# PORTFOLIO WEB-APPLICATION

**PROJECT WORK FILE**

OF MAJOR PROJECT

**BACHELOR OF TECHNOLOGY**

B.Tech. C.S.E.


SUBMITTED BY

**Sanjeet Kour Raina (2002617)**

**Saksham Sharma (2002611)**

**Veeno Rai (2002654)**

GUIDED By

**Ms. Ramanpreet Kaur**




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Chandigarh Engineering College – Landran**

**Mohali, Punjab – 140307**


August 2022 – January 2023

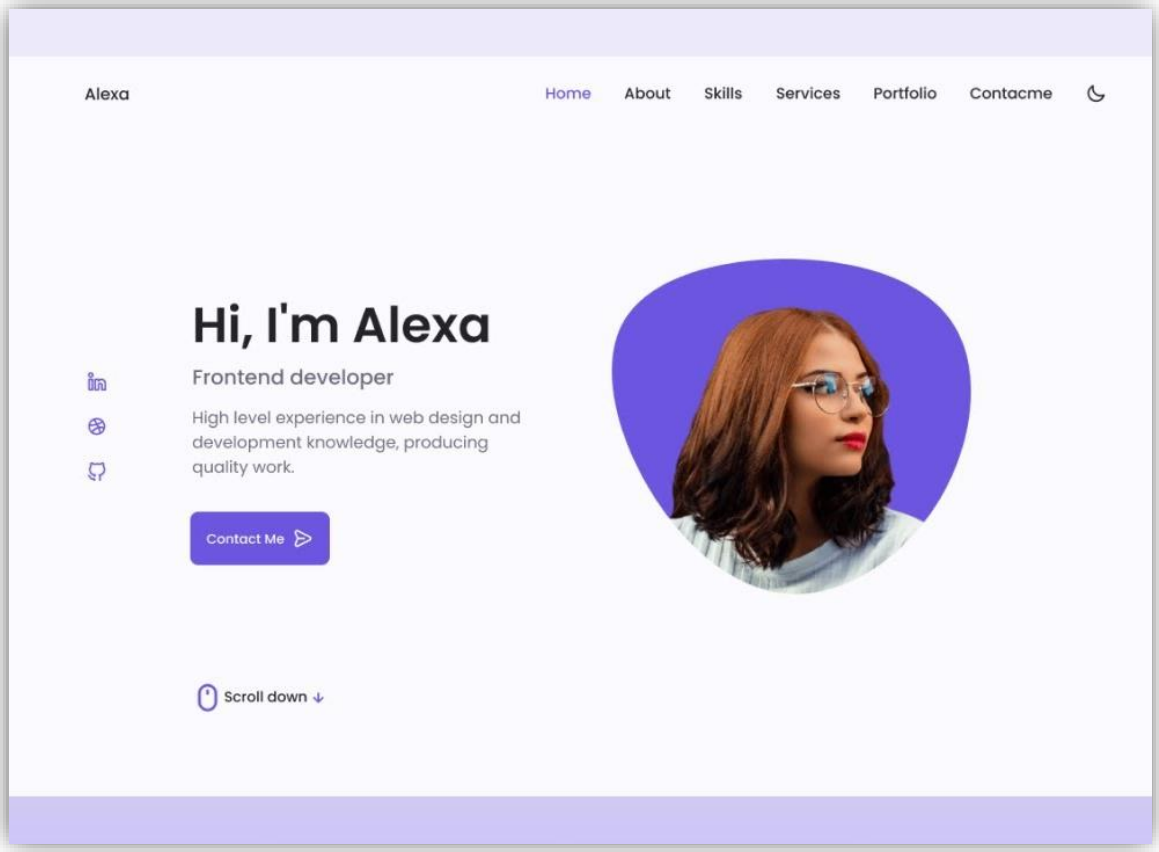
# INDEX

# INTRODUCTION

A personal portfolio website is a professional website that provides information about what you do, what services you may offer, and how to contact you or your company.

Portfolio websites are an easy way to promote yourself, your brand, or your business. You'll increase your visibility for clients, managers, or employers to find you organically. Essentially, it's a digital business card and project portfolio all in one that's accessible, sharable, and drives results.

In this project we learnt about **HTML, CSS, and JavaScript/ES6** and **React** class and function-based components to create a fully functional hosted portfolio web application.



# TECHNOLOGY STACK

## HTML:

**HTML** stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

## CSS:

**C**ascading **S**tyle **S**heets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

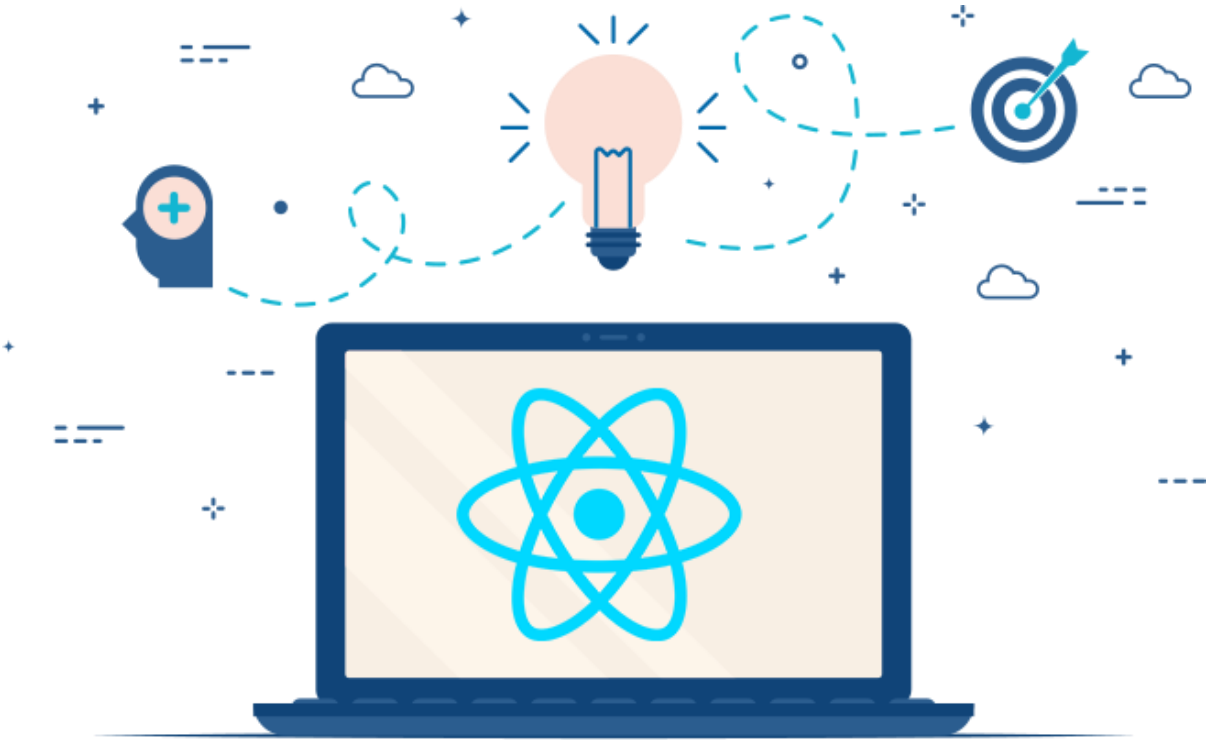
## JAVASCRIPT:

**JavaScript** is a lightweight, cross-platform, and interpreted scripting language. It is well-known for the development of web pages; many non-browser environments also use it. JavaScript can be used for **Client-side** developments as well as **Server-side** developments. JavaScript contains a standard library of objects, like **Array**, **Date**, and Math, and a core set of language elements like **operators**, **control structures**, and **statements**.

# REACT JS:

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library responsible only for the view layer of the application.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.



# Node JS:

NodeJS is event-driven and single-threaded which means the server contains a single thread that is processed one after another. Here, thread means a series of operations that the server needs to perform. Whenever there's a request made from the client, the server handles it and here that server is NodeJS which handles it with a single thread. Parallelly all the requests are made on the server and a response is given to multiple clients at the same time.

**Express.js** is a Node js web application server framework, which is specifically designed for building single-page, multi-page, and hybrid web applications.
It has become the standard server framework for node.js. Express is the backend part of something known as the MEAN stack.
The MEAN is a free and open-source JavaScript software stack for building dynamic web sites and web applications which has the following components;
**1) MongoDB** – The standard NoSQL database
**2) Express.js** – The default web applications framework

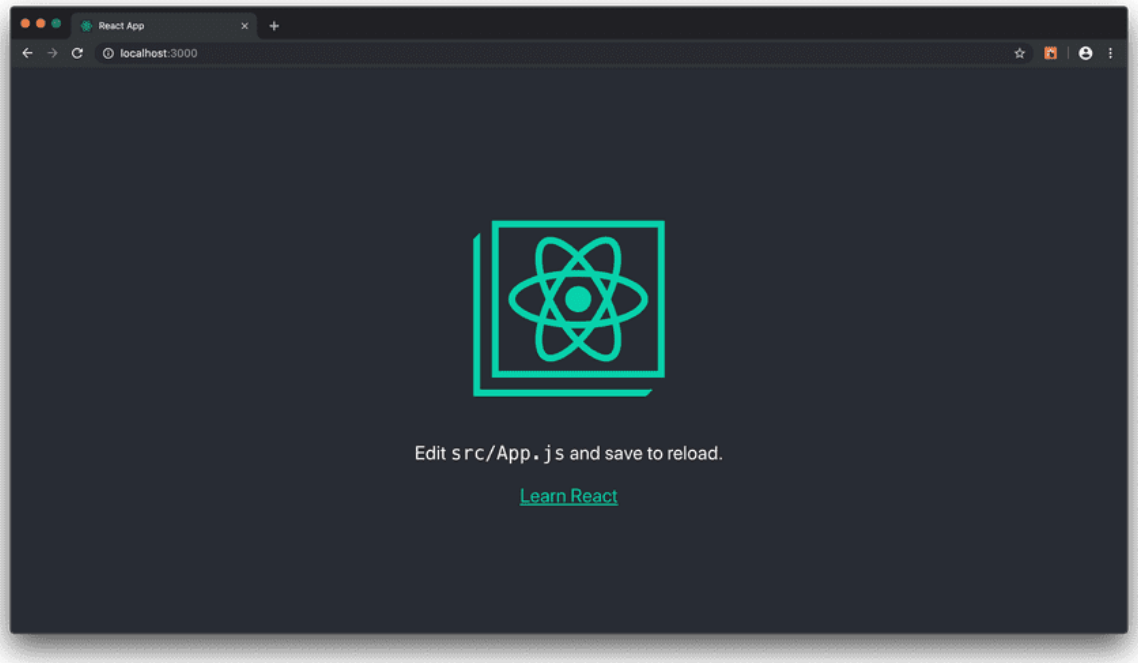
# METHODOLOGY

## Creating React App

We begin by initializing a new React project using the command `npm create-react-app new-app`. In terminal, run the following command one directory above where you want the project to live. Once the loading is complete, you can start the application by using command `npm start` and go to localhost:3000 to see your application.

We'll start by cleaning up the project a bit and delete some files that are created by the boilerplate. Delete all the files in the `/src` folder except `App.js`, `index.js`, `index.css`.

Let's also get rid of some code in the `App.js` file so we can start from scratch.



# Adding Different Components

In the folder under news-app in src folder create a folder for different components.This folder will include different file /components which we can add to our main file (App.js).

1.Navbar Component (file name: Navbar.js): This will include code for navbar.

2.About Component (file name: About.js): This will include the introduction template in which the animated text with photo is placed along with some personal detials .

3.Projects Component (file name: Projects.js): In this component we will create templates along with some function for displaying our projects.

4.Skills and Testimonials Component (file name: Skills.js & testimonial.js): In this component we will create templates for skills and testimonials and embed them with functions to automatically update them with different input values.

```
my-portfolio
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.js
    ├── data.js
    ├── index.css
    ├── index.js
    └── components
        ├── About.js
        ├── Contact.js
        ├── Navbar.js
        ├── Projects.js
        ├── Skills.js
        └── Testimonials.js
```

Then we will create the basic structure of each React component and export it from that file with **export default**:

```
// src/App.js

import React from "react";
import About from "./components/About";
import Contact from "./components/Contact";
import Navbar from "./components/Navbar";
import Projects from "./components/Projects";
import Skills from "./components/Skills";
import Testimonials from "./components/Testimonials";

export default function App() {
  return (
    <main className="text-gray-400 bg-gray-900 body-font">
      <Navbar />
      <About />
      <Projects />
      <Skills />
      <Testimonials />
      <Contact />
    </main>
  );
}
```

# Building the Project Components

## About Section:

Let's start on our first section, the about section. This will consist of a basic introduction to ourselves and what skills we specialize in.

It's also going to include some links to the contact form as well as our past projects. Since these links will be to different parts of the same page, we can use the hashes: "/#projects" and "/#contact".

To make these links work and to be able to jump to each section, we will set the id attribute of the projects section to "projects" and those of the contact section to "contact".

function `fetchMoreData` which makes a `GET` request to the News API using the `fetch` function from the Fetch API.

```
import React from "react";

export default function About() {
  return (
    <section id="about">
      <div className="container mx-auto flex px-10 py-20 md:flex-row flex-col
items-center">
        <div className="lg:flex-grow md:w-1/2 lg:pr-24 md:pr-16 flex flex-col
md:items-start md:text-left mb-16 md:mb-0 items-center text-center">
          <h1 className="title-font sm:text-4xl text-3xl mb-4 font-medium
text-white">
            Hi, I'm Reed.
            <br className="hidden lg:inline-block" />I love to build amazing
            apps.
          </h1>
          <p className="mb-8 leading-relaxed">
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Qui
            laborum quasi, incidunt dolore iste nostrum cupiditate voluptas?
            Laborum, voluptas natus?
          </p>
          <div className="flex justify-center">
            <a
              href="#contact"
              className="inline-flex text-white bg-green-500 border-0 py-2 px-
6 focus:outline-none hover:bg-green-600 rounded text-lg">
              Work With Me
            </a>
            <a
              href="#projects"
              className="ml-4 inline-flex text-gray-400 bg-gray-800 border-0
py-2 px-6 focus:outline-none hover:bg-gray-700 hover:text-white rounded text-
lg">
              See My Past Work
            </a>
          </div>
        </div>
        <div className="lg:max-w-lg lg:w-full md:w-1/2 w-5/6">
          <img
            className="object-cover object-center rounded"
            alt="hero"
            src="./coding.svg"
          />
        </div>
      </div>
```

```
    </section>
  );
}
```

Our projects section will consist of a section element with an id of "projects". This will feature a gallery of all the projects that we've built, which will include images.
It'll have the title of the project, along with the technologies we use to make it, and a link to it (if it is deployed).

Let's fill out the section for all the skills and technologies that we know. This will consist of a simple list of all of the major tools that we're familiar with and can use in our employers or clients projects.

Once again, we are going to import an array from our data folder. But this array consists of number of strings which represent each of the skills that we know such as JavaScript, React, and Node:

```
import { BadgeCheckIcon, ChipIcon } from "@heroicons/react/solid";
import React from "react";
import { skills } from "../data";

export default function Skills() {
  return (
    <section id="skills">
      <div className="container px-5 py-10 mx-auto">
        <div className="text-center mb-20">
          <ChipIcon className="w-10 inline-block mb-4" />
          <h1 className="sm:text-4xl text-3xl font-medium title-font text-
white mb-4">
            Skills &amp;amp; Technologies
          </h1>
          <p className="text-base leading-relaxed xl:w-2/4 lg:w-3/4 mx-auto">
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Nisi sit
            ipsa delectus eum quo voluptas aspernatur accusantium distinctio
            possimus est.
          </p>
        </div>
        <div className="flex flex-wrap lg:w-4/5 sm:mx-auto sm:mb-2 -mx-2">
          {skills.map((skill) => (
            <div key={skill} className="p-2 sm:w-1/2 w-full">
              <div className="bg-gray-800 rounded flex p-4 h-full items-
center">
```

```
                <BadgeCheckIcon className="text-green-400 w-6 h-6 flex-shrink-
0 mr-4" />
                <span className="title-font font-medium text-white">
                  {skill}
                </span>
              </div>
            </div>
          ))}
        </div>
      </div>
    </section>
  );

}
```

## Testimonial Section:

In the Testimonials component, we are going to list a couple of testimonials maybe from past clients or people who are familiar with our work. These are going to consist of a couple of cards that feature the testimonial itself as well as who it's from and the company that this person is from.

We are also importing a testimonials array with a number of objects that feature the quote, image, name, and company.

At the end of our landing page, we're going to include our contact form to allow potential employers to reach out to us. This form will have 3 inputs: a name, email, and message input.

To receive these form submissions, we will use the tool Netlify Forms to very easily take care of saving those messages.

## Contact Section:

Once that's done, we'll head back to Contact.js. We're going to use JavaScript in order to submit this form. First of all, we're going to create some dedicated state for each of the values that are typed in the form for name, email, and message:

To handle submission of the form, we will add the onSubmit prop to it. The function that will be called, handleSubmit, will make a post request to the endpoint "/" with all of our form data.

We will set the headers of the request to indicate that we are sending over form data. For the request body, we will include the form name as well as all of the form data from the name, email,
and message state variables.

```
import React from "react";
export default function Contact() {
  const [name, setName] = React.useState("");
  const [email, setEmail] = React.useState("");
  const [message, setMessage] = React.useState("");

  function encode(data) {
    return Object.keys(data)
      .map(
        (key) => encodeURIComponent(key) + "=" + encodeURIComponent(data[key])
      )
      .join("&");
  }

  function handleSubmit(e) {
    e.preventDefault();
    fetch("/", {
      method: "POST",
      headers: { "Content-Type": "application/x-www-form-urlencoded" },
      body: encode({ "form-name": "contact", name, email, message }),
    })
      .then(() => alert("Message sent!"))
      .catch((error) => alert(error));
  }

  return (
    <section id="contact" className="relative">
      <div className="container px-5 py-10 mx-auto flex sm:flex-nowrap flex-
wrap">
        <div className="lg:w-2/3 md:w-1/2 bg-gray-900 rounded-lg overflow-
hidden sm:mr-10 p-10 flex items-end justify-start relative">
          <iframe
            width="100%"
            height="100%"
            title="map"
            className="absolute inset-0"
```

```
            frameBorder={0}
            marginHeight={0}
            marginWidth={0}
            style={{ filter: "opacity(0.7)" }}
            src="https://www.google.com/maps/embed/v1/place?q=97+warren+st+new
+york+city&key=AIzaSyBFw0Qbyq9zTFTd-tUY6dZWTgaQzuU17R8"
          />
          <div className="bg-gray-900 relative flex flex-wrap py-6 rounded
shadow-md">
            <div className="lg:w-1/2 px-6">
              <h2 className="title-font font-semibold text-white tracking-
widest text-xs">
                ADDRESS
              </h2>
              <p className="mt-1">
                97 Warren St. <br />
                New York, NY 10007
              </p>
            </div>
            <div className="lg:w-1/2 px-6 mt-4 lg:mt-0">
              <h2 className="title-font font-semibold text-white tracking-
widest text-xs">
                EMAIL
              </h2>
              <a className="text-indigo-400 leading-relaxed">
                reedbarger@email.com
              </a>
              <h2 className="title-font font-semibold text-white tracking-
widest text-xs mt-4">
                PHONE
              </h2>
              <p className="leading-relaxed">123-456-7890</p>
            </div>
          </div>
        </div>
        <form
          netlify
          name="contact"
          onSubmit={handleSubmit}
          className="lg:w-1/3 md:w-1/2 flex flex-col md:ml-auto w-full md:py-8
mt-8 md:mt-0">
          <h2 className="text-white sm:text-4xl text-3xl mb-1 font-medium
title-font">
            Hire Me
          </h2>
          <p className="leading-relaxed mb-5">
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Illum
            suscipit officia aspernatur veritatis. Asperiores, aliquid?
```

```
          </p>
          <div className="relative mb-4">
            <label htmlFor="name" className="leading-7 text-sm text-gray-400">
              Name
            </label>
            <input
              type="text"
              id="name"
              name="name"
              className="w-full bg-gray-800 rounded border border-gray-700
focus:border-indigo-500 focus:ring-2 focus:ring-indigo-900 text-base outline-
none text-gray-100 py-1 px-3 leading-8 transition-colors duration-200 ease-in-
out"
              onChange={(e) => setName(e.target.value)}
            />
          </div>
          <div className="relative mb-4">
            <label htmlFor="email" className="leading-7 text-sm text-gray-
400">
              Email
            </label>
            <input
              type="email"
              id="email"
              name="email"
              className="w-full bg-gray-800 rounded border border-gray-700
focus:border-indigo-500 focus:ring-2 focus:ring-indigo-900 text-base outline-
none text-gray-100 py-1 px-3 leading-8 transition-colors duration-200 ease-in-
out"
              onChange={(e) => setEmail(e.target.value)}
            />
          </div>
          <div className="relative mb-4">
            <label
              htmlFor="message"
              className="leading-7 text-sm text-gray-400">
              Message
            </label>
            <textarea
              id="message"
              name="message"
              className="w-full bg-gray-800 rounded border border-gray-700
focus:border-indigo-500 focus:ring-2 focus:ring-indigo-900 h-32 text-base
outline-none text-gray-100 py-1 px-3 resize-none leading-6 transition-colors
duration-200 ease-in-out"
              onChange={(e) => setMessage(e.target.value)}
            />
          </div>
```

```
          <button
            type="submit"
            className="text-white bg-indigo-500 border-0 py-2 px-6
focus:outline-none hover:bg-indigo-600 rounded text-lg">
            Submit
          </button>
        </form>
      </div>
    </section>
  );
}
```

# Adding Backend Node JS for Contact form submission

### Install Nodemailer and other dependencies

First run the following to initialize a **package.json** in the root directory of the project.

**npm init**

**Then install the following dependencies:**

- **Express: Node.js web framework**
- **Nodemailer: a module to send emails**
- **dotenv: allows use of environment variables**
- **multiparty: parses form data for Node.js apps**

**Install the dependencies by running:**

```
npm install express nodemailer dotenv multiparty
```

### Create server.js

In the root directory of your project, create a server.js file. In the first lines, we should import our dependencies:

```
const express = require("express");
const nodemailer = require("nodemailer");
const multiparty = require("multiparty");
require("dotenv").config();
```

And then initialize our app with express:

```
// instantiate an express app
const app = express();

//make the contact page the the first page on the app
app.route("/").get(function (req, res) {
  res.sendFile(process.cwd() + "/public/index.html");
});

//port will be 5000 for testing
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Listening on port ${PORT}...`);
});
```

## Set up Nodemailer and POST Route

Now all there's left to do is to set up the POST route to receive the submitted form data, parse it and send it via Nodemailer.

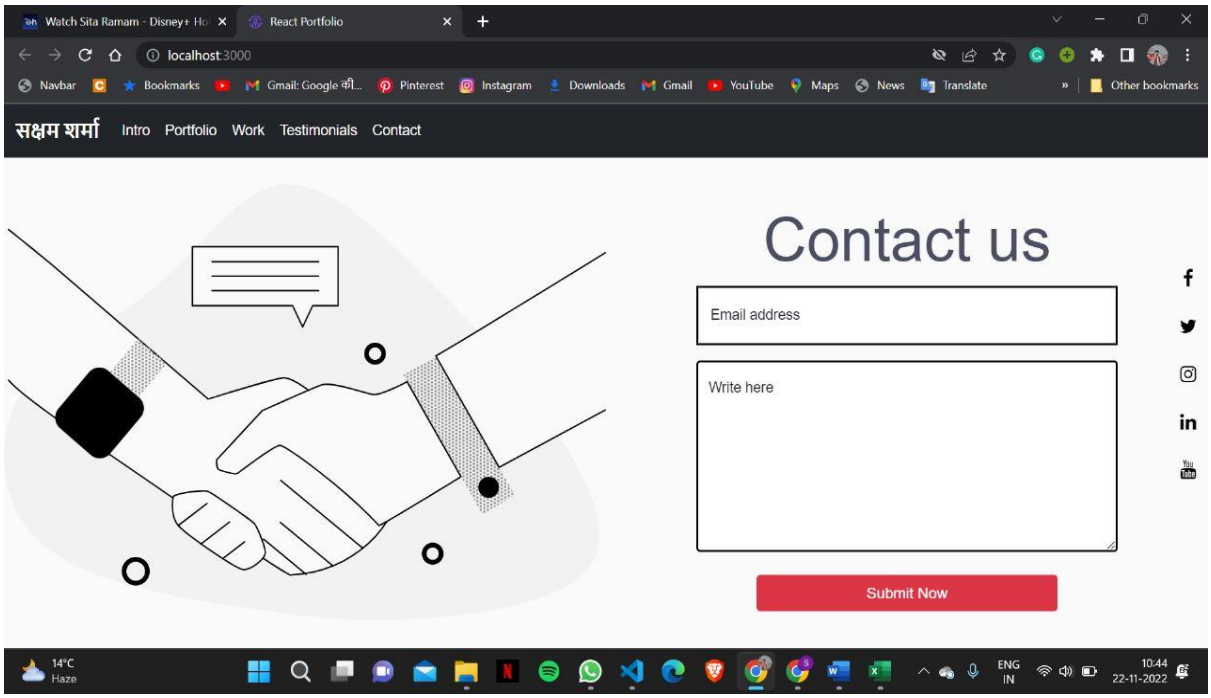
Now we have our transporter object. Next, we need to verify this connection to make the credentials are correct and Nodemailer is authorized to send emails from that address.

Finally, we create our POST route to do the following:

1. Accepts the form data submitted and parse it using **multiparty**.
2. After parsing it, create a **mail** object with **from**, **to**, **subject** and **text** properties.
3. Use **transporter.sendMail()** to send the email and done.

Code look like this::

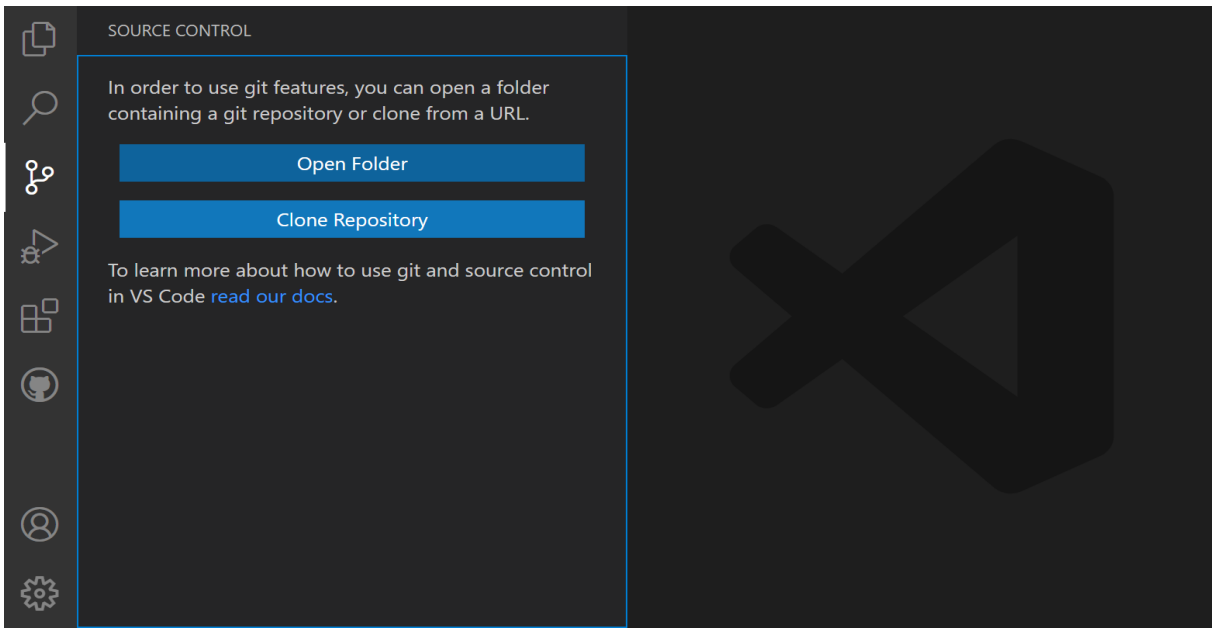
```
app.post("/send", (req, res) => {
  //1.
  let form = new multiparty.Form();
  let data = {};
  form.parse(req, function (err, fields) {
    console.log(fields);
    Object.keys(fields).forEach(function (property) {
      data[property] = fields[property].toString();
    });

    //2. You can configure the object however you want
    const mail = {
      from: data.name,
      to: process.env.EMAIL,
      subject: data.subject,
      text: `${data.name} <${data.email}> \n${data.message}`,
    };

    //3.
    transporter.sendMail(mail, (err, data) => {
      if (err) {
        console.log(err);
        res.status(500).send("Something went wrong.");
      } else {
        res.status(200).send("Email successfully sent to recipient!");
      }
    });
  });
});
```



# Deploying Your Portfolio

Now to make our portfolio live, we need to push our application to GitHub.

we can first create a new Github repository. After that, we will run git add ., git commit -m "Deploy", create our git remote, and git push -u origin master.
Once our project is on GitHub, we can head over to Netlify and select the option "Choose Site from Git". Then we will choose GitHub for our continuous deployment, and pick the GitHub repository to which we just pushed our code.



# SYSTEM REQUIREMENT

- OS: 64-bit Windows 7 or later or OS X 10.11 or later.
- Processor: 1.5GHz or faster.
- Memory: 4GB (4,096MB) RAM.
- Free HDD or SSD space: 3GB at least.
- Internet

# BIBLOGRAPHY

The various sites and platforms used during the creation of this project are given below:

- https://reactjs.org/
- https://getbootstrap.com/
- https://www.npmjs.com/
- https://codewithharry.com/
- https://youtube.com/
- https://www.w3schools.com/w3css/defaulT.asp

**Platform:**

- VS Code
- Git Hub
- YouTube