



COLLEGE CODE: 9111

COLLEGE NAME: SRM MADURAI

DEPARTMENT: B.E.CSE 3RD YEAR

STUDENT NM-ID:

70dce9f2f20d79e9ba24630db9eab6ab f98e24c41131947bf1448d94d60586c0 6a524ed8299c3188fe440fbd006889e8 e66ee7a9fdeac53a4a9408959cd37f7b 599943f7508888684994d0e4c72ce874

ROLL NO: 911123104009

911123104050

911123104023

911123104304

911123104013

DATE: 22/09/2025

Completed the project named as Phase 3 MVP Implementation

TECHNOLOGY PROJECT NAME: PORTFOLIO WEBSITES

SUBMITTED BY:

B.GOKULA GAURAV 6381360697
S.V.KARTHIVEL 7708803515
K.R.J.SREE KANTH 9944895465
R.V.SIVAPPRASATH 9488215506
K.GOWTHAM 8248027257

Portfolio Website Project with Code

1. Project Setup

Tools and Technologies

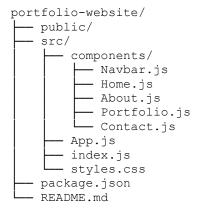
- React.js (Frontend framework)
- Node.js and npm (for package management)
- VSCode (Code editor)
- Git (Version control)

Initializing the Project

Use Create React App for quick setup:

```
npx create-react-app portfolio-website
cd portfolio-website
npm start
```

Folder Structure



2. Core Features Implementation

Navbar Component (Navigation)

Home Component

const handleSubmit = e => {
 e.preventDefault();

```
import React from 'react';
const Home = () => (
 <section id="home">
   <h1>Welcome to My Portfolio</h1>
   Frontend Developer | React Enthusiast
 </section>
);
export default Home;
About Component
import React from 'react';
const About = () => (
 <section id="about">
   <h2>About Me</h2>
   Passionate developer with experience in building responsive websites.
 </section>
);
export default About;
Portfolio Component (Image Gallery)
import React from 'react';
const projects = [
 { id: 1, title: 'Project One', img: 'https://via.placeholder.com/150' },
  { id: 2, title: 'Project Two', img: 'https://via.placeholder.com/150' },
const Portfolio = () => (
 <section id="portfolio">
   <h2>My Work</h2>
   <div className="gallery">
     {projects.map(project => (
        <div key={project.id} className="project-card">
         <img src={project.img} alt={project.title} />
         <h3>{project.title}</h3>
       </div>
     ) ) }
   </div>
 </section>
);
export default Portfolio;
Contact Component (Form with State Handling)
import React, { useState } from 'react';
const Contact = () => {
 const [formData, setFormData] = useState({ name: '', email: '', message: '' });
 const handleChange = e => {
   setFormData({ ...formData, [e.target.name]: e.target.value });
```

```
alert(`Thank you, ${formData.name}! Your message has been received.`);
    setFormData({ name: '', email: '', message: '' });
  return (
    <section id="contact">
      <h2>Contact Me</h2>
      <form onSubmit={handleSubmit}>
        <input name="name" value={formData.name} onChange={handleChange}</pre>
placeholder="Your Name" required />
        <input name="email" type="email" value={formData.email} onChange={handleChange}</pre>
placeholder="Your Email" required />
        <textarea name="message" value={formData.message} onChange={handleChange}</pre>
placeholder="Your Message" required />
       <button type="submit">Send</button>
      </form>
    </section>
 );
} ;
export default Contact;
```

App.js to bring it all together

```
import React from 'react';
import Navbar from './components/Navbar';
import Home from './components/Home';
import About from './components/About';
import Portfolio from './components/Portfolio';
import Contact from './components/Contact';
const App = () => (
  <>
   <Navbar />
   <main>
     <Home />
     <About />
     <Portfolio />
      <Contact />
    </main>
  </>
);
export default App;
```

3. Data Storage (Local State / Database)

Local State Example

Using React useState in the Contact form (already shown above).

Persisting form input with Local Storage

```
import React, { useState, useEffect } from 'react';

const Contact = () => {
  const [formData, setFormData] = useState(() => {
    const saved = localStorage.getItem('contactForm');
    return saved ? JSON.parse(saved) : { name: '', email: '', message: '' };
```

```
});
useEffect(() => {
  localStorage.setItem('contactForm', JSON.stringify(formData));
}, [formData]);

// rest of form logic remains same
};
```

Optional: Using Firebase for Data Storage

- Set up Firebase project
- Install Firebase SDK: npm install firebase
- Example to save contact form data to Firestore:

```
import { initializeApp } from 'firebase/app';
import { getFirestore, collection, addDoc } from 'firebase/firestore';

const firebaseConfig = {
    // Your config here
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

const handleSubmit = async e => {
    e.preventDefault();
    try {
       await addDoc(collection(db, 'contacts'), formData);
       alert('Message sent!');
    } catch (e) {
       alert('Error sending message.');
    }
};
```

4. Testing Core Features

Manual Testing

- Navigate through the site
- Submit contact form with valid and invalid inputs
- Check responsiveness on mobile and desktop

Unit Testing with Jest & React Testing Library

```
npm install --save-dev @testing-library/react jest

Example test for Contact form:
import { render, screen, fireEvent } from '@testing-library/react';
import Contact from './Contact';

test('renders contact form and submits', () => {
    render(<Contact />);
```

```
fireEvent.change(screen.getByPlaceholderText(/Your Name/i), { target: { value: 'John'
} });
fireEvent.change(screen.getByPlaceholderText(/Your Email/i), { target: { value:
'john@example.com' } });
fireEvent.change(screen.getByPlaceholderText(/Your Message/i), { target: { value:
'Hello!' } });
fireEvent.click(screen.getByPlaceholderText(/Your Name/i).value).toBe('');
expect(screen.getByPlaceholderText(/Your Name/i).value).toBe('');
});
```

5. Version Control (GitHub)

Initialize Git Repository

```
git init
git add .
git commit -m "Initial commit: Setup portfolio project with core components"
```

Create a GitHub repository and push code

```
git remote add origin https://github.com/yourusername/portfolio-website.git
git branch -M main
git push -u origin main
```

Best Practices

- Use clear, descriptive commit messages
- Create feature branches for new features: git checkout -b feature/contact-form
- Use Pull Requests to review and merge code
- Keep .gitignore updated (ignore node modules, build folders)