toggle3 = () => {
        this.setState({
            modal2: false,
            fname: "",
            lname: "",
            email: "",
            gender: "",
            password: "",
            phone: "",
            dob: '',
            confirmpass: ""

        });
    }

    onChangeEmailV(event){
        this.setState({
            loginEmail:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangePassV(event){
        this.setState({
            loginPass:event.target.value,
            [event.target.name]: event.target.value
        })
    }


    onChangeFname(event){
        this.setState({
            fname:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangeLname(event){
        this.setState({
            lname:event.target.value,
            [event.target.name]: event.target.value
        })
    }
```

```
    onChangeEmail(event){
        this.setState({
            email:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangeGender(event){
        this.setState({
            gender:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangeDOB(event){
        this.setState({
            dob:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangePassword(event){
        this.setState({
            password:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangeConfirmPassword(event){
        this.setState({
            confirmpass:event.target.value,
            [event.target.name]: event.target.value
        })
    }

    onChangePhone(event){
        this.setState({
            phone:event.target.value,
            [event.target.name]: event.target.value
        })

    }



    submitUser(event) {
        event.preventDefault();
        event.target.className += " was-validated";
        if(this.state.password == this.state.confirmpass){
            if(this.state.fname != ''){
                if(this.state.lname != ''){
                    if(this.state.email != ''){
                        if(this.state.gender != ''){
                            if(this.state.phone != '' ){
                                if(this.state.password != ''){
                                    if(this.state.confirmpass != ''){
```

```
                                                    console.log("Validation
complete");
                                                        const newuserDetail = {
                                                            firstName:
this.state.fname,
                                                            lastName:
this.state.lname,
                                                            phoneNumber:
this.state.phone,
                                                            stdNumber:
this.state.gender,
                                                            email: this.state.email,
                                                            password:
this.state.password,
                                                        }
axios.post(constants.backend_url + 'api/userDetail/add', newuserDetail)
                                                            .then(res => {
                                                                console.log(res)
console.log(newuserDetail);
                                                                if
(res.data.userDetail === 'successful') {
                                                                    Swal.fire(
                                                                        '',
                                                                        'User
Details added successfully !.',
                                                                        'success'
                                                                    );
this.setState({
                                                                        fname:
'',
                                                                        lname:
'',
                                                                        email:
'',
                                                                        gender:
'',
                                                                        password:
'',
                                                                        phone:
'',
                                                                        dob: '',
confirmpass: ''
                                                                    })
                                                                } else {
                                                                    Swal.fire(
                                                                        '',
                                                                        'Email
already in use !',
                                                                        'error'
                                                                    )
                                                                }
                                                            }
```

```
                                                    );


                                        }else{console.log(" confirm pass
empty");}
                                    }else{console.log("pass empty");}
                                }else{console.log("phone empty");}
                            }else{console.log("gender empty");}
                        }else{console.log("email empty");}
                    }else{console.log("lname empty");}
                }else{console.log("fname == ''");}
            }else{console.log("pass != confirm pass");
                Swal.fire(
                    '',
                    'password and confirm password are not the same !',
                    'error'
                );
            }
        };


        validateUser(event){
            event.preventDefault();

            if(this.state.loginEmail != '' ){
                this.setState({
                    loginEmailV: false
                })
                if(this.state.loginPass != '' ){
                    this.setState({
                        loginPassV: false
                    })

                    axios.get(constants.backend_url +
'api/userDetail/validateUser/' + this.state.loginEmail + '/' +
this.state.loginPass)
                        .then(res => {
                                if (res.data.Message !== 'unsuccessful') {

localStorage.setItem("CustomerLogged","CustomerLogged");

localStorage.setItem("CustomerObject","CustomerLogged");
                                    // this.context.router.push('/');
                                    this.props.history.push('/profile');

                                    window.location.reload();

                                    this.setState({
                                        loginEmail: '',
                                        loginPass:'',
                                        loginEmailV: false,
                                        loginPassV:false
                                    });

                                } else {
                                    Swal.fire(
                                        '',
```

```
                                    'Login unsuccessful  !',
                                    'error'
                                )
                            }
                        }
                    );


            }else{
                console.log('email field empty');
                this.setState({
                    loginPassV: true
                })
            }
        }else{
            console.log('email field empty');
            this.setState({
                loginEmailV: true
            })


        }
    };

    render() {
        return (
            <div id='apppage'>
                {this.props.children}
                <MDBView>
                <MDBContainer>

                    {/*<MDBBtn color="primary"
onClick={this.toggle}>Login</MDBBtn>*/}
                    <MDBBtn color="secondary" style={{width: "300px",
position: "absolute", top:"300px", left:"100px"  }}  onClick={this.toggle}>
<i className="fas fa-user"></i>   Login</MDBBtn>
                    <MDBModal isOpen={this.state.modal} toggle={this.toggle}>
                        <MDBModalHeader toggle={this.toggle}
></MDBModalHeader>
                        <MDBModalBody>
                            <form className="needs-validation"
onSubmit={this.validateUser} >
                                <MDBCard>
                                    <MDBCardBody className="mx-4">
                                        <div className="text-center">
                                            <h2 className="loginh3 mb-5">
                                                <strong className="loginh3
">LOGIN</strong>
                                            </h2>
                                        </div>
                                        {/*<input
value={this.state.loginEmail} onChange={this.onChangeEmailV}/>*/}
                                        <MDBInput label="Your email" group
type="email" validate error="wrong" success="right"
value={this.state.loginEmail} onChange={this.onChangeEmailV}/>
                                        {
                                            this.state.loginEmailV  ?
```

```
                                                <MDBAlert color="danger">
                                                    Please enter a value for
email !
                                                </MDBAlert> : ''
                                            }
                                            <br/>
                                            {/*<input
value={this.state.loginPass} onChange={this.onChangePassV}/>*/}
                                            <MDBInput label="Your password" group
type="password" validate containerClass="mb-0" value={this.state.loginPass}
onChange={this.onChangePassV}/>
                                            {
                                                this.state.loginPassV  ?
                                                <MDBAlert color="danger">
                                                    Please enter a value for
email !
                                                </MDBAlert> : ''
                                            }
                                            <br/>
                                            <div className="text-center mb-3">
                                                <MDBBtn type="submit"
gradient="blue" rounded className="btn-block z-depth-1a">
                                                    LOGIN
                                                </MDBBtn>
                                            </div>
                                        </MDBCardBody>
                                    </MDBCard>
                                    <MDBModalFooter className="mx-5 pt-3 mb-1">
                                        <p className="font-small grey-text d-flex
justify-content-end">
                                            Not a member?
                                            <MDBBtn outline color="info"
size="sm" onClick={this.toggle2}>Sign In</MDBBtn>
                                        </p>
                                    </MDBModalFooter>
                                </form>
                            </MDBModalBody>
                        </MDBModal>
                    </MDBContainer>
                    </MDBView>

                    <MDBContainer>
                        <MDBModal isOpen={this.state.modal2}
toggle={this.toggle3}>
                            <MDBModalHeader toggle={this.toggle3}
></MDBModalHeader>
                            <MDBModalBody>
                                <MDBCard>
                                    <MDBCardBody className="mx-2">
                                        <div className="text-center">
                                            <h2 className="loginh3 mb-1">
                                                <strong className="loginh3
">REGISTER</strong>
                                            </h2>
                                        </div>
                                        {/*<MDBInput label="Your email" group
type="email" validate error="wrong" success="right"/>*/}
```

```
                                        {/*<MDBInput label="Your password" group
type="password" validate containerClass="mb-0"/>*/}

                                        <form className="needs-validation"
onSubmit={this.submitUser} noValidate>
                                            <MDBRow>
                                                {/*<MDBCol  md="12"
className="mb-3">*/}
                                                <label htmlFor="firstnameid"
className="grey-text">First name</label>
                                                <input value={this.state.fname}
name="fname" onChange={this.onChangeFname} type="text" id="firstnameid"
className="form-control" placeholder="First name" required/>
                                                <div className="invalid-
feedback">Please provide the first name.</div>

                                                {/*</MDBCol>*/}
                                                {/*<MDBCol  md="12"
className="mb-3">*/}
                                                <label htmlFor="lastnameid"
className="grey-text">Last name</label>
                                                <input value={this.state.lname}
name="lname" onChange={this.onChangeLname} type="text" id="lastnameid"
className="form-control" placeholder="Last name" required/>
                                                <div className="invalid-
feedback">Please provide the last name.</div>

                                                {/*</MDBCol>*/}
                                                {/*<MDBCol md="4" className="mb-
3">*/}
                                                <label htmlFor="emailid"
className="grey-text">Email</label>
                                                <input value={this.state.email}
onChange={this.onChangeEmail}  type="email" id="emailid" className="form-
control" name="email" placeholder="Your Email address"/>
                                                <div className="invalid-
feedback">Please provide an email.</div>

                                                {/*</MDBCol>*/}
                                            </MDBRow>
                                            <MDBRow>
                                                {/*<MDBCol md="6" className="mb-
0">*/}
                                                <label htmlFor="phoneid"
className="grey-text">Contact number</label>
                                                <input
onChange={this.onChangePhone} type="text" value={this.state.phone}
id="phoneid" className="form-control" name="phone" placeholder="Contact
number" required/>
                                                <div className="invalid-
feedback">Please provide the contact number</div>

                                                {/*</MDBCol>*/}


                                                {/*<MDBCol md="6" className="mb-
0">*/}
```

```
                                            <label htmlFor="passwordid"
className="grey-text">Password</label>
                                            <input
onChange={this.onChangePassword} type="password" value={this.state.password}
id="passwordid" className="form-control" name="password"
placeholder="Password" required/>
                                            <div className="invalid-
feedback">Please provide the password</div>

                                            {/*</MDBCol>*/}


                                            {/*<MDBCol md="6" className="mb-
0">*/}
                                            <label htmlFor="conpasswordid"
className="grey-text">Confirm Password</label>
                                            <input
onChange={this.onChangeConfirmPassword} type="password"
value={this.state.confirmpass} id="conpasswordid" className="form-control"
name="confirmpass" placeholder="Confirm Password" required/>
                                            <div className="invalid-
feedback">Please provide the confirm password</div>
                                            {
                                                this.state.password !=
this.state.confirmpass ? <MDBAlert color="danger">
                                                    password and confirm
password does not match
                                                </MDBAlert> : ''
                                            }


                                            {/*</MDBCol>*/}

                                            {/*<MDBCol md="6" className="mb-
0">*/}
                                            <label htmlFor="genderid"
className="grey-text">Student Number</label>
                                            <input value={this.state.gender}
onChange={this.onChangeGender} type="text" id="genderid" className="form-
control" name="gender" placeholder="Student Number" required/>
                                            {/*<MDBSelect*/}
                                            {/*
options={this.state.options}*/}
                                            {/*    selected="Choose your
option"*/}
                                            {/*    color="primary"*/}
                                            {/*    label="Example label"*/}
                                            {/*/>*/}
                                            {/*<select id="cars"
onChange={this.changeHandler}>*/}
                                            {/*    <option
value="volvo">Volvo</option>*/}
                                            {/*    <option
value="saab">Saab</option>*/}
                                            {/*    <option
value="opel">Opel</option>*/}
                                            {/*    <option
```

```
value="audi">Audi</option>*/}
                                                    {/*</select>*/}
                                                    <div className="invalid-
feedback">Please provide ID.</div>
                                                    {/*</MDBCol>*/}
                                                </MDBRow>
                                                <br></br>
                                                <MDBBtn  gradient="blue" rounded
className="btn-block z-depth-1a" type="submit">
                                                    REGISTER
                                                </MDBBtn>
                                            </form>
                                        </MDBCardBody>
                                    </MDBCard>
                                </MDBModalBody>
                            </MDBModal>
                        </MDBContainer>
                    </div>
                );
            }
}
```

Logout.js

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

export default class Logout extends Component {

    componentDidMount(){
        localStorage.removeItem('CustomerLogged');
        localStorage.removeItem('CustomerId');
        this.props.history.push('/');
        window.location.reload();
    }
    render() {

        return (
            <div>
            </div>
        );
    }

}
```

UserManage.js

```
import React, {Component} from "react";
import {BrowserRouter as Router, NavLink} from 'react-router-dom';
import {
    MDBMask,
    MDBRow,
    MDBCol,
    MDBView,
    MDBContainer,
    MDBBtnGroup,
    MDBIcon, MDBFormInline,
```

```
    MDBCard,
    MDBCardBody,
    MDBBtn, MDBTableHead, MDBTableBody, MDBTable, MDBAlert, MDBPagination,
MDBPageItem, MDBPageNav,
} from 'mdbreact';
import './UserManage.css';
import 'sweetalert2/src/sweetalert2.scss';
import Swal from 'sweetalert2/dist/sweetalert2.js'
import constants from "../../../constants/constants";
import axios from "axios";
import {InputGroup} from 'react-bootstrap';


export default class UserDetails extends Component {

    constructor(props) {
        super(props);
        this.state = {
            cartitem: '',
            detailList: [],
            currentPage: 1,
            userPerPage: 5,
            empty: false,
            searched: false,
            CustomerName: '',
            foundSearch: false,
            searchList: []


        }

        this.sweetalertfunction = this.sweetalertfunction.bind(this);
        this.getDetails = this.getDetails.bind(this);
        this.changePage = this.changePage.bind(this);
        this.firstPage = this.firstPage.bind(this);
        this.prevPage = this.prevPage.bind(this);
        this.nextPage = this.nextPage.bind(this);
        this.lastPage = this.lastPage.bind(this);
        this.onChangeName = this.onChangeName.bind(this);



    }

    componentDidMount() {
        this.getDetails();

    }

    sweetalertfunction(id) {

        console.log("button clicks");
        const swalWithBootstrapButtons = Swal.mixin({
            customClass: {
                confirmButton: 'btn btn-success',
                cancelButton: 'btn btn-danger'
            },
```

```
            buttonsStyling: false
        })

        swalWithBootstrapButtons.fire({
            title: 'Are you sure?',
            text: "You won't be able to revert this!",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonText: 'Yes, delete it!',
            cancelButtonText: 'No, cancel!',
            reverseButtons: true
        }).then((result) => {
            if (result.value) {
                axios.get(constants.backend_url +
'api/userDetail/deleteUser/' + id).then(response => {
                    if (response.data.userDelete === 'success') {
                        swalWithBootstrapButtons.fire(
                            '',
                            'Delete Failed !.',
                            'error'
                        )
                    } else {
                        Swal.fire(
                            '',
                            'Customer Deleted !',
                            'success'
                        )
                        this.getDetails();
                    }
                })

            } else if (
                /* Read more about handling dismissals below */
                result.dismiss === Swal.DismissReason.cancel
            ) {
                swalWithBootstrapButtons.fire(
                    'Cancelled',
                    'Customer details not deleted',
                    'error'
                )
            }
        })
    }


    getDetails() {
        console.log("get user details");
        axios.get(constants.backend_url +
'api/userDetail/getAllusers').then(response => {
            this.setState({detailList: response.data})
        }).catch(function (error) {
            console.log(error);
        })
    }

    changePage(event) {
        this.setState({
```

```
            CustomerName: event.target.value,
        });
    }

    firstPage() {
        if (this.state.currentPage > 1) {
            this.setState({
                currentPage: 1
            })
        }
    }

    prevPage() {
        if (this.state.currentPage > 1) {
            this.setState({
                currentPage: this.state.currentPage - 1
            })
        }

    }

    nextPage() {

        if (this.state.currentPage < Math.ceil(this.state.detailList.length /
this.state.userPerPage)) {
            this.setState({
                currentPage: this.state.currentPage + 1
            })
        }

    }

    lastPage() {

        if (this.state.currentPage < Math.ceil(this.state.detailList.length /
this.state.userPerPage)) {
            this.setState({
                currentPage: Math.ceil(this.state.detailList.length /
this.state.userPerPage)
            })
        }

    }

    onChangeName(e) {
        this.setState({
            CustomerName: e.target.value,
        });

        console.log("search user");
        if (e.target.value !== '') {

            axios.get(constants.backend_url + 'api/userDetail/seachUser/' +
e.target.value).then(response => {
                if (response.data.message === 'not found') {
                    this.setState({
                        searched: false,
```

```
                        // foundSearch: false

                    })

                    console.log("not found");
                } else {

                    this.setState({
                        searchList: response.data,
                        searched: true,
                        // foundSearch: true
                    })
                    console.log(" found");
                }
            }).catch(function (error) {
                console.log(error);
            })
        } else {
            this.setState({
                searched: false,

            })
            this.getDetails();
        }


    }

    render() {
        const {detailList, currentPage, userPerPage, CustomerName} =
this.state;
        const lastIndex = currentPage * userPerPage;
        const firstIndex = lastIndex - userPerPage;
        const currentUsers = detailList.slice(firstIndex, lastIndex);
        const totalPages = Math.ceil(detailList.length / userPerPage);


        return (

            <div id='parallaxintro'>


                <MDBView>
                    <MDBMask className='rgba-white-light'/>
                    <MDBContainer className='d-flex justify-content-center
align-items-center'
                                  style={{height: '100%', width: '100%',
paddingTop: '0rem'}}>
                        <MDBRow>
                            <div className=" container-fluid
AddItemHeight">
                                <MDBRow>
                                    <MDBCol size="12">
                                        <MDBCard>
                                            <MDBCardBody>
                                                <form>
                                                    <MDBTable
```

```
responsive>
                                                                    <MDBTableHead
color="secondary-color" textWhite>
                                                                        <tr>

<th>Name</th>

<th>Email Address</th>

<th>Contact Number</th>

<th>Student Number</th>

<th>Action</th>
                                                                        </tr>

</MDBTableHead>

                                                                    {

currentUsers.length === 0 ?
                                                                        <tr>

<td colSpan="12" style={{

textAlign: "center",

fontWeight: "bold"

}}>

<MDBAlert color="danger">

No Users Registered

</MDBAlert>

</td>
                                                                        </tr>

:

currentUsers.map(item => {

return (

<MDBTableBody>

<tr>

<td>{item.firstName + "  " + item.lastName}</td>

<td>{item.email}</td>

<td>{item.phoneNumber}</td>
```

```
<td>{item.stdNumber}</td>
<td>
<MDBBtn tag="a" size="sm"
color="danger"
onClick={() => this.sweetalertfunction(item._id)}>
<MDBIcon size="lg"
icon="times-circle"/>
</MDBBtn>
</td>
</tr>
</MDBTableBody>
                                                                                )
                                                                            })}
                                                                    </MDBTable>
                                                                    <div
style={{"float": "left", "color": "#6f42c1"}}> Showing
                                                                        Page
{currentPage} of {totalPages} </div>
                                                                    <div
style={{"float": "right"}}>
                                                                        <InputGroup>
<InputGroup.Prepend></InputGroup.Prepend>
<MDBBtnGroup>
<MDBBtn color="secondary" size="sm"
disabled={currentPage === 1 ? true : false}
onClick={this.firstPage}>First</MDBBtn>
<MDBBtn color="secondary" size="sm"
disabled={currentPage === 1 ? true : false}
onClick={this.prevPage}>Prev</MDBBtn>
</MDBBtnGroup>
                                                                                <input
type="text" className="pageNumCss"
name="currentPage" value={currentPage}
onChange={this.changePage} disabled/>
```

```
<InputGroup.Append>

<MDBBtnGroup>

<MDBBtn color="secondary" size="sm"

disabled={currentPage === totalPages ? true : false}

onClick={this.nextPage}>Next</MDBBtn>

<MDBBtn color="secondary" size="sm"

disabled={currentPage === totalPages ? true : false}

onClick={this.lastPage}>Last</MDBBtn>

</MDBBtnGroup>

</InputGroup.Append>
                                                                    </InputGroup>
                                                                </div>
                                                            </form>
                                                        </MDBCardBody>
                                                    </MDBCard>
                                                </MDBCol>
                                            </MDBRow>
                                        </div>
                                </MDBRow>
                            </MDBContainer>
                        </MDBView>
                    </div>
            );
        }
}
```

Back End

Lecture.model.js

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;

let adminDetails = new Schema({
    Module :{
        type :String,
        required: true
    },
    Lecture :{
        type :String,
        required: true
    },
    Incharge :{
```

```
        type: String,
        required: true
    },
    Description :{
        type: String,
        required: true
    },
    Video :{
        type :String,
        required: true
    }
});

module.exports = mongoose.model('AdminDetail',adminDetails);
```

User.model.js

```
const mongoose = require('mongoose');
const Schema= mongoose.Schema;

let userdetails = new Schema({
    firstName :{
        type :String,
        required: true
    },
    lastName :{
        type :String,
        required: true
    },
    phoneNumber :{
        type: String,
        required: true
    },
    email :{
        type: String,
        required: true
    },
    password :{
        type :String,
        required: true
    },
    stdNumber :{
        type :String,
        required: true
    }
});

module.exports = mongoose.model('userdetails',userdetails);
```

lecture.server.routes

```
const express = require('express');
const mongoose = require('mongoose');
const router = express.Router();
let AdminDetail = require('../Models/Lecture.model')
```

```
router.route('/add').post(function (req,res) {
    let adminDetail = new AdminDetail(req.body);
    adminDetail.save()
        .then(sup=>{
            res.status(200).json({'AdminDetail':'successful'});
        }).catch(err=>{
        res.status(400).send('fail');
    });
});

router.route('/getAlldetail').get(function (req,res) {

    AdminDetail.find().exec().then(item => {

        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});

router.route('/deleteAdmin/:id').get(function (req, res) {
    let id=req.params.id;
    console.log("Delete Admin Called!");
    AdminDetail.deleteOne({_id:id}).then(sup=>{
        console.log("successful");
        res.status(200).json({'adminDelete':'successful'});
    }).catch(err=>{
        console.log("fail");
        res.status(400).send('fail');
    });
});

router.get("/validateUser/:email/:password",function (req,res) {
    let email = req.params.email;
    let password = req.params.password;
    AdminDetail.findOne({ Email: email, password: password },)
        .exec()
        .then(userValid =>{
            if( userValid ){
                res.status(200).json({"Message": userValid});
            }else{
                console.log("Login failed");
            }
        }).catch(err=>{
        res.status(500).json(err);
    })
});

router.route('/updateDetail/:id/:name/:email/:position/:password/:video').get
(function (req, res) {
    console.log("update function called")
    let id = req.params.id;
    let name=req.params.name;
    let email=req.params.email;
    let position=req.params.position;
```

```
    let password=req.params.password;
    let video=req.params.video;

    AdminDetail.updateOne({_id : id},{$set: {Module:name, Lecture: email,
Incharge: position, Description: password, Video: video}}).then(sup=>{
        console.log(" successfully edited");
        console.log(sup);
        res.status(200).json({'adminUpdate':'successful'});
    }).catch(err=>{
        console.log("update fail");
        res.status(400).send('fail');
    });
});

module.exports = router;
```

user.server.routes

```
const express = require('express');
const mongoose = require('mongoose');
const router = express.Router();
let userdetails = require('../Models/User.model');


router.route('/add').post(function (req,res) {
    let userDetail = new userdetails(req.body);
    console.log("------------------------------------");
    console.log(req.body);
    console.log("------------------------------------");
    userdetails.findOne({ email: req.body.email  },)
        .exec()
        .then(userValid =>{
            if( userValid ){
                res.status(200).json({'userDetail': "userAvailable"});
            }else{
                console.log("**************success");

                userDetail.save().then(users => {
                    if(users){
                        res.status(200).json({'userDetail':'successful'});
                    }else{
                        res.status(400).send('failed');
                    }
                }).catch(err=>{
                    res.status(500).json(err);
                })
                res.status(200).json({'userDetail':'successful'});
            }
        }).catch(err=>{
        res.status(500).json(err);
    })


});
```

```
//
router.get("/validateUser/:email/:password",function (req,res) {
    let email = req.params.email;
    let password = req.params.password;
    userdetails.findOne({ email: email, password: password },)
        .exec()
        .then(userValid =>{
            console.log("User Valid");
            console.log(userValid);
            console.log("User Valid");
            if( userValid ){
                res.status(200).json({"Message": userValid});
            }else{
                res.status(200).json({"Message": "unsuccessful"});
            }
        }).catch(err=>{
        res.status(500).json(err);
    })
});


router.route('/getAllusers').get(function (req,res) {
    console.log("getDetails --- usermanage")
    userdetails.find().exec().then(item => {
        console.log(item)
        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});

//
router.route('/getDetailuser/:id').get(function (req,res) {
    console.log("getDetailsof the user---------------------------------------
-------");
    let id = req.params.id;

    userdetails.find({ _id : id }).exec().then(item => {
        console.log(item.email)
        res.status(200).json(item)
    }).catch(err => {
        res.status(500).json(err);
    });
});
//
router.route('/deleteUser/:id').get(function (req, res) {
    let id=req.params.id;
    console.log("Delete Called!");
    userdetails.deleteOne({_id:id}).then(sup=>{
        console.log("successful");
        res.status(200).json({'userDelete':'successful'});
    }).catch(err=>{
        console.log("fail");
        res.status(400).send('fail');
    });
});
```

```
router.route('/seachUser/:cust_name').get(function (req,res) {
    console.log("search entered");
    let name = req.params.cust_name;
    userdetails.find({firstName : name}).exec().then(item => {
        if( item ){
            res.status(200).json(item);
        }else{
            res.status(404).json({"message": "not found"});
        }
    })
        .catch(err=>{
            res.status(500).json(err);
        })
});
//
router.route('/updateDetail/:id/:fname/:lname/:email/:phone/:dob/:gender/:pas
sword').get(function (req, res) {
    console.log("update function called")
    let id = req.params.id;
    let fname=req.params.fname;
    let lname=req.params.lname;
    let email=req.params.email;
    let phone=req.params.phone;
    let dob=req.params.dob;
    let gender=req.params.gender;
    let password=req.params.password;


    userdetails.updateOne({_id : id},{$set: {firstName :fname, lastName:
lname, phoneNumber: phone, gender: gender, email: email, password: password,
dob: dob}}).then(sup=>{
        console.log(" successfully updated user");
        console.log(sup);
        res.status(200).json({'userUpdate':'successful'});
    }).catch(err=>{
        console.log("update fail");
        res.status(400).send('fail');
    });
});



module.exports = router;
```

server.js

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const cors = require('cors');


const PORT = 4000;
const mongoose = require('mongoose');
```

```
app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.json({limit: "100mb"}));
app.use(bodyParser.urlencoded({limit: "100mb", extended: true,
parameterLimit:500000}));

mongoose.connect("mongodb+srv://admin:admin@cluster0-
8wjef.mongodb.net/iLearning?retryWrites=true&w=majority", {useNewUrlParser:
true});
const connection = mongoose.connection;


connection.once('open', function () {
    console.log('mongoDB database Connections extablished Successfully')
});

//start the server using express
app.listen(PORT, function () {
    console.log("Server is running on PORT: " + PORT);
});

const userDetail = require('./Routes/user.server.route');
 const adminDetail = require('./Routes/lecture.server.route');
app.use('/api/userDetail',userDetail);
 app.use('/api/adminDetail',adminDetail);
```