Bengaluru, Karnataka, India



AN INTERNSHIP REPORT ON

“ SGPA & CGPA CALCULATOR ”

## BACHELOR OF ENGINEERING

In

#### COMPUTER SCIENCE AND ENGINEERING

##### Submitted by: CHINMAYEE S (1RR14EE350)



**Rajarajeshwari college of engineering**

(Affiliated to Visvesvaraya Technological University and Approved by AICTE, New Delhi)Accredited By NAAC with ‘A’ Grade

ISO 9001-2015 and 14001-2015 certified Institute

**BengAluru, KArnAtAkA 560057**

##### 2021-2022

**ABOUT THE COMPANY**

CST is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At CST , we believe that service and quality is the key to success

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

* Development - We develop responsive, functional and super-fast websites.
* We keep User Experience in mind while creating websites. A website should load quickly and shouldbe accessible even on a small view-port and slow internet connection.
* Mobile Application - We offer a wide range of professional Android, iOS& Hybrid app development services for our global clients, from a start up to a large enterprise.
* Design - We offer professional Graphic design, Brochure design & Logo design. We are expertsin crafting visual content to convey the right message to the customers.
* Consultancy - We are here to provide you with expert advice on your design and development requirement.
* Videos - We create a polished professional video that impresses your audience.

# Table of Contents

#### Contents Page No.

**Table of Contents I**

[Overview of the project 1](#_TOC_250003)

About 2

[Tools used 3](#_TOC_250002)

[Implementation 4-25](#_TOC_250001)

[Snapshots 26-27](#_TOC_250000)

Biblography 28

##### (I)

### OVERVIEW OF THE PROJECT

**Project Name**: SGPA & CGPA Calculator

This project is based on Web Development and its Applications. The main objective of this project is to learn the implementation of HTML, CSS and JavaScript at frontend and PHP, MySQL database at backend, The webpage of this project is created using HTML and styling of the webpage is done using CSS.

It is a web application that simplifies the task of storing a users results . The system is flexible to be used and reduces the need of frequently searching and accessing an individuals marks. The system is developed to provide an easy means for storing semester wise results. Individuals have to login and enter their marks in a form.They can then access the result and print it in a PDF format.

1. **USER MODULE**

This module is mainly dedicated to the candidates who are looking to store their result. They can log in into the system, via the credentials provided to them & once they are in, they have to enter their data by filling an online form . They can download the result from the same module, in PDF file format.

1. **ADMIN MODULE**

This module is maintained by the admin and only he/she can manage and have access to every account. Admin can add or delete the users as well. They can modify the functionalities of the system, too. Admin can also add users on the fly to the application to make the application full of new functionalities.

**SGPA & CGPA Calculator**

**Overview & Details:**

SGPA & CGPA Calculator is solution mainly dedicated to the candidates who are looking to store their result. They can log in into the system, via the credentials provided to them & once they are in, they have to enter their data by filling an online form . They can download the result from the same module, in PDF file format.

**Key details on SGPA and CGPA:**

SGPA, which stands for Semester Grade Point Average is an evaluation method that highlights the semester wise performance of the student. It can be calculated by simply adding all the credit points awarded for the subjects and then dividing it by the total credits allotted to that semester.

For example, in a total of 3 subjects, you scored the following grade points:

Subject 1: 8

Subject 2: 6

Subject 3: 7

The total credit for each subject is 10. 1.

Now, first we will multiply, the grade point with the total credit point for each subject:

Subject 1: 8\*10 = 80

Subject 2: 6\*10 = 60

Subject 3: 7\*10 = 70 2.

To calculate SGPA here, you need to add all these grade points and then divide it by the total credits, i.e.

Total grade points: 210

Total credits: 30 To get SGPA,

divide grade points by total credits,

=210/30

=7 SGPA

CGPA refers to the cumulative grade point average which literally translates to the sum total of all your credit points. This system helps in assessing the overall academic performance of a student. Although the evaluation criteria may vary from one country to another, the CGPA system is among the most common evaluation way in most professional/technical courses

CGPA= (SGPAs of All semesters in an academic year)/ Number of semesters

Thus, by adding up all the SGPAs you have got in an academic year by the total number of semesters, you will find CGPA from SGPA.

For example,

Suppose that you scored 7 SGPA and 9 SGPA in your two semesters.

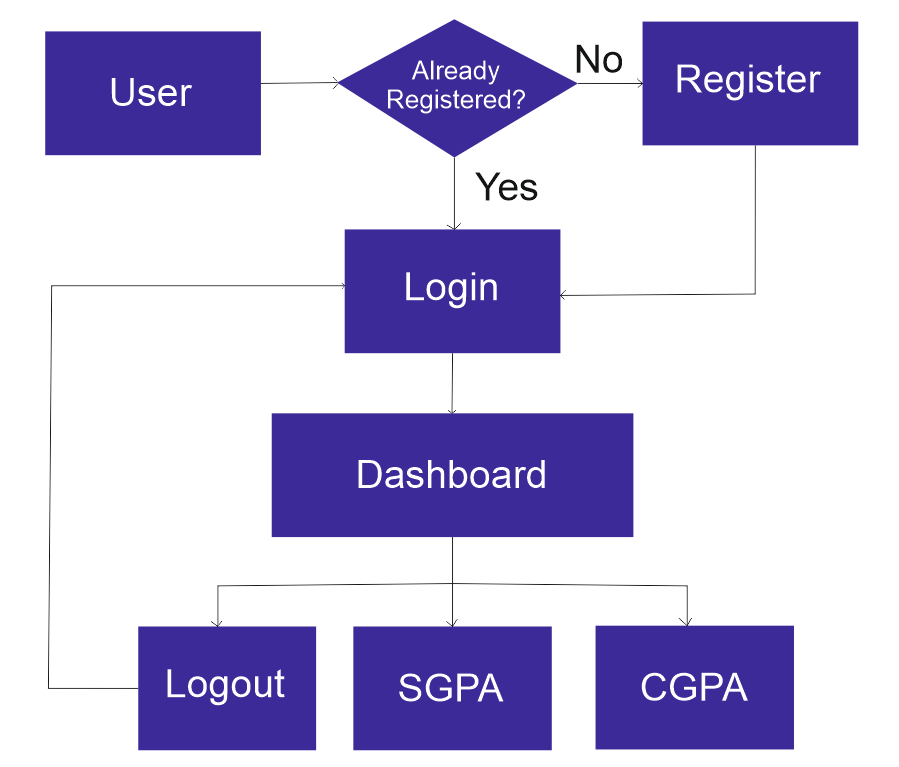
1. First you need to add both these SGPAs.

2. Then, divide the total SGPA with the number of semesters, i.e. 2.

Your CGPA would be:

CGPA = SGPA of all semesters in a year/Number of semesters = (7+9)/2 = 8 CGPA

**Flow Diagram:**



### TOOLS USED

**Software Requirements**

* + Visual Studio Cod.
  + Google Chrome or Microsoft Edge of latest version.
  + Front End: HTML, CSS, JS, React
  + Backend : Php, MySQL, Xampp
  + Linux 7.1 or Windows XP/7/8/10 OS or Mac OS

**Hardware Requirements**

* + Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).
  + Monitor with a refresh rate of at least 40Hz for a smooth GUI experience (optional).

# IMPLEMENTATION

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <link rel="stylesheet" type="text/css" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css" />

    <meta

      name="description"

      content="SGPA & CGPA Notes"

    />

    <title>SGPA & CGPA Notes</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './app';

import 'bootstrap/dist/css/bootstrap.min.css';

import 'react-toastify/dist/ReactToastify.css';

import { BrowserRouter } from "react-router-dom";

ReactDOM.render(

  <BrowserRouter>

    <App />

  </BrowserRouter>

  ,

  document.getElementById('root')

);

body {

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

    sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  width: 100%;

  height: 100%;

  background-color: rgba(128, 128, 128, 0) !important;

  background-image: url("./assests/bg.jpg");

  background-repeat: no-repeat;

  background-position: 50%;

  background-size: cover;

  background-attachment: fixed;

  width: 100%;

  height: 100%;

  top: 0;

  left: 0;

  z-index: -100;

}

code {

  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

    monospace;

}

App.js

import React, { Component } from 'react';

import Navigation from './navigation/navigation';

import { ToastContainer } from 'react-toastify';

export default class App extends Component {

  render() {

    return (

      <div id='#wrapper' className='container'>

        <Navigation />

        <ToastContainer autoClose={2000} />

      </div>

    )

  }

}

Login.js

import React, { useState } from 'react';

import DocumentTitle from 'react-document-title';

import { useNavigate } from "react-router-dom";

import Form from '../components/form';

import Card from '../components/card';

import LoginItems from './loginItems.json';

import Alert from '../components/alert';

import { toast } from 'react-toastify';

import './login.scss'

export default function Login() {

    let navigate = useNavigate();

    const [formValidationErrors, setFormValidationErrors] = useState(null);

    const handleLoginClick = (validatedData) => {

        setFormValidationErrors(null)

        const tempData = {

            username: validatedData.username,

            password: validatedData.password

        }

        const signInData = JSON.parse(localStorage.getItem('signInData'))

        if (signInData === null) {

            setFormValidationErrors("Username or password does not exists!!!")

        } else {

            const result = signInData.filter(data => data.username === validatedData.username && data.password === validatedData.password)

            if (result.length === 0) {

                setFormValidationErrors("Username or password does not exists!!!")

            } else {

                signInData.push(tempData)

                localStorage.setItem(

                    'loginUserData',

                    validatedData.username)

                toast("Login In Successfull")

                navigate('/dashboard');

                window.location.reload()

            }

        }

    }

    return (

        <DocumentTitle title='Login'>

            <div className='login'>

                <div className='row justify-content-center'>

                    <div className='col-lg-4 col-md-12'>

                        <Card title="Login">

                            {formValidationErrors === null ? null :

                                <Alert variant="danger" content={formValidationErrors} onClose={setFormValidationErrors} />

                            }

                            <Form data={LoginItems} handleSubmitClick={handleLoginClick} />

                            <div className='signup'><a href='/signup'>Click here</a> to add new profile</div>

                        </Card>

                    </div>

                </div>

            </div>

        </DocumentTitle>

    )

}

.login{

    margin-top: 100px;

}

.login .signup{

    margin-top: 10px;

}

{

    "col": "12",

    "items": [

        {

            "id": "username",

            "label": "Username",

            "type": "text",

            "value": "",

            "required": true

        },

        {

            "id": "password",

            "label": "Password",

            "type": "password",

            "value": "",

            "required": true

        },

        {

            "id": "login",

            "type": "button",

            "label": "Login",

            "value": "",

            "buttonType": "submit"

        }

    ]

}

Signup.js

import React, { useState } from 'react';

import DocumentTitle from 'react-document-title';

import Form from '../components/form';

import Card from '../components/card';

import SignupItems from './signupItems.json';

import Alert from '../components/alert';

import { useNavigate } from "react-router-dom";

import './signup.scss'

import { toast } from 'react-toastify';

export default function Signup() {

    let navigate = useNavigate();

    const [formValidationErrors, setFormValidationErrors] = useState(null);

    const handlesignupClick = (validatedData) => {

        setFormValidationErrors(null)

        const tempData = {

            username: validatedData.username,

            password: validatedData.password

        }

        const signInData = JSON.parse(localStorage.getItem('signInData'))

        if (signInData === null) {

            localStorage.setItem(

                'signInData',

                JSON.stringify(

                    [tempData]))

            toast("Sign In Successfull")

            navigate('/login');

        } else {

            const result = signInData.filter(data => data.username === validatedData.username)

            if (result.length > 0) {

                setFormValidationErrors("Username already exists!!!")

            } else {

                signInData.push(tempData)

                localStorage.setItem(

                    'signInData',

                    JSON.stringify(

                        signInData))

                toast("Sign In Successfull")

                navigate('/login');

            }

        }

    }

    return (

        <DocumentTitle title='signup'>

            <div className='signup'>

                <div className='row justify-content-center'>

                    <div className='col-lg-4 col-md-12'>

                        <Card title="Sign In">

                            {formValidationErrors === null ? null :

                                <Alert variant="danger" content={formValidationErrors} onClose={setFormValidationErrors} />

                            }

                            <Form data={SignupItems} handleSubmitClick={handlesignupClick} />

                            <div className='signup'><a href='/login'>Click here</a> to login</div>

                        </Card>

                    </div>

                </div>

            </div>

        </DocumentTitle>

    )

}

.signup{

    margin-top: 100px;

}

.signup .signup{

    margin-top: 10px;

}

{

    "col": "12",

    "items": [

        {

            "id": "username",

            "label": "Username",

            "type": "text",

            "value": "",

            "required": true

        },

        {

            "id": "password",

            "label": "Password",

            "type": "password",

            "value": "",

            "required": true

        },

        {

            "id": "signup",

            "type": "button",

            "label": "Sign In",

            "value": "",

            "buttonType": "submit"

        }

    ]

}

Alert.js

import React, { Component } from 'react';

import { Alert } from 'react-bootstrap';

export default class alert extends Component {

    render() {

        const { variant, content, onClose } = this.props;

        return (

            <Alert variant={variant} onClose={() => onClose(null)} dismissible>

                {content}

            </Alert>

        )

    }

}

Card.js

import React, { Component } from 'react';

import { Card } from 'react-bootstrap';

import "./card.scss"

export default class card extends Component {

    render() {

        const { title, children } = this.props;

        return (

            <Card >

                <Card.Body>

                    <Card.Title>{title}</Card.Title>

                    {children}

                </Card.Body>

            </Card>

        )

    }

}

.card-body {

    flex: 1 1 auto;

    padding: 2rem 2rem;

}

.card {

    border: none !important;

    box-shadow: 1px 0 20px rgb(0 0 0 / 5%);

    background-color: rgba(255, 255, 255, 0.90);

}

.card .card-title {

    font-size: 30px !important;

    color: #007bff;

    margin-bottom: 20px;

}

Form.js

import React, { Component } from 'react';

import { Form, Row, Col, Button } from 'react-bootstrap';

import "./form.scss"

export default class form extends Component {

    constructor(props) {

        super(props);

        this.state = {

            validated: false,

            data: null,

            validationData: {}

        }

    }

    handleOnChange = (id, value) => {

        this.setState({

            ...this.state, validationData: {

                ...this.state.validationData, [id]: value

            }

        })

    }

    handleSubmit = (event) => {

        event.preventDefault();

        this.setState({ ...this.state, validated: true }, () => {

            this.props.handleSubmitClick(this.state.validationData)

        })

    };

    updateValidationData = () => {

        const { data } = this.state;

        let tempValidationData = {}

        data.items.map(item => (

            tempValidationData[item.id] = item.value ? item.value : ""

        ))

        this.setState({ ...this.state, validationData: tempValidationData })

    }

    componentDidMount = () => {

        this.setState({ ...this.state, data: this.props.data }, () => {

            this.updateValidationData()

        })

    }

    render() {

        const { validated, data, validationData } = this.state;

        return (

            <>

                {data === null ? null :

                    <Form noValidate validated={validated} onSubmit={this.handleSubmit}>

                        <Row>

                            {data.items.map(item => (

                                <Form.Group key={item.id} as={Col} md={data.col} >

                                    {item.type === "button" ? null :

                                        <Form.Label>{item.label}</Form.Label>}

                                    {item.type === "textarea" ?

                                        <Form.Control

                                            id={item.id}

                                            as="textarea"

                                            rows={item.rows}

                                            placeholder={item.placeholder ? item.placeholder : ""}

                                            value={validationData[item.id] ? validationData[item.id].value : ""}

                                            disabled={item.disabled ? item.disabled : false}

                                            onChange={(event) => this.handleOnChange(item.id, event.target.value)}

                                        /> :

                                        item.type === "button" ?

                                            <Button

                                                id={item.id}

                                                type={item.buttonType}

                                                disabled={item.disabled ? item.disabled : false}

                                            >

                                                {item.label}

                                            </Button> :

                                            <Form.Control

                                                id={item.id}

                                                required={item.required ? item.required : false}

                                                type={item.type}

                                                placeholder={item.placeholder ? item.placeholder : ""}

                                                value={validationData[item.id] ? validationData[item.id].value : ""}

                                                disabled={item.disabled ? item.disabled : false}

                                                onChange={(event) => this.handleOnChange(item.id, event.target.value)}

                                            />}

                                    <Form.Control.Feedback type="invalid">{item.feedback ? item.feedback : "This field is required."}</Form.Control.Feedback>

                                </Form.Group>

                            ))}

                        </Row>

                    </Form>}

            </>

        )

    }

}

.form-label {

    color: black;

    margin-bottom: 10px;

    font-size: 18px;

}

.form-control {

    background-color: rgba(255, 255, 255, 0);

    border: 1px solid black;

    margin-bottom: 10px;

}

.form-control:focus {

    background-color: rgba(255, 255, 255, 0);

    border: 1px solid #007bff;

    box-shadow: 1px 0 20px rgb(0 0 0 / 5%);

}

.btn-primary {

    background-color: #007bff ;

    border-color: #007bff;

    margin-top: 10px;

    font-size: 18px;

}

Dashboard.js

import React, { useState, useEffect, useRef } from 'react'

import DocumentTitle from 'react-document-title';

import { Accordion, Table, Form, Button } from 'react-bootstrap';

import Select from 'react-select';

import "./dashboard.scss";

import html2canvas from "html2canvas";

import { jsPDF } from "jspdf";

import { useNavigate } from "react-router-dom";

export default function Dashboard() {

    const [userId, setUserId] = useState(null)

    const [semesterSelected, setSemesterSelected] = useState(null)

    const [userDetails, setUserDetails] = useState({})

    const [semesterData, setSemisterData] = useState({})

    const [semesterMetaData, setSemesterMetaData] = useState(null)

    const [userCgpa, setUserCgpa] = useState(null)

    const [userMetaData, setUserMetaData] = useState({})

    const [updateClicked, setUpdateClicked] = useState(false)

    let navigate = useNavigate()

    const SemesterOptions = [

        { label: "Semester 1", value: "Semester1" },

        { label: "Semester 2", value: "Semester2" },

        { label: "Semester 3", value: "Semester3" },

        { label: "Semester 4", value: "Semester4" },

        { label: "Semester 5", value: "Semester5" },

        { label: "Semester 6", value: "Semester6" },

        { label: "Semester 7", value: "Semester7" },

        { label: "Semester 8", value: "Semester8" },

    ]

    const inputRefSgpa = useRef(null);

    const inputRefCgpa = useRef(null);

    const printDocument = (ref) => {

        html2canvas(ref.current, {

            useCORS: true,

            allowTaint: true,

            scrollY: -window.scrollY,

            dpi: 1200

        }).then(canvas => {

            const image = canvas.toDataURL('image/jpeg', 100);

            const doc = new jsPDF('l', 'px', 'a4');

            const pageWidth = doc.internal.pageSize.getWidth();

            const pageHeight = doc.internal.pageSize.getHeight();

            const widthRatio = pageWidth / canvas.width;

            const heightRatio = pageHeight / canvas.height;

            const ratio = widthRatio > heightRatio ? heightRatio : widthRatio;

            const canvasWidth = (canvas.width - 20) \* ratio;

            const canvasHeight = (canvas.height - 20) \* ratio;

            const marginX = (pageWidth - canvasWidth) / 2;

            const marginY = (pageHeight - canvasHeight) / 2;

            doc.addImage(image, 'JPEG', marginX, marginY, canvasWidth, canvasHeight);

            doc.output('dataurlnewwindow');

        });

    };

    const getGradePoint = (marks) => {

        if (marks >= 90) {

            return 10;

        } else if (marks < 90 && marks >= 80) {

            return 9;

        } else if (marks < 80 && marks >= 70) {

            return 8;

        } else if (marks < 70 && marks >= 60) {

            return 7;

        } else if (marks < 60 && marks >= 50) {

            return 6;

        } else if (marks < 50 && marks >= 45) {

            return 5;

        } else if (marks < 45 && marks >= 40) {

            return 4;

        } else {

            return 0;

        }

    }

    const getMetaData = (semData) => {

        let sum = 0

        let totalCredits = 0

        Object.keys(semData).map(item => {

            const marks = semData[item].marks === "" ? 0 : parseFloat(semData[item].marks)

            const credit = semData[item].credit === "" ? 0 : parseFloat(semData[item].credit)

            const gradePoint = getGradePoint(marks)

            sum += credit \* gradePoint

            totalCredits += credit

        })

        const sgpa = sum > 0 ? (sum / totalCredits).toFixed(2) : 0

        const percentage = sgpa ? ((sgpa - 0.75) \* 10).toFixed(2) : 0

        return { sgpa: String(sgpa), percentage: String(percentage) }

    }

    const handleUpdate = () => {

        setUserDetails((previousState) => ({

            ...previousState,

            [semesterSelected.value]: semesterData

        }))

        setUpdateClicked(true)

        setUserMetaData((previousState) => ({

            ...previousState,

            [semesterSelected.value]: semesterMetaData

        }))

    }

    const handleClickAdd = () => {

        const subCount = Object.keys(semesterData).length + 1

        setSemisterData((previousState) => ({

            ...previousState,

            [subCount]: {

                "subjectCode": "",

                "credit": "",

                "marks": ""

            }

        }))

    }

    const handleClickMinus = () => {

        const subCount = Object.keys(semesterData).length - 1

        if (subCount === 0) {

            setSemisterData({

                "1": {

                    "subjectCode": "",

                    "credit": "",

                    "marks": ""

                }

            })

        } else {

            setSemisterData((previousState) => {

                const newData = { ...previousState }

                delete newData[Object.keys(semesterData)[subCount]]

                return newData;

            })

        }

    }

    useEffect(() => {

        setSemesterMetaData(getMetaData(semesterData))

    }, [semesterData])

    const updateUserDetails = () => {

        const userDetails = JSON.parse(localStorage.getItem(userId))

        let userMetaData = JSON.parse(localStorage.getItem(userId + "MetaData"))

        if (userDetails !== null) {

            setUserDetails(userDetails)

        }

        if (userMetaData !== null) {

            ["Semester1", "Semester2", "Semester3", "Semester4",

                "Semester5", "Semester6", "Semester7", "Semester8"].map(item => {

                    userMetaData[item] = userMetaData[item] ? userMetaData[item] : { sgpa: "", percentage: "" }

                })

            setUserMetaData(userMetaData)

        } else {

            setUserMetaData({

                Semester1: { sgpa: "", percentage: "" },

                Semester2: { sgpa: "", percentage: "" },

                Semester3: { sgpa: "", percentage: "" },

                Semester4: { sgpa: "", percentage: "" },

                Semester5: { sgpa: "", percentage: "" },

                Semester6: { sgpa: "", percentage: "" },

                Semester7: { sgpa: "", percentage: "" },

                Semester8: { sgpa: "", percentage: "" },

            })

        }

    }

    const handleOnChangeValueSemester = (subject, id, value) => {

        setSemisterData((previousState) => ({

            ...previousState,

            [subject]: {

                ...previousState[subject],

                [id]: value

            }

        }))

    }

    const handleOnChangeValueCgpa = (semester, id, value) => {

        setUserMetaData((previousState) => ({

            ...previousState,

            [semester]: {

                ...previousState[semester],

                [id]: value,

                percentage: ((value - 0.75) \* 10).toFixed(2)

            }

        }))

    }

    useEffect(() => {

        if (userMetaData !== null) {

            let sum = 0

            let count = 0

            Object.keys(userMetaData).map(semester => {

                if (userMetaData[semester].sgpa !== "") {

                    sum += parseFloat(userMetaData[semester].sgpa)

                    count++;

                }

            })

            const cgpa = sum > 0 ? (sum / count).toFixed(2) : 0

            const percentage = cgpa > 0 ? ((cgpa - 0.75) \* 10).toFixed(2) : 0

            setUserCgpa({ cgpa: cgpa, percentage: percentage })

        }

    }, [userMetaData])

    useEffect(() => {

        if (semesterSelected !== null) {

            setSemisterData(userDetails[semesterSelected.value] ? userDetails[semesterSelected.value] : {

                "1": {

                    "subjectCode": "",

                    "credit": "",

                    "marks": ""

                }

            })

        }

    }, [semesterSelected])

    useEffect(() => {

        updateUserDetails()

    }, [userId])

    useEffect(() => {

        if (userId !== null) {

            localStorage.setItem(userId, JSON.stringify(userDetails))

        }

    }, [userDetails])

    useEffect(() => {

        if (userId !== null && updateClicked === true) {

            setUpdateClicked(false)

            localStorage.setItem(userId + "MetaData", JSON.stringify(userMetaData))

        }

    }, [updateClicked])

    useEffect(() => {

        const userID = localStorage.getItem("loginUserData")

        setUserId(userID)

    }, [])

    return (

        <DocumentTitle title='Dashboard'>

            <div className='dashboard'>

                <h1 className='title'>SGPA & CGPA Notes</h1>

                <div className='row'>

                    <div className='col-12' >

                    <Button

                        className="logout"

                        id="download"

                        onClick={() => {

                            navigate("/login")

                        }}

                    >

                        Logout

                    </Button>

                    </div>

                </div>

                <Accordion defaultActiveKey="1">

                    <Accordion.Item eventKey="0">

                        <Accordion.Header >SGPA</Accordion.Header>

                        <Accordion.Body>

                            <div className='row'>

                                <Select

                                    className="m-b-20"

                                    placeholder="Select Semester"

                                    value={semesterSelected}

                                    onChange={(selecetd) => setSemesterSelected(selecetd)}

                                    theme={(theme) => ({

                                        ...theme,

                                        colors: {

                                            ...theme.colors,

                                            primary: '#0000000',

                                        },

                                    })}

                                    options={SemesterOptions}

                                />

                            </div>

                            {Object.keys(semesterData).length === 0 ? null :

                                <>

                                    <div className='row'>

                                        <div className='col-12'>

                                            <i className="fas fa-plus-circle "

                                                onClick={handleClickAdd}>

                                            </i>

                                            <i className="fas fa-minus-circle "

                                                onClick={handleClickMinus}>

                                            </i>

                                        </div>

                                        <div className='semester\_table'>

                                            <Table bordered hover ref={inputRefSgpa}>

                                                <thead>

                                                    <tr>

                                                        <th className="text-center" colSpan={4}>

                                                            {`SGPA Report - ${semesterSelected.label}`}</th>

                                                    </tr>

                                                    <tr>

                                                        <th>Sl. No.</th>

                                                        <th>Subject Code          </th>

                                                        <th>Credits  </th>

                                                        <th>Marks  </th>

                                                    </tr>

                                                </thead>

                                                <tbody>

                                                    {

                                                        Object.keys(semesterData).map((subject, index) => (

                                                            <tr key={subject}>

                                                                <td>{index + 1}</td>

                                                                {Object.keys(semesterData[subject]).map(item => (

                                                                    <td key={`${subject}\_${item}`}>

                                                                        <Form.Control

                                                                            type={(item === "subjectCode") ? "text" : "number"}

                                                                            id={`${subject}\_${item}`}

                                                                            value={semesterData[subject][item]}

                                                                            onChange={(event) => handleOnChangeValueSemester(subject, item, event.target.value)}

                                                                        />

                                                                    </td>

                                                                ))}

                                                            </tr>

                                                        ))

                                                    }

                                                    {semesterMetaData === null ? null :

                                                        <>

                                                            <tr>

                                                                <td className="text-center" colSpan={3}>Total SGPA</td>

                                                                <td>{semesterMetaData.sgpa}</td>

                                                            </tr>

                                                            <tr>

                                                                <td className="text-center" colSpan={3}>Total Percentage</td>

                                                                <td>{semesterMetaData.percentage}</td>

                                                            </tr>

                                                        </>

                                                    }

                                                </tbody>

                                            </Table>

                                        </div>

                                        <div className='controls row'>

                                            <div className='col-lg-1'>

                                                <Button

                                                    id="update"

                                                    onClick={handleUpdate}

                                                >

                                                    Update

                                                </Button>

                                            </div>

                                            <div className='col-lg-3'>

                                                <Button

                                                    id="download"

                                                    onClick={() => printDocument(inputRefSgpa)}

                                                >

                                                    Download Report

                                                </Button>

                                            </div>

                                        </div>

                                    </div>

                                </>

                            }

                        </Accordion.Body>

                    </Accordion.Item>

                    <Accordion.Item eventKey="1">

                        <Accordion.Header onClick={updateUserDetails}>CGPA</Accordion.Header>

                        <Accordion.Body>

                            <div className='row'>

                                {Object.keys(userMetaData).length === 0 ? null :

                                    <>

                                        <div className='semester\_table'>

                                            <Table bordered hover ref={inputRefCgpa}>

                                                <thead>

                                                    <tr>

                                                        <th className="text-center" colSpan={4}>

                                                            {`CGPA Report`}</th>

                                                    </tr>

                                                    <tr>

                                                        <th>Semester</th>

                                                        <th>SGPA</th>

                                                        <th>Percentage</th>

                                                    </tr>

                                                </thead>

                                                <tbody>

                                                    {

                                                        Object.keys(userMetaData).map((semester, index) => (

                                                            <tr key={semester}>

                                                                <td>{index + 1}</td>

                                                                {Object.keys(userMetaData[semester]).map(item => (

                                                                    <td key={`${semester}\_${item}`}>

                                                                        {console.log(item)}

                                                                        <Form.Control

                                                                            readOnly={item === "sgpa" ? false : true}

                                                                            type={(item === "subjectCode") ? "text" : "number"}

                                                                            id={`${semester}\_${item}`}

                                                                            value={userMetaData[semester][item]}

                                                                            onChange={(event) => handleOnChangeValueCgpa(semester, item, event.target.value)}

                                                                        />

                                                                    </td>

                                                                ))}

                                                            </tr>

                                                        ))

                                                    }

                                                    {userCgpa === null ? null :

                                                        <>

                                                            <tr>

                                                                <td className="text-center" colSpan={2}>Total CGPA</td>

                                                                <td>{userCgpa.cgpa}</td>

                                                            </tr>

                                                            <tr>

                                                                <td className="text-center" colSpan={2}>Total Percentage</td>

                                                                <td>{userCgpa.percentage}</td>

                                                            </tr>

                                                        </>

                                                    }

                                                </tbody>

                                            </Table>

                                        </div>

                                        <div className='controls row'>

                                            <div className='col-lg-3'>

                                                <Button

                                                    id="download"

                                                    onClick={() => printDocument(inputRefCgpa)}

                                                >

                                                    Download Report

                                                </Button>

                                            </div>

                                        </div>

                                    </>}

                            </div>

                        </Accordion.Body>

                    </Accordion.Item>

                </Accordion>

            </div>

        </DocumentTitle>

    )

}

.dashboard {

    box-shadow: 1px 0 20px rgb(0 0 0 / 5%);

    background-color: rgba(255, 255, 255, 0.90);

    min-height: 100vh;

    padding: 30px;

}

.dashboard .title {

    text-align: center;

    color: #007bff;

    font-weight: bold;

    margin-bottom: 30px;

}

.dashboard .accordion-item {

    border: none;

    box-shadow: 1px 0 20px rgb(0 0 0 / 5%);

}

.dashboard .accordion-button:not(.collapsed) {

    font-size: 25px;

    font-weight: bold;

    color: white;

    background-color: #007bff;

    border: none;

    box-shadow: inset 0 -1px 0 rgb(0 0 0 / 13%);

}

.dashboard .accordion-item {

    margin-top: 20px;

}

.dashboard .accordion-button.collapsed {

    color: #007bff;

    font-weight: bold;

    font-size: 20px;

}

.dashboard i {

    color: #007bff;

    cursor: pointer;

    font-size: 40px;

    margin-top: 20px;

    margin-bottom: 20px;

    margin-left: 20px;

    float: right;

}

.controls .btn {

    margin-top: 20px;

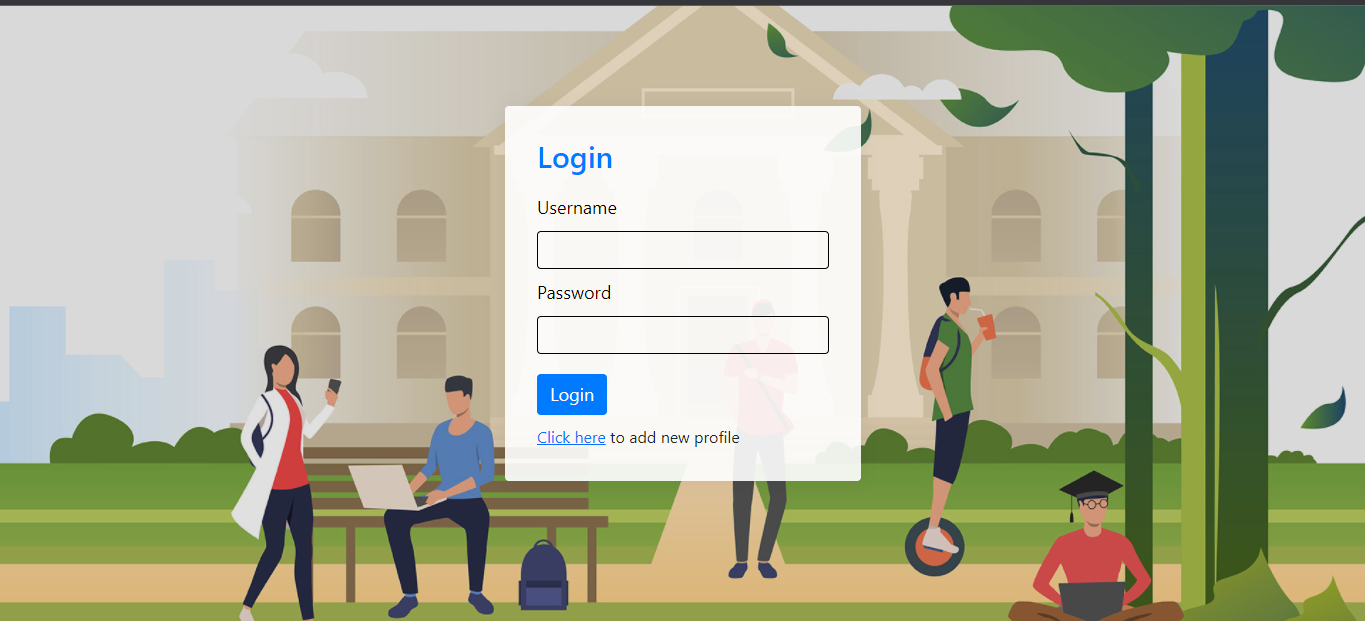
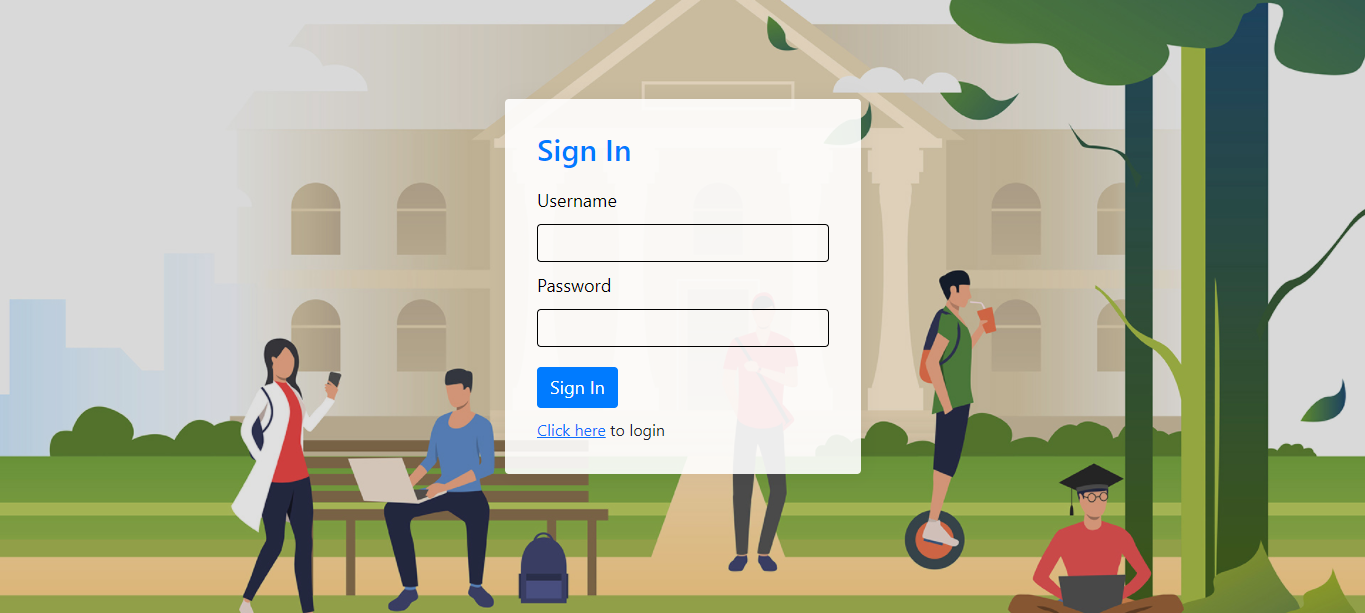
}

// .dashboard .semester\_table{

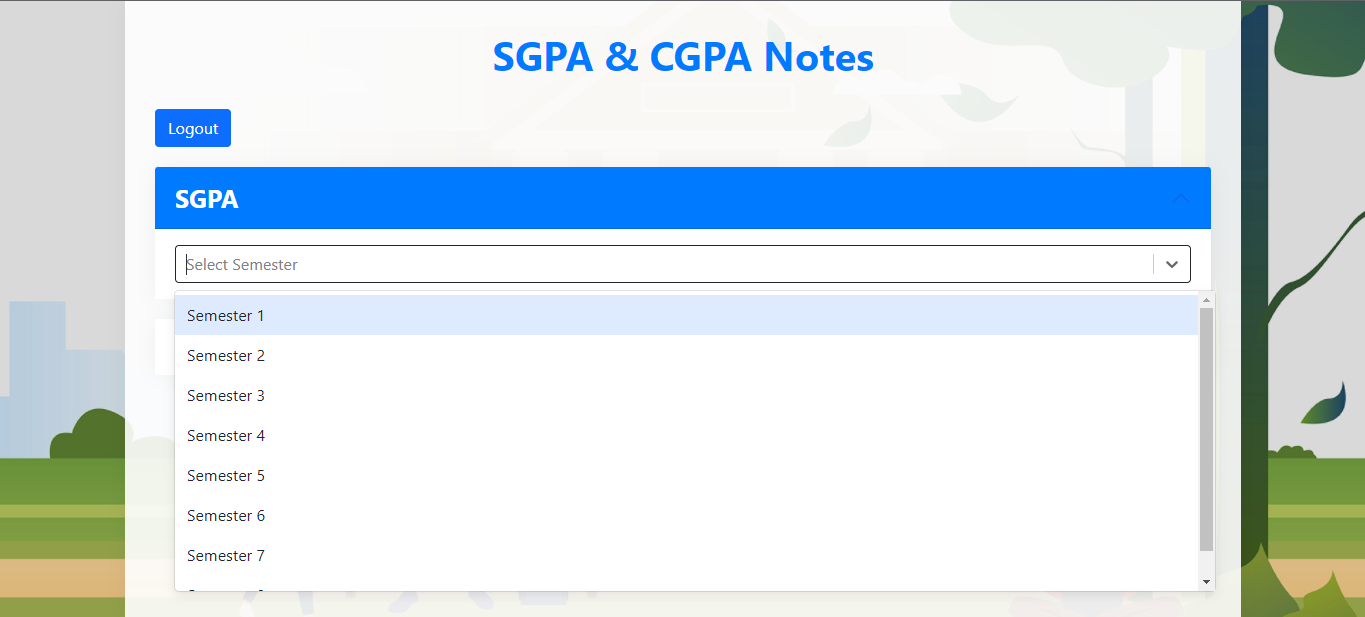
//     padding: 20px;

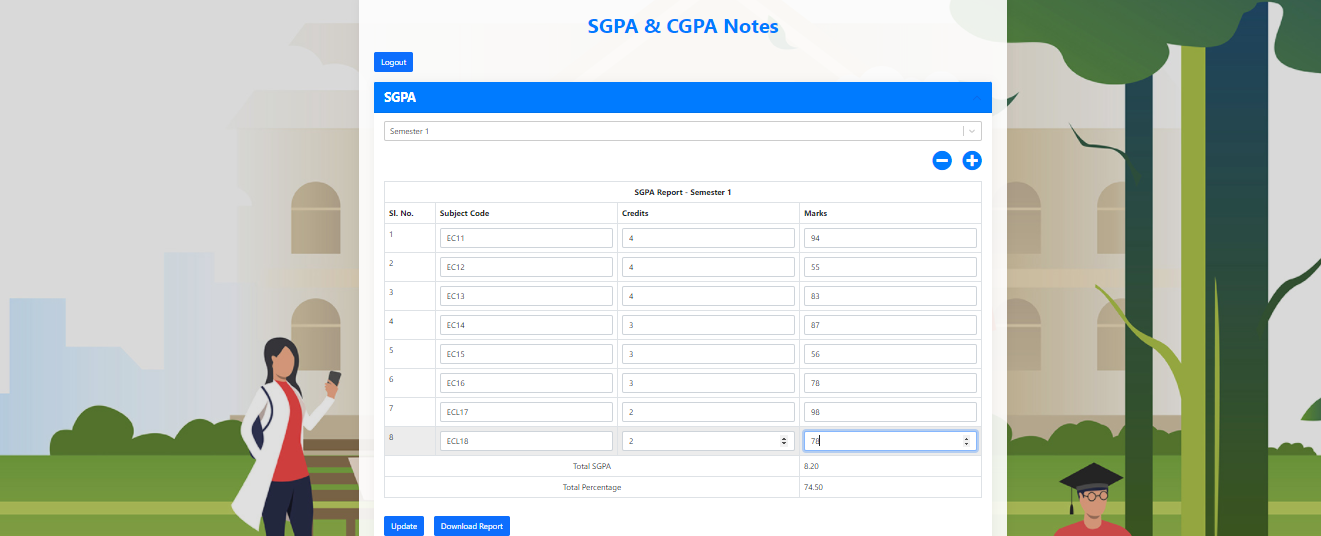
// }

### SNAPSHOTS











### BIBLIOGRAPHY

* https://[www.w3schools.com](http://www.w3schools.com/)
* https://www.geeksforgeeks org
* https://freefrontend.com
* <https://css-tricks.com/>
* https://www.takeiteasyengineers.com
* https://dev.to/mychi\_darko/php-tips-and-tricks-4kpn