1.Create a git repository and clone it for changes and publish the changes using git bash.

-Open GitHub website and create a new repository.
-Name and create the repository.
-Copy the GitHub link
-Open Github in the local computer and clone the repository using 'git clone'
  $git clone "<git repository link>"
-Search and open the file in the computer
-cd <filename>
-open a notepad file and enter any html code. Later save and close the file.
-Next all the html file to the repository using the 'git add'
    -$ git add .
-Commit the changes written in the code
    -$ git commit -m "first commit"
-Push all the commands using 'git push'
    -$ git push

2.Create a realtime database in firebase for the student management system and explore the features of Firebase Real Time Database.

Index.html

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Firebase Operations</title>
        <link rel = "stylesheet" href =
"https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min
.css">
        <link rel = "stylesheet" href="styles.css">
    </head>
    <body class = "container">
        <div class = "container2">
            Name: <input type="text" id = "name" class = "input"
autocomplete = "off"><br><br>
            Email: <input type = "email" id = "email" class = "input"
><br><br>
```

```
            Phone: <input type = "number" id = "phone" class = "input"
><br><br>
            <button class = "btn btn-primary"
onclick="insert()">Insert</button>
            <button class = "btn btn-primary"
onclick="read()">Read</button>
            <button class = "btn btn-primary"
onclick="update()">Update</button>
            <button class = "btn btn-primary"
onclick="Delete()">Delete</button>
        </div>
    </body>
    <script
src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"></script>
    <script
src="https://www.gstatic.com/firebasejs/8.4.1/firebase-database.js"></scri
pt>
    <script>
        // Your web app's Firebase configuration
        var firebaseConfig = {
          apiKey: "AIzaSyDUi-HbawhVVsPBM1gUcWLrEcG4uDwpFrU",
          authDomain: "we-exam.firebaseapp.com",
          projectId: "we-exam",
          storageBucket: "we-exam.appspot.com",
          messagingSenderId: "477124477779",
          appId: "1:477124477779:web:46ae7ce1e4f70807ee17a6"
        };
        // Initialize Firebase
        firebase.initializeApp(firebaseConfig);
      </script>
      <script src = "firebaseOperations.js"></script>
</html>
```

firebaseOperations.js

```
var Name,email,phone;
var database = firebase.database().ref('users');
function getData(){
    Name = document.getElementById("name").value;
    email = document.getElementById("email").value;
    phone = document.getElementById("phone").value
}
function read(){
    var phone = document.getElementById("phone").value;
console.log("Read");
database.child(phone).on('value',(snapshot) => {
    obj = snapshot.val()
    if(obj!=null){
        var s=`Name: ${obj['Name']}\n Email: ${obj['Email']}\n Phone:
${obj['Phone']} `;

        alert(s)

    }
    else
        alert('Data not found');
});
}
function insert(){
    getData();
    console.log("Insert");
    database.child(phone).set({Name,email,phone,})
}
function update(){
    getData();
    console.log("Update");
    database.child(phone).set({Name,email,phone,})
}
function Delete(){
    var phone = document.getElementById("phone").value;
    console.log("Delete");
    database.child(phone).set(null);
}
```

Styles.css

```
body {
  background: lightgrey;
}
```

```css
.container2 {
  width: 400px;
  background-color: cadetblue;
  position: absolute;
  left: 40%;
  top: 30%;
  padding: 50px;
  border-bottom: 2px solid grey;
  justify-content: center;
}
.input {
  box-sizing: border-box;
  background: inherit;
  border-top: hidden;
  border-left: hidden;
  border-right: hidden;
  outline: none;
  color: red;
  font-weight: bold;
  font-size: inherit;
}
```

3.Develop an express web application that can interact with a service to perform CRUD operations on student data. (Use Postman)

Index.js

```javascript
const express = require('express');
const users = require('./users');
const fs = require('fs');

const app = express();

app.use(express.json());
app.listen(1111, () => console.log("Server listening"));
app.get('/api/users', (req, res) => {
    res.send(users);
})

app.get('/api/users/:id', (req, res) => {
    const user = users.find((user) => user.userId ===
parseInt(req.params.id));
    if (!user)
        res.status(404).send(`No user exists with the given user id :
${req.params.id}`);
    else
        res.send(user);
})

app.post('/api/users', (req, res) => {
    console.log(req.body);
    users.push(req.body);
    res.send(`User with name ${req.body.name} added succesfully`);
    fun(users);
})

app.put('/api/users/:id', (req, res) => {
    let user = users.find((user) => user.userId ===
parseInt(req.params.id));
    if (!user)
        res.status(404).send(`No user exists with the given user id :
${req.params.id}`)
    else {
        const index = users.indexOf(user);
        user = req.body;
        users.splice(index, 1, user);
        res.send(`User with name ${req.body.name} updated successfully`);
        fun(users);
    }
})

app.delete('/api/users/:id', (req, res) => {
    const user = users.find((user) => user.userId ===
parseInt(req.params.id));
```

```
    if (!user)
        res.status(404).send(`No user exists with the given user id :
${req.params.id}`);
    else {
        const index = users.indexOf(user);
        users.splice(index, 1);
        fun(users);
        res.send(user);
    }
})

const fun = (users) => {
    var s = JSON.stringify(users, null, 2);
    s = "const users = " + s + '\nmodule.exports = users;';
    fs.writeFile('users.js', s, () => {
        console.log('Written successfully');
    });
}
```

Users.js

```
const users = [
  {
    "userId": 1,
    "name": "Pranay Kumar",
    "age": 21,
    "contact": 9381244438
  },
  {
    "userId": 2,
    "name": "Nikhil Chandra",
    "age": 20,
    "contact": 8394568236
  },
  {
    "userId": 3,
    "name": "Ritin",
    "age": 21,
    "contact": 862145638
  },
  {
    "userId": 4,
    "name": "Prasuna",
    "age": 20,
    "contact": 8569456328
  },
  {
    "userId": 5,
    "name": "Chotu",
    "age": 19,
    "contact": 8934587329
  },
  {
    "userId": 7,
    "name": "Yelchu",
    "age": 69
  }
]
module.exports = users;
```

4.For the above application create authorized end points using JWT (JSON Web Token).

Index.js

```
const express = require('express');
const users = require('./users');
const fs = require('fs');
const jwt = require("jsonwebtoken");
const app = express();

app.use(verifyToken);
app.use(express.json());
app.listen(1111, () => console.log("Server listening"));

app.get('/api/users',(req, res) => {
   res.send(users);
})
```

```
app.get('/api/users/:id', (req, res) => {
    const user = users.find((user) => user.userId === parseInt(req.params.id));
    if (!user)
        res.status(404).send(`No user exists with the given user id : ${req.params.id}`);
    else
        res.send(user);
})

app.post('/api/users', (req, res) => {
    console.log(req.body);
    users.push(req.body);
    res.send(`User with name ${req.body.name} added succesfully`);
    fun(users);
})

app.put('/api/users/:id', (req, res) => {
    let user = users.find((user) => user.userId === parseInt(req.params.id));
    if (!user)
        res.status(404).send(`No user exists with the given user id : ${req.params.id}`)
    else {
        const index = users.indexOf(user);
        user = req.body;
        users.splice(index, 1, user);
        res.send(`User with name ${req.body.name} updated successfully`);
        fun(users);
    }
})

app.delete('/api/users/:id', (req, res) => {
    const user = users.find((user) => user.userId === parseInt(req.params.id));
    if (!user)
        res.status(404).send(`No user exists with the given user id : ${req.params.id}`);
    else {
        const index = users.indexOf(user);
        users.splice(index, 1);
        fun(users);
        res.send(user);
    }
})

const fun = (users) => {
    var s = JSON.stringify(users, null, 2);
    s = "const users = " + s + '\nmodule.exports = users;';
    fs.writeFile('users.js', s, () => {
        console.log('Written successfully');
    });
}

function verifyToken(req, res, next) {
    if(req.url == '/api/login')
    {
        next();
        return;
    }
    const bearerHeader = req.headers["authorization"];
    if (typeof bearerHeader !== "undefined") {
        const bearerToken = bearerHeader.split(" ")[1];
        req.token = bearerToken;
        next();
    } else {
        res.sendStatus(403);
    }
}

app.get("/api/login", (req, res) => {
    const user = {
        id: 1,
        username: "pranay",
        email: "pranay@gmail.com"
    };
    jwt.sign({ user}, "secretkey", (err, token) => {
        res.json({token});
    });
});
```

Users.js

```
const users = [
  {
    "userId": 1,
    "name": "Pranay Kumar",
    "age": 21,
    "contact": 9381244438
  },
  {
    "userId": 2,
    "name": "Nikhil Chandra",
    "age": 20,
    "contact": 8394568236
  },
  {
    "userId": 3,
    "name": "Ritin",
    "age": 21,
    "contact": 862145638
  },
  {
    "userId": 4,
    "name": "Prasuna",
    "age": 20,
    "contact": 8569456328
  },
  {
    "userId": 5,
    "name": "Chotu",
    "age": 19,
    "contact": 8934587329
  }
]
module.exports = users;
```

5.Create an angular application for the student management system. Include necessary pages.

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Student</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

```html
<body>
  <app-root></app-root>
</body>
</html>
```

App.component.css

```css
*{
    font-size: large;
}
.container{
    width: fit-content;
    background-color: yellow;
    padding: 50px;
    margin: 0 auto;
}
.input{
    box-sizing: border-box;
    background: inherit;
    border-top: hidden;
    border-left: hidden;
    border-right: hidden;
    outline: none;
    color: red;
    font-weight: bold;
    font-size: inherit;
}
button{
    background-color: lightblue;
    border-radius: 10px;
    padding: 5px;
    margin: 10px;
}
```

App.component.html

```html
<div class="container">
  Name: <input type="text"  [(ngModel)]="name"><br><br>
  Email: <input type = "email"  [(ngModel)]="email"><br><br>
  Phone: <input type = "number" [(ngModel)]="phone"><br><br>
  <button (click)="insert()">Insert</button>
  <button (click)="read()">Read</button>
  <button (click)="update()">Update</button>
  <button (click)="delete()">Delete</button>
</div>
<div>
  <p>{{name1}}</p>
  <p>{{email1}}</p>
  <p>{{phone1}}</p>
</div>
```

App.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'student';
  data: userData[]=[];
  name: string="";
  email:string="";
  phone:string="";
  name1: string="";
  email1:string="";
  phone1:string="";
  js:any;
  clear(){
    this.name1 = "";
    this.phone1 = "";
    this.email1 = "";
  }
  insert(){
    this.clear();
    const user = this.data.find((user) => user.phone == this.phone);
    if(!user)
```

```
      this.data.push({name:this.name,email:this.email,phone:this.phone});
    console.log(this.data);
  }
  read(){
    const user:any = this.data.find((user) => user.phone == this.phone);
    if(user){
      this.name1 = user.name;
      this.phone1 = user.phone;
      this.email1 = user.email;
    }
  }
  update(){
    this.clear();
    const user:any = this.data.find((user) => user.phone == this.phone);
    if(user){
      const index = this.data.indexOf(user);
      this.js = {name:this.name,email:this.email,phone:this.phone};
      console.log(this.js);
      this.data.splice(index, 1, this.js);
    }
    console.log(this.data);
  }
  delete(){
    this.clear();
    const user:any = this.data.find((user) => user.phone == this.phone);
    if(user){
      const index = this.data.indexOf(user);
      this.data.splice(index,1);
    }
    console.log(this.data);
  }

}
class userData{
  name:string="";
  email:string="";
  phone:string="";
}
```

6.Create a service in angular that fetches the weather information from openweathermap and the display the current and historical weather information using graphical representation using chart.js

App.component.html

```
City: <input type = "text"  [(ngModel)]="cityName"><br>
<button (click)="loadData()">Get Weather</button>
<div *ngIf="load">
  <h3>{{day[0]}}:  {{temp[0]}}C</h3>
  <h3>{{day[1]}}:   {{temp[1]}}C</h3>
  <h3>{{day[2]}}:  {{temp[2]}}C</h3>
  <h3>{{day[3]}}:  {{temp[3]}}C</h3>
  <h3>{{day[4]}}:  {{temp[4]}}C</h3>
</div>
```

App.component.ts

```
import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'weather';
  load = false;
  city:any;
  cityName:string = '';
  temp:any[] = [];
  day:any[] = [];
  constructor(private h:HttpClient){}
  loadData(){
```

```
this.h.get(`https://api.openweathermap.org/data/2.5/forecast?q=${this.city
Name}&appid=5aac677d8bdf1b1c517aa7041a47aab8&units=metric`).subscribe((res
)=>{
      this.load=true;
      this.city =  res;
      for(var i=0;i<40;i+=8){
        var num = this.city.list[i].main.temp;
        var date = this.city.list[i].dt_txt;
        var day = date.split(" ")[0];
        this.temp.push(num);
        this.day.push(day);
      }
    })
  }
}
```

App.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http'
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

7.Develop an angular application that displays employees information and transform the data in the required form using pipes.

Custom.components.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-custom',
  templateUrl: './custom.component.html',
  styleUrls: ['./custom.component.css']
})
export class CustomComponent implements OnInit {

  constructor() { }
  data:any=[
    {
      name:"Pranay",age:21,roll:"18B81A0589",gpa:8.72
    },
    {
      name:"Chintu",age:20,roll:"18B81A0389",gpa:4.999
    },
    {
      name:"Juniper",age:22,roll:"18B81A0485",gpa:6.15
    }


  ]
  ngOnInit(): void {
  }

}
```

Custom.component.html

```
<div class="container">
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Roll Number</th>
                <th>Branch</th>
                <th>GPA</th>
                <th>Pass/Fail</th>
            </tr>
        </thead>

        <tbody>
            <tr *ngFor="let i of data">
                <td>{{i.name}}</td>
                <td>{{i.age}}</td>
                <td>{{i.roll}}</td>
                <td>{{i.roll|branch}}</td>
                <td>{{i.gpa}}</td>
                <td>{{i.gpa|pass}}</td>
            </tr>
        </tbody>
    </table>
</div>
```

Index.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Demo</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>

<body style="background-color: darkviolet;">
    <app-root></app-root>
</body>

</html>
```

Custom.components.css

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat&display=swap');
th {
   /* border: 2px solid red; */
   border-collapse: collapse;
   color: rgb(61, 149, 226);
}

td,
th {
   padding: 10px;
}

table {
   margin: 50px auto;
   padding: 100px;
   font-family: 'Monteserrat', sans-serif;
   /* border: 5px solid goldenrod; */
   border-collapse: collapse;
   background-color: darkviolet;
}

td {
   padding-left: 20px;
   padding-right: 20px;
   color: aliceblue;
}
```

Branch.pipe.ts

```
import { Pipe, PipeTransform } from "@angular/core";

@Pipe({name:"branch"})
export class BranchPipe implements PipeTransform{

   transform(value: any) {
      if(value[7]=='5')
         return "CSE";
      else if(value[7]=='3')
         return 'MECH';
```

```
        else if(value[7]=='4')
            return 'ECE';
        return "Invalid";
    }
}
```

Pass.pipe.ts

```
import { Pipe, PipeTransform } from "@angular/core";

@Pipe({name:"pass"})
export class PassPipe implements PipeTransform{

    transform(value: any) {
        if(value>5)
            return "Pass";
        return "Fail";
    }
}
```

8.Develop a react application for the student management system. Include necessary pages.

app.js :-

```
import React, { Component } from 'react';
import { BrowserRouter as Router, Route } from "react-router-dom";
import Students from './components/DashBoard/StudentData/students';
import AddStudents from './components/DashBoard/StudentData/Addstudent';
import LogOut from './components/auth/LogOut';
import Details from './components/DashBoard/StudentData/Details';
import NavBar from './components/DashBoard/navbar/navBar';
import UserProfile from './components/auth/UserProfile';
class App extends Component {

  render() {
    return (
      <div>
        <Router>
        <div className="App">
          <NavBar/>
          <Route exact path="/students" component={Students} />
          <Route exact path="/Addstudent" component={AddStudents} />
          <Route exact path="/Addstudent/:id" component={AddStudents} />
          <Route exact path="/LogOut" component={LogOut} />
          <Route exact path="/Details/:id" component={Details} />
          <Route exact path="/UserProfile" component={UserProfile} />
        </div>
      </Router>
      </div>
    );
  }
}

export default App;
```

Addstudent.js:-

```
import React, { Component } from 'react';
import * as firebase from 'firebase';
import '../../../config/fb';
import Input from '../../../UICom/Input';
import Button from '../../../UICom/Button';

class AddStudents extends Component {
    constructor() {
        super();
        this.state = {
            StudentName: "",
            StudentFName: "",
            StudentAge: '',
            StudentGender: '',
            editId: null,
            edit: false,
            User: {},
            studentPervData: null


        }
        this.ref = firebase.database().ref();
    }
    componentDidMount(){
        console.log({AddStudents: "componentDidMount"})
        firebase.auth().onAuthStateChanged((user) => {
            if (user) {
                this.setState({User: user})
            } else {
                console.log({AddStudents: "current user null"})
            }
         });
         this.getStudentsData()
    }

    onEdit = (StudentData, id) => {
        const Editstudent = StudentData.find((stu) =>{
            return stu.StudentID === id
        })
        if (Editstudent){
                this.setState({
                editId: id,
                edit: "Edit Student",
                StudentName: Editstudent.name,
                StudentFName: Editstudent.fname,
                StudentAge: Editstudent.age,
                StudentGender: Editstudent.gender,
             })
        }

    }
    getStudentsData = () => {
        this.ref.child(`Student`).once("value", (snapshot) => {
            const data = snapshot.val();
            const TempArr = [];
            for (let key in data) {
                TempArr.push({ StudentID: key, name: data[key].name, fname: data[key].fname, age:
data[key].age, gender: data[key].gender });
            }
            const EditID = this.props.match.params.id;
            if(EditID){
                this.onEdit(TempArr, EditID)
            }
        })
    }
    onAdd = (event) => {
        event.preventDefault();
        const { StudentName, StudentFName, StudentAge, StudentGender, editId } = this.state;
        if (StudentName === '' || StudentFName === '' || StudentGender === '' || StudentAge === '')
{
            return
        }
        else if (editId !== null) {
            this.ref.child(`Student/${this.state.editId}`).update({ name: StudentName, fname:
StudentFName, age: StudentAge, gender: StudentGender });
        }
        else {
            this.ref.child(`Student`).push({ name: StudentName, fname: StudentFName, age:
StudentAge, gender: StudentGender })
        }
        this.setState({ StudentName: '',
        StudentFName: '', StudentAge: '', StudentGender: '',
        edit: false,
        editId: '', })
        setTimeout(()=> {
            this.props.history.push('/students');
        },1000)
    }
    componentWillUnmount(){
        console.log({AddStudents: "componentwillUnmount"})
    }
    whenChange = (event) => {
        const { name, value } = event.target;
        this.setState({ [name]: value })

    }
    render() {
        return (
            <div className="container">
                <div className="center teal darken-3 white-text">
                    <h4>Add Student Form</h4>
                </div>
                <div className="teal lighten-5">
                    <form onSubmit={this.onAdd}>
                        <Input v={this.state.StudentName} oc={this.whenChange}  t="text" f='name'
d='name' l='Name' n="StudentName" />
                        <Input v={this.state.StudentFName} oc={this.whenChange}  t="text" f='fname'
d='fname' l='Father Name' n="StudentFName" />
                        <Input v={this.state.StudentAge} oc={this.whenChange}  t="number" f='age'
d='age' l='Age' n="StudentAge" />
                        <Input v={this.state.StudentGender} oc={this.whenChange}  t="text" f='gender'
d='gender' l='Gender' n="StudentGender" />
                        {this.state.edit ? (<Button cn="btn-small right" t={this.state.edit} />
                        ) : (
                        <Button cn="btn-small right" t="Add Student" />)}
                    </form>
                </div>
            </div>
        )
    }
```

```
}
export default AddStudents;

navbar.js :-
import React,{Component} from 'react';
import {NavLink, Link, withRouter} from "react-router-dom";
import * as firebase from 'firebase';
import "../../../config/fb";
import LogIn from '../../auth/LogIn';
import SignedInLinks from './SignedInLinks';
import Drawer from '@material-ui/core/Drawer';

import './navBar.css'

class NavBar extends Component{
  constructor() {
    super()
    this.state = {
     User: null,
     signIn: false,
     name: "",
     left: false,
     status: "",

    }
    this.ref = firebase.database().ref()
   }
   componentDidMount = () => {
    this.authListener();
   }
   authListener = () => {
    firebase.auth().onAuthStateChanged((user) => {
     if (user) {
      console.log("Current User Signed In");
      this.setState({User: user})
      this.getStatus(user.uid)
     } else {
      this.setState({User: null})
      console.log("No Signed In user")
     }
    });
   }
   getStatus = (uid) => {
    this.ref.child(`Status${uid}`).on("value", (snapshot)=> {
          const s = snapshot.val()
          let tems = ""
          for(let key in s){
            tems += s[key].status
          }
        this.setState({status: tems})
        })
      }
      changeName = () => {
        const user = this.state.User
        const name = user.displayName;
        if(name){
          let char = name.slice(0,1);
            let index = null
            let a = [...name]
            for(let i = 0; i < a.length; i++){
              if(a[i] === " "){
                index += a.indexOf(" ");
                char += a[index+1]
                }
            }
            return char;
        }
        else{
          const email = user.email;
          let char = email.slice(0,1);
          return char;
        }
      }
      toggleDrawer = (open) => () => {
        this.setState({
          left : open,
        });
      };
    render(){
    const sideList = (
    <div className="list_width">
      <ul className="collection with-header">
      <li className="collection-header teal">
      <NavLink to="/UserProfile"><h5 className="white-text">{this.state.User ?
(this.state.User.displayName ? (this.state.User.displayName
        ) : (this.state.User.email)
        ) : (null)}</h5>
      </NavLink>
      </li>
      <li className="collection-item"><NavLink className="grey-text" exact
activeClassName="black-text" to="/students">Students</NavLink></li>
      {this.state.status === "teacher" ? (<li className="collection-item"><NavLink
className="grey-text" exact activeClassName="black-text" to='/Addstudent'>Add
Students</NavLink></li>) : (null)}
      <li className="collection-item"><NavLink className="grey-text" exact
activeClassName="black-text" to='/LogOut'>Log out</NavLink></li>
      </ul>
      </div>
    );
        return (
            <div>
            {this.state.User ? (
        <nav className="nav-wrapper teal darken-4">
        <div className="container">
        <span onClick={this.toggleDrawer(true)} className="btn-small btn-floating transparent
hide-on-large-only">
        <i className="material-icons">menu</i>
        </span>
        &nbsp;
        &nbsp;
        &nbsp;
        <span className="flow-text teal darken-4 hide-on-large-only">Student Management
System</span>
        <Drawer open={this.state.left} onClose={this.toggleDrawer(false)}>
          <div onClick={this.toggleDrawer(false)}>
            {sideList}
          </div>
        </Drawer>
        <span className="brand-logo hide-on-med-and-down">Student Management
System</span>
        <ul className="right hide-on-med-and-down">
        <li><Link to="/students">Students</Link></li>
        <SignedInLinks s={this.state.status} name={this.changeName()}/>
        </ul>
        </div>
        </nav>) : (<LogIn />)}
        </div>
        )
    }
}
export default withRouter(NavBar);
```