BMI Calculator

TEAM MEMBERS:

ROLL NUMBER	NAME
22ALR107	VAANMUGILAN T
22ALR091	SHYAM S
22ALR074	RAGUL G

INTRODUCTION:

The BMI (Body Mass Index) Calculator Project is a comprehensive full-stack web application designed to help users calculate and track their BMI. The application is built using modern web development technologies, ensuring a seamless and interactive user experience. It serves as an excellent learning project for developers aiming to understand the intricacies of full-stack development, including front-end and back-end integration, database management, and deployment.

KEY FEATURES:

- User Registration and Authentication
- BMI Calculation
- Responsive Design
- User Profile Dashboard
- RESTful API
- Data Validation and Error Handling
- Interactive UI Elements
- Scalability and Performance
- Deployment and Version Control
- Security Features

REQUIREMENTS:

- **Front-End Programming**

    1. **CSS (Cascading Style Sheets):** CSS is used to define the visual appearance and layout of web pages. It allows developers to apply styles, such as colors, fonts, spacing, and positioning, to HTML elements, creating a visually appealing and consistent user interface.

    2. **JavaScript:** JavaScript is a powerful programming language that adds interactivity and dynamic behavior to web pages. It enables developers to manipulate and modify HTML and CSS, handle user interactions, perform calculations, validate forms, and make asynchronous requests to servers.

    3. **React:** React is a popular JavaScript library for building user interfaces. It provides a component-based approach to web development, allowing developers to create reusable UI components that update efficiently based on changes in data.React uses a virtual DOM (Document Object Model) to optimize performanceand facilitate the building of complex web applications.

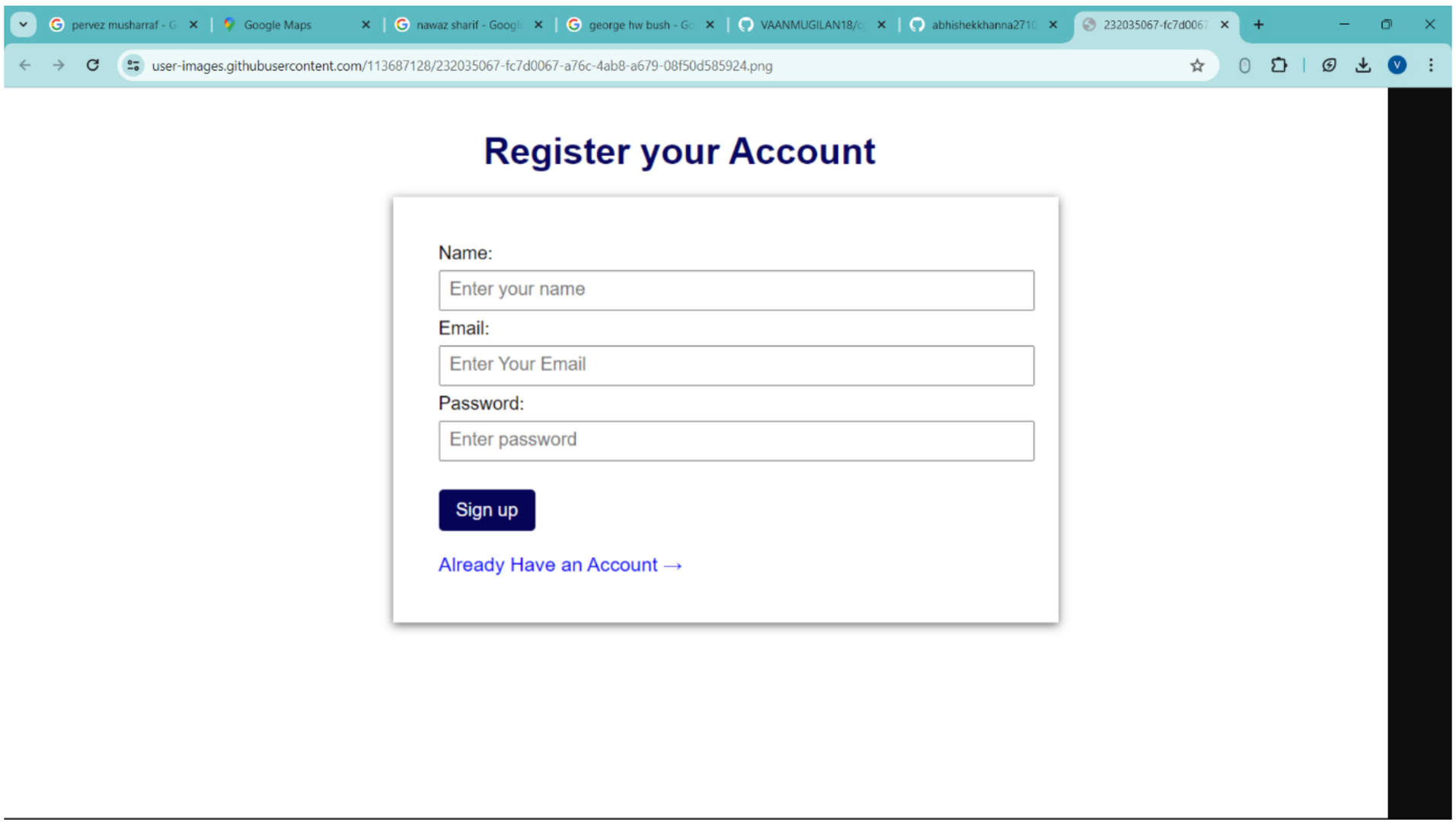
- **Back-End Programming**

    1. **Database Management:** Backend programming involves working with databases to store and retrieve data. Developers interact with databases using query languages like SQL (Structured Query Language) or NoSQL databases. They design and implement database schemas, optimize queries, and ensure data integrity and security.

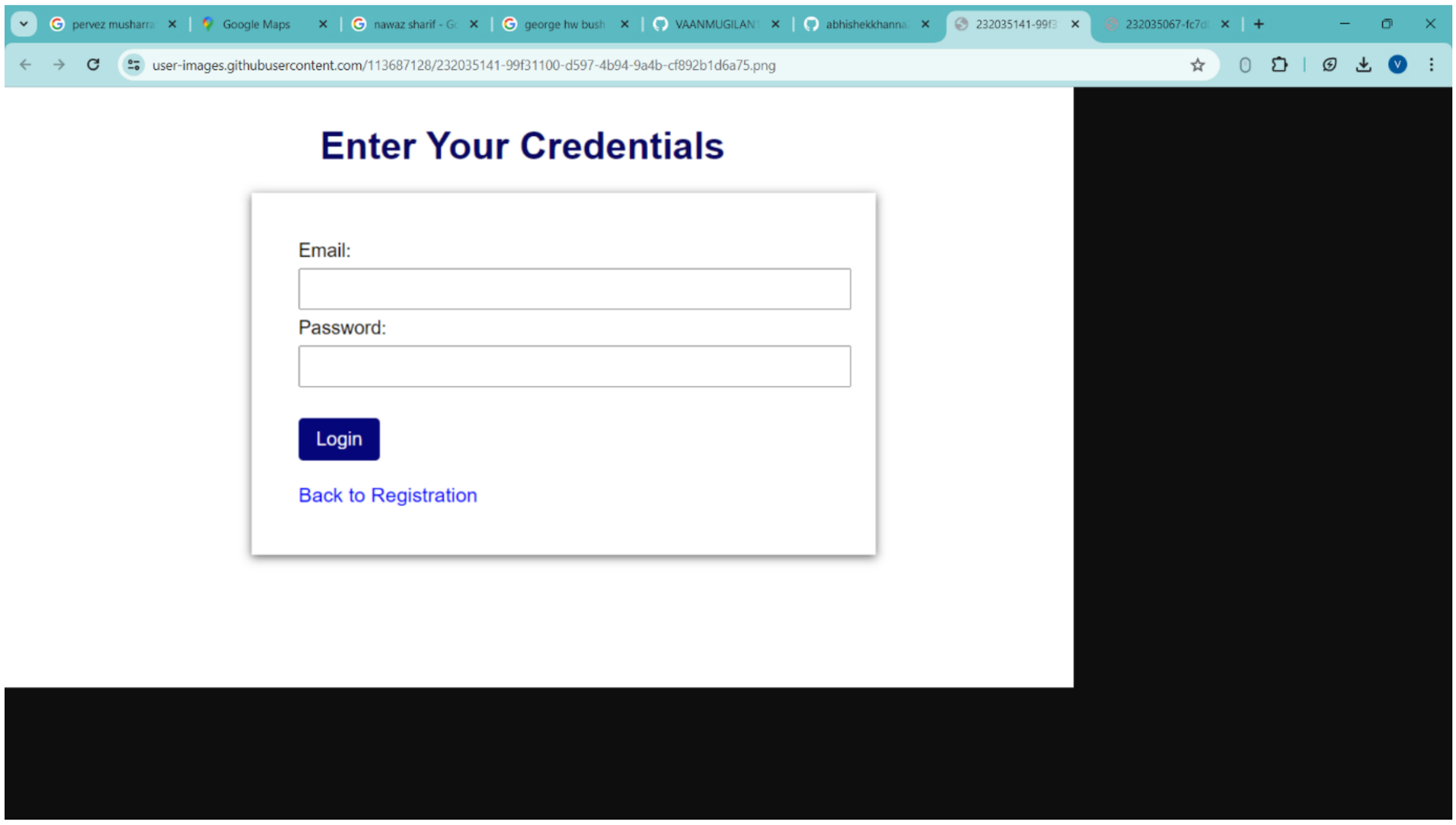
    2. **APIs (Application Programming Interfaces)**: Backend developers build APIs that allow communication between the frontend and backend components of a web application. APIs define the protocols and rules for how different software components can interact and exchange data. Commonly used API standards include RESTful APIs and GraphQL.

The Experience of Project:

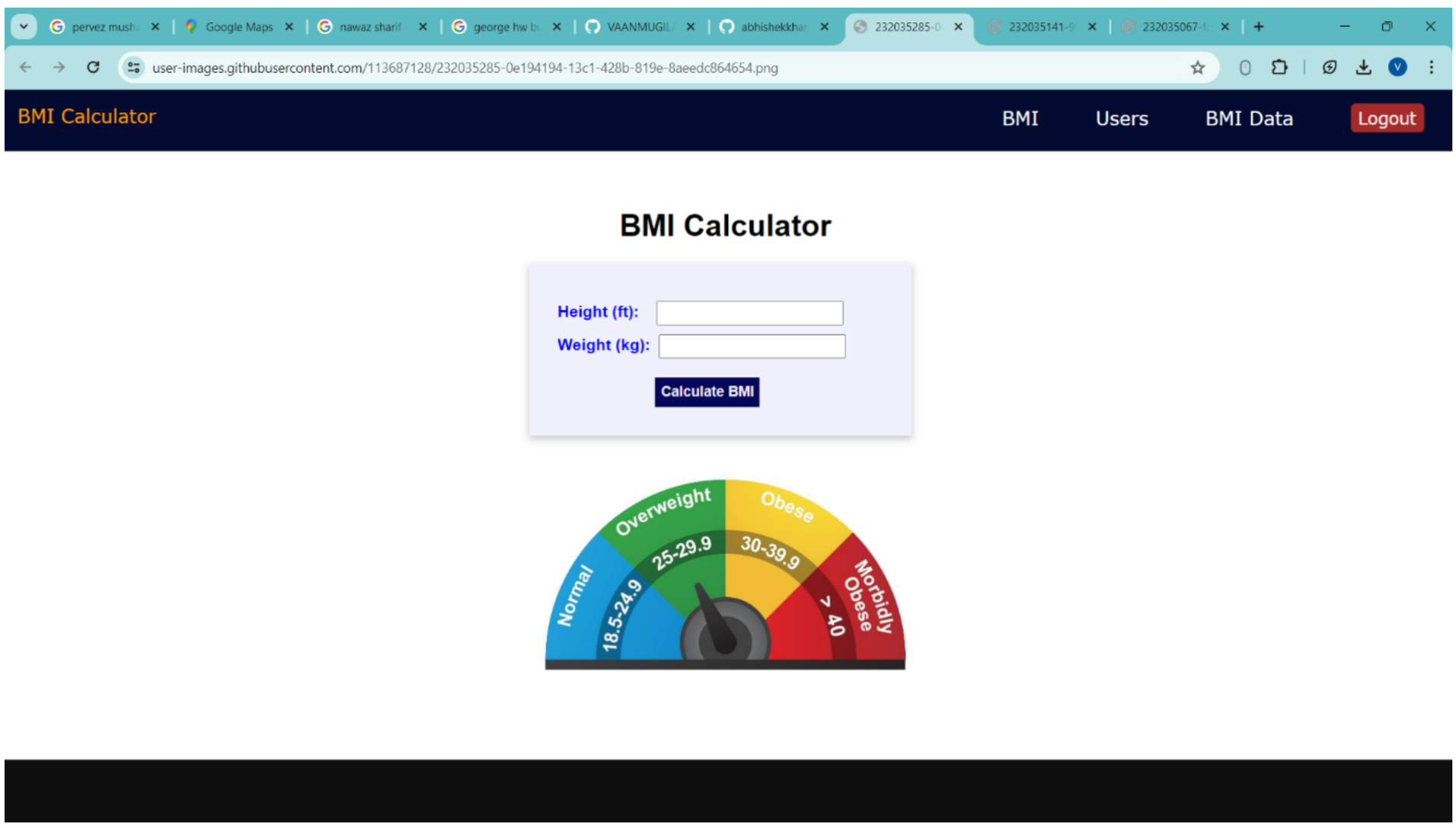
1)Firstly Create Your Account:



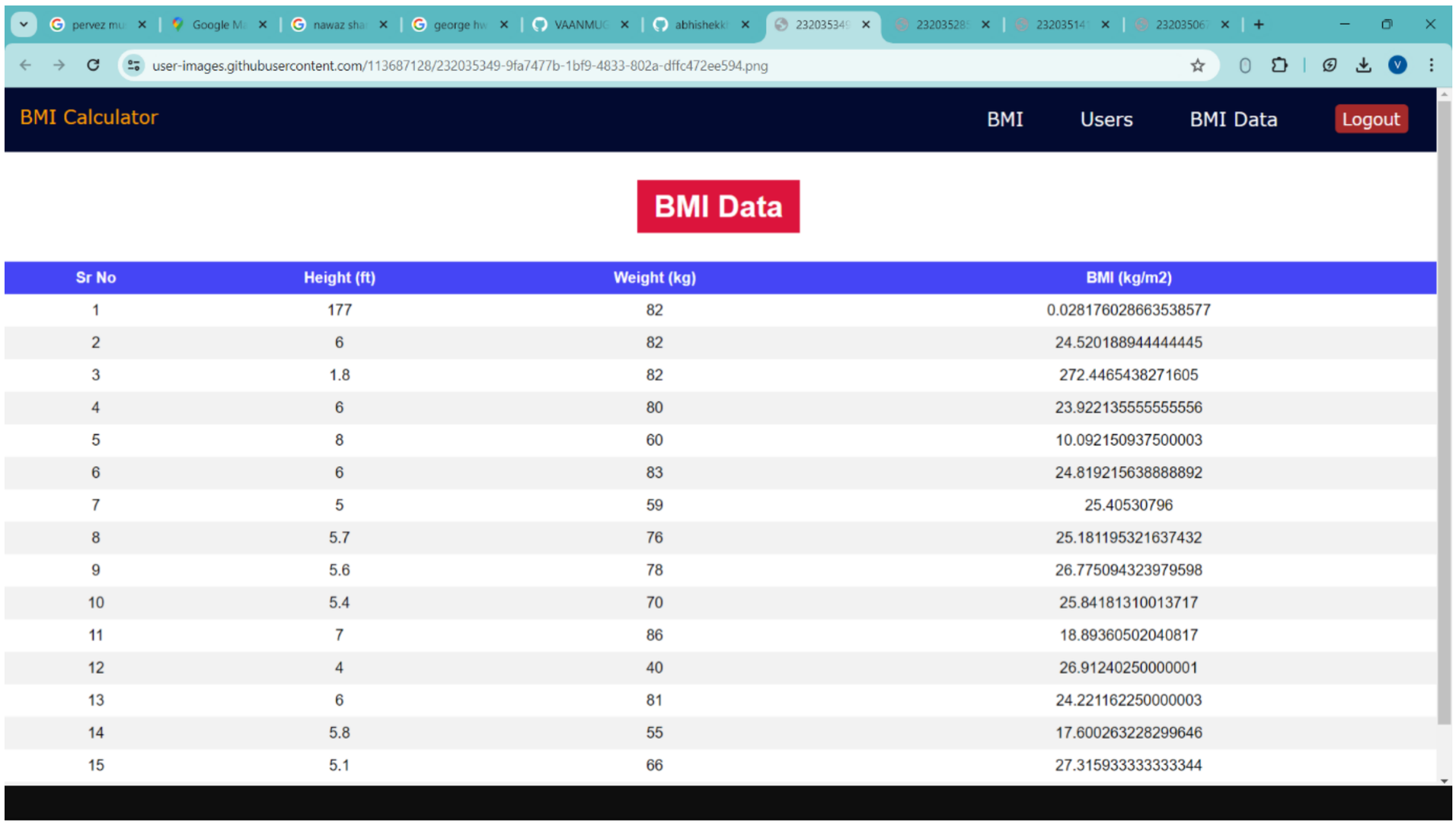
2)Fill your credentials and login it:



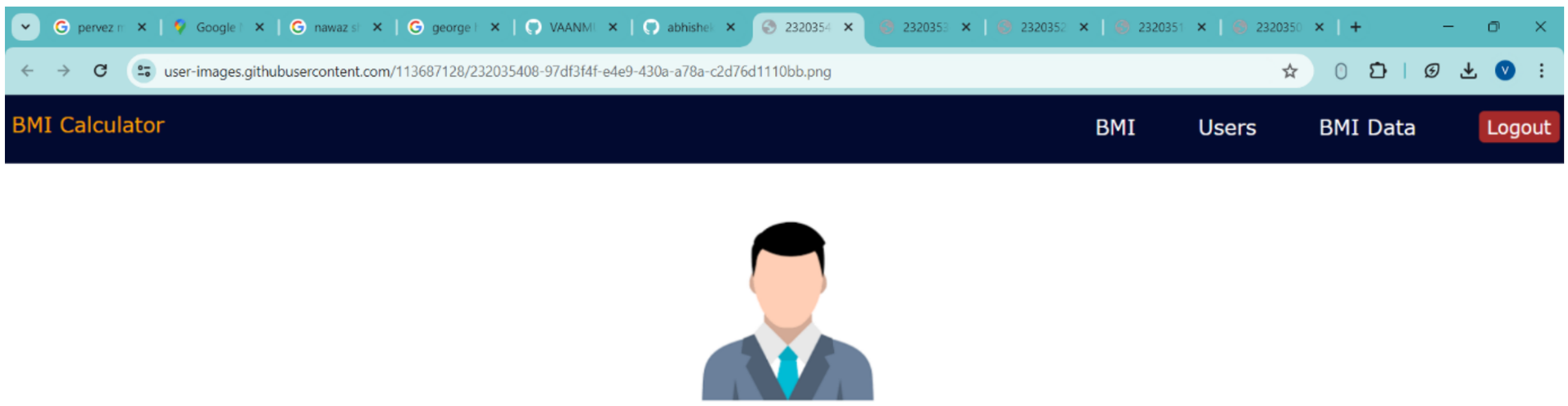
3)Set your Height and Weight and click submit:



4)BMI Calculation Page:



5)User Page:



The Features Developed:

- Login Page
- SignUp Page
- BMI Page
- Users Page
- BMI Data Page

SAMPLE CODING:

Index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <link rel="icon" type="image/svg+xml" href="/vite.svg" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Vite + React</title>
 </head>
 <body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
 </body>
</html>
```

.gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
```

```
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
.ntvs
*.njsproj
*.sln
*.sw?
```

vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
})
```

Index.js

```
require('dotenv').config();
const express = require("express");
const app = express();
const cors = require('cors');
const connection = require("./db/Connection.js")
const userRoutes = require("./Routes/UserRoutes");
```

```
const authRoutes = require("./Routes/Auth");
const BmiRoutes = require("./Routes/Bmi")


//MongoDb Connection
connection();

// middlewares
app.use(express.json())
app.use(cors());

// router file linked
app.use("/api/users", userRoutes)
// app.use("/api/users/:id", userRoutes)
app.use("/api/auth", authRoutes)
app.use("/api/calculate-bmi", BmiRoutes)
app.use("/api/logout", userRoutes)


const port = process.env.PORT || 8000;

app.get("/", (req, res) => {
   res.send("Hello World Bmi ");
})

app.listen(8000, () => {
   console.log(Server has started on localhost//:${port})
})
```

Connection.js

```
const mongoose = require("mongoose");
module.exports = () => {
    const connectionParams = {
        useNewUrlParser: "true",
        useUnifiedTopology: "true"
    }
    try {
        mongoose.connect(process.env.MONGODB_URL, connectionParams)
        console.log("Connected Database");
    } catch (error) {
        console.log(error.message)
        console.log("Not connected");
    }
}
```

BmiSchema.js

```
const mongoose = require('mongoose');
const bmiSchema = new mongoose.Schema({
    height: { type: Number, required: true },
    weight: { type: Number, required: true },
    bmi: { type: Number, required: true }
});

const Bmi = mongoose.model('Bmi', bmiSchema);

module.exports = Bmi;
```

UserSchema.js

```
const mongoose = require("mongoose")
const jwt = require('jsonwebtoken')
const joi = require('joi');
const passwordComplexity = require("joi-password-complexity")

const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    }
})

userSchema.methods.generateAuthToken = function () {
    const token = jwt.sign({ _id: this._id }, process.env.JWTPRIVATEKEY, { expiresIn:
"1hr" })
    return token;
}
```

```
const User = mongoose.model("user", userSchema);


const validate = (data) => {
    const schema = joi.object({
        name: joi.string().required().label("Name"),
        email: joi.string().email().required().label("Email"),
        password: passwordComplexity().required().label('Password')
    })
    return schema.validate(data)
}


module.exports = { User, validate };
```

Auth.js

```
const router = require("express").Router();
const { User } = require("../models/UserSchema");
const joi = require("joi");
const bcrypt = require("bcrypt");

router.post("/", async (req, res) => {
    try {
        const { error } = validate(req.body)
        if (error) {
            return res.status(400).send({ message: error.details[0].message })
        }
        const user = await User.findOne({ email: req.body.email });
```

```
if (!user) {
    return res.status(401).send({ message: "Invalid Email or Password" })
}

const validPassword = await bcrypt.compare(
    req.body.password, user.password
);

if (!validPassword) {
        return res.status(401).send({ message: "Invalid Email or Password" })
    }


    const token = user.generateAuthToken();
    let userOne = await User.findOne({ email: req.body.email });
    if (!userOne) {
        return res.status(401).send({ message: "Invalid Email or Password" })
    }
    res.status(200).send({ data: userOne._id, message: "Logged in Successfully" })
} catch (error) {
    console.log(error.message)
    res.status(500).send({ message: "Internal Server Errors" })
    }
})

const validate = (data) => {
    const schema = joi.object({
        email: joi.string().email().required().label("Email"),
        password: joi.string().required().label("Password")
    })
    return schema.validate(data)
}
```

```
module.exports = router;
```

Bmi.js

```
const router = require("express").Router();
const Bmi = require("../models/BmiSchema")
const auth = require("./Auth")

router.post('/', async (req, res) => {
    const { height, weight } = req.body;

    // Calculate BMI
    const heightInMeters = height / 3.281;
    const bmi = weight / (heightInMeters ** 2);



    const newBmi = new Bmi({ height, weight, bmi });
    await newBmi.save();

    // BMI value
    res.json({ bmi });
});



router.get('/', async (req, res) => {
    try {
        const bmis = await Bmi.find({});
        res.json(bmis);
    } catch (err) {
```

```
        console.error(err);
        res.status(500).send('Internal server error');
    }
});



module.exports = router;
```

UserRoutes.js

```
const router = require("express").Router();
const bcrypt = require('bcrypt');

const { User, validate } = require("../models/UserSchema");



router.post("/", async (req, res) => {
    try {
        const { error } = validate(req.body);
        if (error) {
            return res.status(400).send({ message: error.details[0].message })
        }
        const user = await User.findOne({ email: req.body.email })
        if (user) {
            return res.status(409).send({ message: "User with given Email already
Exists" })
        }
```

```
    const salt = await bcrypt.genSalt(Number(process.env.SALT))
    const hashpassword = await bcrypt.hash(req.body.password, salt);

    await new User({ ...req.body, password: hashpassword }).save();
    res.status(201).send({ message: "User Created Successfully" })
} catch (error) {
    console.log(error)
    res.status(500).send({ message: "Internal Server Error" })
}
})



// GET one users
router.get('/:id', async (req, res) => {
    try {
        const user = await User.findById(req.params.id);
        console.log(req.params.id)
        res.status(200).json(user);
        console.log(user)

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


    } catch (error) {
        console.error(error.message);
        res.status(500).json({ message: 'Internal server error' });
    }
});
```

```
// Logout user
router.post('/', async (req, res) => {
    try {
        req.user.tokens = req.user.tokens.filter(token => token.token !== req.token);
        await req.user.save();

        res.send('Logged out successfully');
    } catch (error) {
        console.error(error);
        res.status(500).send('Internal server error');
    }
});

module.exports = router;
```

Navbar.css

```
body {
    margin: 0 auto;
}

.navbar {
    top: 0;
    left: 0;
    padding-top: 10px;
    position: relative;
    width: 100%;
    height: 60px;
    background-color: rgb(2, 10, 48);
    margin-bottom: 30px;
```

```
}

.Links {
    text-decoration: none;
    color: white;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 20px;
    margin: 10px 30px;
    display: flex;
}

.LinkTag {
    float: right;
    display: flex;
    justify-content: center;
    align-items: center;
}

.Link {
    text-decoration: none;
    color: rgb(255, 162, 1);
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 20px;
    margin: 40px 20px;
    /* margin-top: 20px!important; */
    position: relative;
    top: 11px;
}

.Loglink{
    background-color: brown;
    color: white;
```

```
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size:18px;
    margin: 10px 30px;
    text-decoration: none;
    padding: 4px 8px;
    border-radius: 5px;
}
```

Navbar.jsx

```
import React from 'react';
import { Link } from 'react-router-dom';
import './Navbar.css';

function Navbar() {
    const handleout = (e) => {
        alert("User Logout Successfully")
    }
    return (
<div className="navbar">

    <Link className='Link' to="">BMI Calculator</Link>
    <div className='LinkTag'>

        <Link className='Links' to="/bmi-calculator">BMI</Link>
        <Link className='Links' to="/user-section">Users</Link>
        <Link className='Links' to="/bmi-data">BMI Data</Link>
        <Link onClick={handleout} className='Loglink' to="/">Logout</Link>
    </div>
</div>
```

```
    )
}

export default Navbar;
```

BMI_data.jsx

```
import React, { useState, useEffect } from 'react';
import Navbar from '../../Components/Navbar/Navbar';
import "./BMI.css"
function BMI_data() {
    const [bmis, setBmis] = useState([]);

    useEffect(() => {
        fetch('https://bmi-bclr.onrender.com/api/calculate-bmi')
            .then(response => response.json())
            .then(data => setBmis(data))
            .catch(error => console.error(error));
    }, []);

    return (
        <div className='Bmi_data'>
            <Navbar />
            <h1>BMI Data</h1>
            <table className="table-style">
                <thead>
                    <tr>
                        <th>Sr No</th>
                        <th>Height (ft)</th>
                        <th>Weight (kg)</th>
                        <th>BMI (kg/m2)</th>
```

```
            </tr>
          </thead>
          <tbody>
            {bmis.map((bmi, index) => (
              <tr key={bmi._id}>
                <td>{index + 1}</td>
                <td>{bmi.height}</td>
                <td>{bmi.weight}</td>
                <td>{bmi.bmi}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    );
}

export default BMI_data;
```

BMI.css

```
.table-style {
    border-collapse: collapse;
    width: 100%;
    font-family: Arial, Helvetica, sans-serif;
    margin: auto;
}

.table-style th,
.table-style td {
    text-align: center;
```

```
    padding: 8px;
}

.table-style th {
    background-color: #4545f5;
    color: white;
}

.table-style tr:nth-child(even) {
    background-color: #f2f2f2;
}

.Bmi_data h1 {
    text-align: center;
    font-family: sans-serif;
    background-color: crimson;
    color: white;
    width: 10%;
    margin: auto;
    padding: 10px;
    margin-bottom: 30px;
}
```

Calculation.css

```
.main_bmi {
    width: 27%;
    margin: auto;
    padding: 10px;
}

.main_bmi h1 {
```

```
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
}

.Bmi_cal {
    padding: 30px;
    box-shadow: rgba(0, 0, 0, 0.24) 0px 3px 8px;
    width: 83%;
    background-color: rgba(236, 236, 251, 0.764);
}

.Bmi_cal input {
    padding: 3px 16px;
    margin-top: 10px;
}

.Bmi_cal label {
    color: blue;
    font-size: 17px;
    font-weight: 600;
    font-family: Arial, Helvetica, sans-serif;
}

.btn {
    background-color: rgb(5, 5, 89);
    color: rgb(241, 240, 240);
    padding: 7px 6px;
    border: none;
    font-size: 15px;
    font-weight: 600;
}
```

```
.btn-top {
    width: 40%;
    margin: auto;
    margin-top: 20px;
}

.Result {
    font-family: Arial, Helvetica, sans-serif;
    font-weight: 600;
    font-size: 20px;
    text-align: center;
    margin-top: 40px;
}

.imago {
    width: 100%;
    margin-top: 40px;
}
```

Calculation.jsx

```
import React, { useState } from 'react'
import Navbar from '../../Components/Navbar/Navbar';
import "./Calculation.css"

function Calculation() {
    const [height, setHeight] = useState('');
    const [weight, setWeight] = useState('');
    const [bmi, setBmi] = useState(null);

    const handleHeightChange = (event) => {
        setHeight(event.target.value);
```

```
};

const handleWeightChange = (event) => {
    setWeight(event.target.value);
};

const handleSubmit = async (event) => {
    event.preventDefault();
    const response = await fetch('https://bmi-bclr.onrender.com/api/calculate-bmi',
{
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ height, weight })
      });
      const data = await response.json();
      setBmi(data.bmi);
  };


  return (
      <div >
        <Navbar />
        <div className='main_bmi'>
          <h1>BMI Calculator</h1>

          <form className='Bmi_cal' onSubmit={handleSubmit}>
            <label>
              Height (ft):    
              <input type="number" value={height}
onChange={handleHeightChange} />
            </label>
```

```
            <br />
            <label>
               Weight (kg):  
               <input type="number" value={weight}
onChange={handleWeightChange} />
            </label>
            <br />
            <div className='btn-top'>
               <button className='btn' type="submit">Calculate BMI</button>
            </div>
         </form>
         {bmi !== null && (
            <p className='Result'><span style={{ color: "red" }}>Result :</span>
Your BMI is {bmi.toFixed(2)} kg/m2</p>
         )}

         <img className='imago'
src="https://hips.hearstapps.com/hmg-prod/images/1076/bmi-myths-main-15157
02962.jpg?crop=1xw:0.786xh;center,top&resize=1200:*" alt="" />
      </div>
      </div>
   )
}

export default Calculation
```

App.css

```
#root {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
```

```
  text-align: center;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}
.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: no-preference) {
  a:nth-of-type(2) .logo {
    animation: logo-spin infinite 20s linear;
  }
}

.card {
```

```
  padding: 2em;
}

.read-the-docs {
  color: #888;
}
```

App.jsx

```
import React from "react"
import { Routes, Route } from "react-router-dom";
import SignUp from "./Pages/Login/SignUp/SignUp"
import Login from "./Pages/Login/Login/Login";
import Calculation from "./Pages/Calculation/Calculation";
import Profile from "./Pages/Profile/Profile";
import BMI_data from "./Pages/BMI_data/BMI_data";

function App() {

  return (
    <div className="App">

      <Routes>
        <Route path="/" exact element={<SignUp />} />
        <Route path="/login" exact element={<Login />} />
        <Route path="/bmi-calculator" exact element={<Calculation />} />
        <Route path="/user-section" exact element={<Profile />} />
        <Route path="/bmi-data" exact element={<BMI_data />} />

      </Routes>

    </div>
```

```
  )
}

export default App
```

index.css

```
:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;

  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: #242424;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  -webkit-text-size-adjust: 100%;
}

a {
  font-weight: 500;
  color: #646cff;
  text-decoration: inherit;
}
a:hover {
  color: #535bf2;
}
```

```
body {
  margin: 0;
  display: flex;
  place-items: center;
  min-width: 320px;
  min-height: 100vh;
}

h1 {
  font-size: 3.2em;
  line-height: 1.1;
}

button {
  border-radius: 8px;
  border: 1px solid transparent;
  padding: 0.6em 1.2em;
  font-size: 1em;
  font-weight: 500;
  font-family: inherit;
  background-color: #1a1a1a;
  cursor: pointer;
  transition: border-color 0.25s;
}
button:hover {
  border-color: #646cff;
}
button:focus,
button:focus-visible {
  outline: 4px auto -webkit-focus-ring-color;
}
```

```
@media (prefers-color-scheme: light) {
  :root {
    color: #213547;
    background-color: #ffffff;
  }
  a:hover {
    color: #747bff;
  }
  button {
    background-color: #f9f9f9;
  }
}
```

main.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App';
import { BrowserRouter } from 'react-router-dom';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>,
)
```

CONCLUSION:

The BMI Calculator Project serves as a robust example of a full-stack web application, integrating various technologies and best practices in modern web development. By completing this project, developers will gain hands-on experience in building and deploying a complete web application, preparing them for more complex development challenges.