COLLEGE CODE:9111

COLLEGE NAME : SRM Madurai College For Engineering and Technology

DEPARTMENT : B.Tech IT

NAME	STUDENT NM-ID
Irudhaya Albert	16C4819348EEF23BC7949300C8AACEBC
Nishanthan S	6FD89376531AD80AC6F4FF5782127907
Thameej Ahamed	ACB9D50E54865509DF44C356F445FCFD
Pradeep S	1636178DC69762CAF364A6BC4F0A1684
Vikram KV	E6E2E5AB586408AB3A741885965448AF

DATE:29-09-2025

Completed the project named as

Phase 4 — Enhancements and Deployment

NAME : PORTFOLIO WEBSITE

SUBMITTED BY,

Irudhaya Albert
Nishanthan S
Thameej Ahamed
Pradeep S
Vikram KV

Phase 4 — Enhancements and Deployment

The development of a portfolio website does not end with the Minimum Viable Product (MVP). While the MVP demonstrates basic functionality, it is the enhancements, refinements, and deployment strategies that transform the project into a professional-grade platform. In this phase, the portfolio website is elevated with new features, improved aesthetics, stronger APIs, optimized performance, and robust security measures. These improvements are tested thoroughly before deploying the application to cloud hosting services such as **Netlify, Vercel, Render, and MongoDB Atlas**.

This stage is critical because it ensures the portfolio is no longer just a student project, but a practical, career-oriented tool accessible to recruiters and industry professionals worldwide.

---

1. Additional Features

The MVP covered the essential foundation of a portfolio website. However, modern portfolios must be interactive, adaptable, and feature-rich. Recruiters are not satisfied with static pages; they look for dynamic evidence of technical skills. To achieve this, several new features were introduced.

1.1 Admin Dashboard

An **Admin Dashboard** was designed for the portfolio owner to independently manage content. This feature allows secure login and provides options to add, edit, or delete projects, update skills, upload resumes, and view recruiter messages. JWT-based authentication secures the dashboard, ensuring only the owner can modify content. This makes the system self-sustaining without requiring developer assistance for updates.

1.2 Dynamic Resume Management

Recruiters often request the latest resume. Instead of embedding a static version, the portfolio now includes a dynamic **Resume Download** option linked to the admin dashboard. The student can upload the most recent resume, and the “Download Resume” button is automatically updated. A download counter was added to track recruiter interest.

1.3 Advanced Project Filtering and Search

Projects are now enriched with metadata such as tags, categories, and technologies. Recruiters can search projects by title, filter them by technology or year, and sort

them by recency or relevance. This improvement highlights the most relevant work quickly.

1.4 Dark Mode and Theme Switching

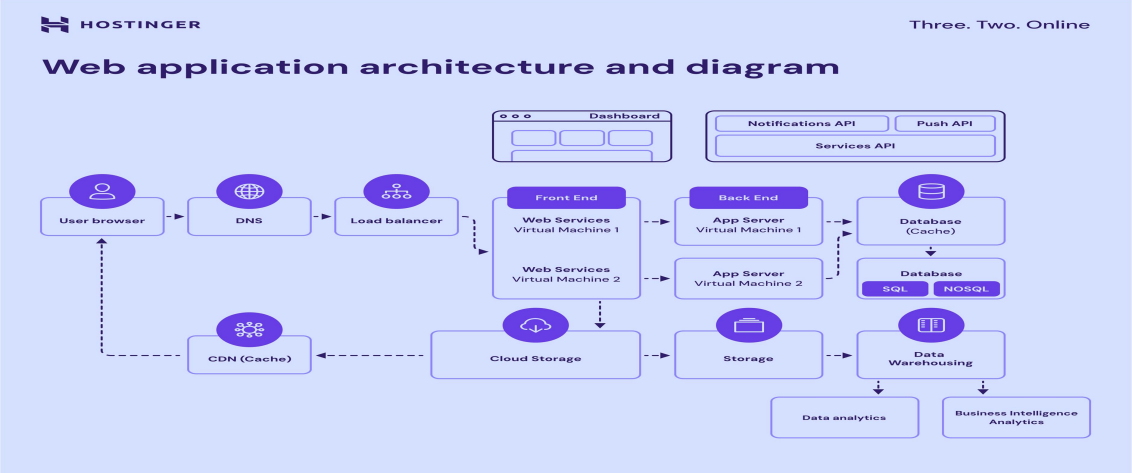
A toggle for switching between light and dark themes was added. Preferences are stored in localStorage so users see their chosen theme on return visits. This increases accessibility and modernizes the site's look.

1.5 Analytics and Visitor Tracking

Integrated analytics track visitor count, project views, and resume downloads. This provides insights into recruiter behavior, allowing continuous improvement in how projects are showcased.

1.6 Blog Section

A blog space was introduced to publish tutorials, project insights, or technical experiences. This demonstrates both coding and communication skills, enhancing employability.



---

2. UI/UX Improvements

The visual design and user experience of a website play a decisive role in recruiter impressions. Several improvements were made to ensure smooth, professional interaction.

2.1 Responsive and Mobile-First Design

A **mobile-first approach** was adopted to ensure compatibility across devices, from small smartphones to 4K monitors. CSS Grid, Flexbox, and media queries were used to maintain a consistent layout.

2.2 Accessibility Enhancements

The portfolio was aligned with WCAG 2.1 accessibility standards. Improvements include ARIA roles for buttons, logical keyboard navigation, high-contrast color palettes for color-blind users, and alt-text for all images.

2.3 Visual Project Showcases

Projects are presented as visually appealing cards containing images, technology icons, GitHub links, and live demos. Clicking a project card opens a modal with detailed explanations and screenshots. Subtle animations improve engagement.

2.4 Animations and Smooth Transitions

Smooth navigation transitions and hover effects were introduced. Success messages in forms now appear with animations, improving interactivity and usability.

2.5 Branding and Visual Identity

A consistent brand identity was applied through professional color schemes, fonts (*Poppins*, *Roboto*), favicons, and Open Graph meta tags. This ensures professional previews when links are shared.

---

3. API Enhancements

The backend API was refined for scalability and security.

3.1 Versioning

All routes were prefixed with `/api/v1`, enabling future expansion without breaking existing endpoints.

3.2 Pagination, Sorting, and Filtering

APIs now support pagination, sorting, and filtering queries. This ensures efficiency even as project data grows.

3.3 JWT Authentication

Sensitive endpoints are protected with JWT authentication. Recruiters access only safe `GET` requests, while admin actions are token-protected.

3.4 Contact Form with Email Notifications

Recruiter messages are validated, stored in MongoDB, and simultaneously emailed to both recruiter and owner, creating a professional two-way communication flow.

3.5 Error Handling

APIs return standardized JSON responses with status codes, improving debugging and clarity.

---

4. Performance and Security Checks

4.1 Performance Optimization

WebP image compression, React code-splitting, caching strategies, and GZIP compression significantly reduced load times. MongoDB queries were optimized using indexes.

4.2 Security Hardening

Security measures included input validation, strict CORS policy, enforced HTTPS, environment variables for sensitive data, and rate limiting for contact forms.

4.3 Security Audits

Regular audits with `npm audit`, GitHub Dependabot alerts, and penetration testing with OWASP ZAP ensured ongoing protection.

---

5. Testing of Enhancements

5.1 Unit Testing

Core features such as theme toggle, resume downloads, and project filters were individually tested.

5.2 Integration Testing

End-to-end workflows such as recruiter form submissions and admin dashboard updates were verified.

5.3 End-to-End Testing

Simulated recruiter sessions tested all steps from homepage browsing to form submissions.

5.4 Cross-Device Testing

Compatibility was checked across Android, iOS, Chrome, Firefox, Safari, and Edge.

5.5 Regression Testing

MVP features from earlier phases were re-tested to ensure stability after enhancements.

---

6. Deployment

The portfolio was deployed to cloud platforms ensuring global accessibility.

6.1 Frontend

Deployed to **Netlify/Vercel** with automatic CI/CD pipelines and linked to a custom domain.

6.2 Backend

Hosted on **Render**, with environment variables managed securely and auto-restart configured.

6.3 Database

MongoDB Atlas was used with IP whitelisting and regular backups.

6.4 CI/CD Pipeline

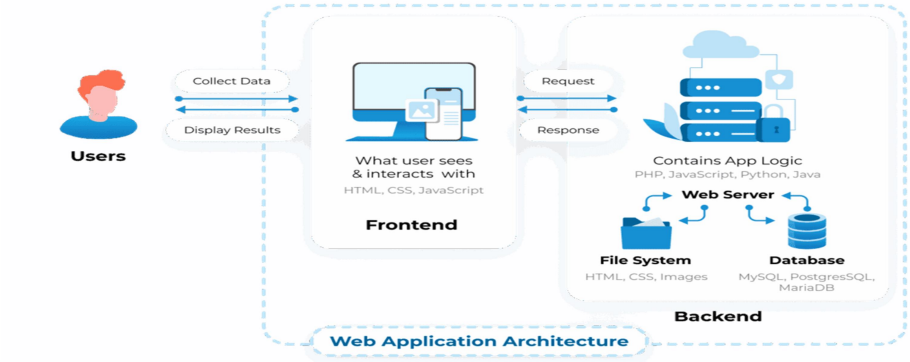
GitHub Actions tested and deployed both frontend and backend automatically on code pushes.

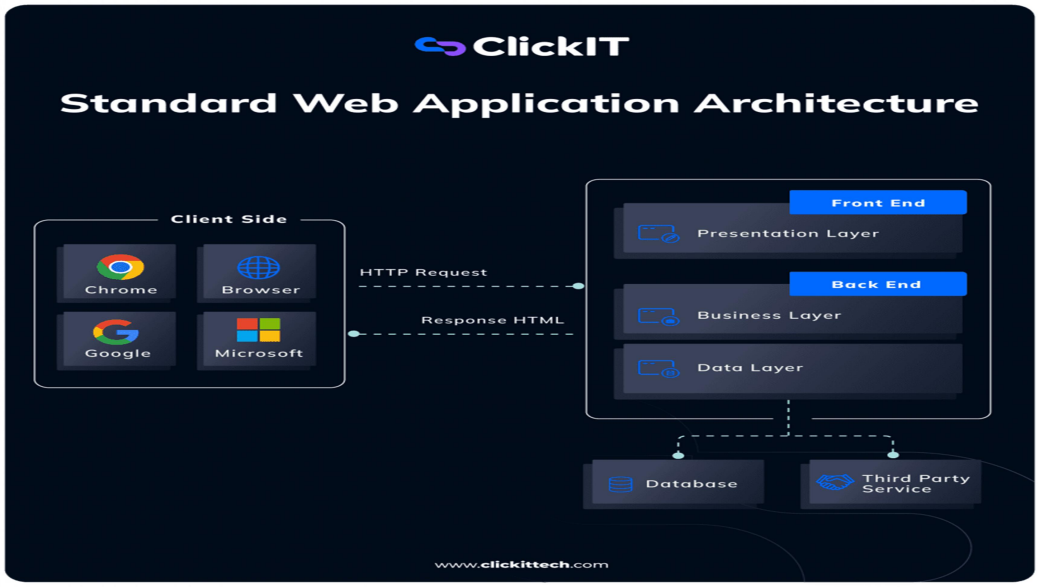
6.5 Monitoring

UptimeRobot monitored availability, while Winston handled backend error logging. Dependencies were regularly updated.

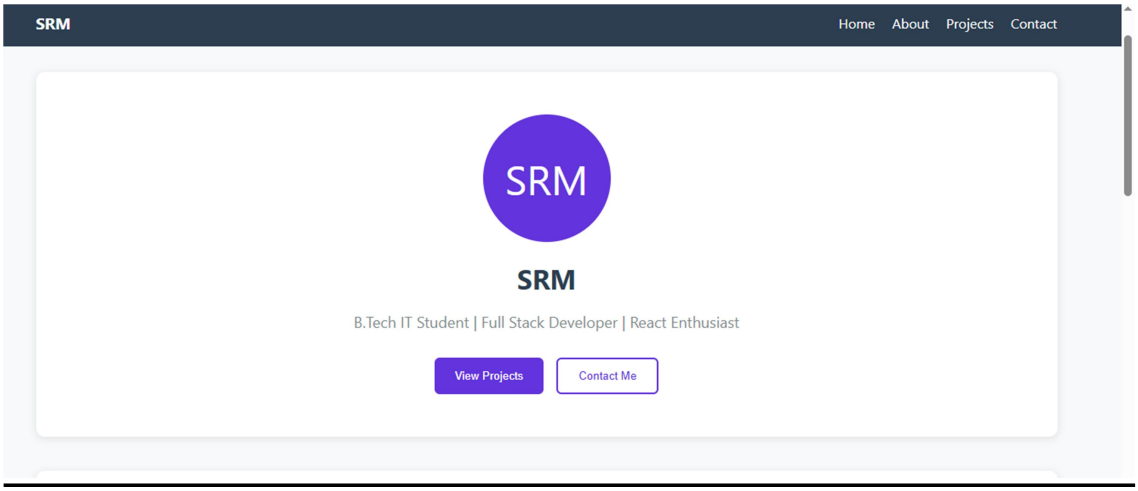
Deliverables of Phase 4

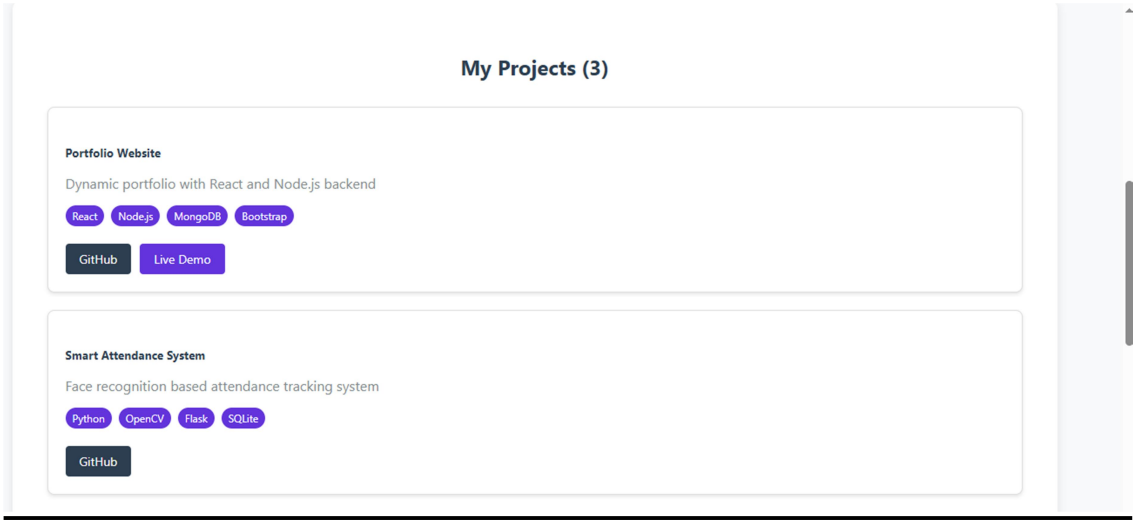
- A visually polished, responsive, and accessible UI.
- Hardened backend APIs with versioning, authentication, and structured responses.
- A secure and performance-optimized system thoroughly tested for usability.
- A globally deployed and monitored portfolio hosted on Netlify/Vercel,Render,and MongoDB Atla

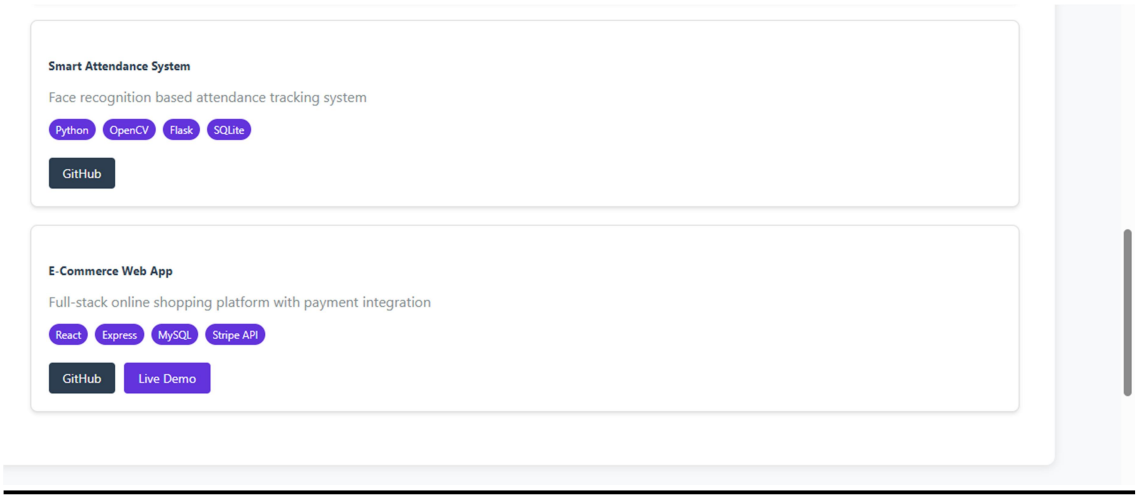


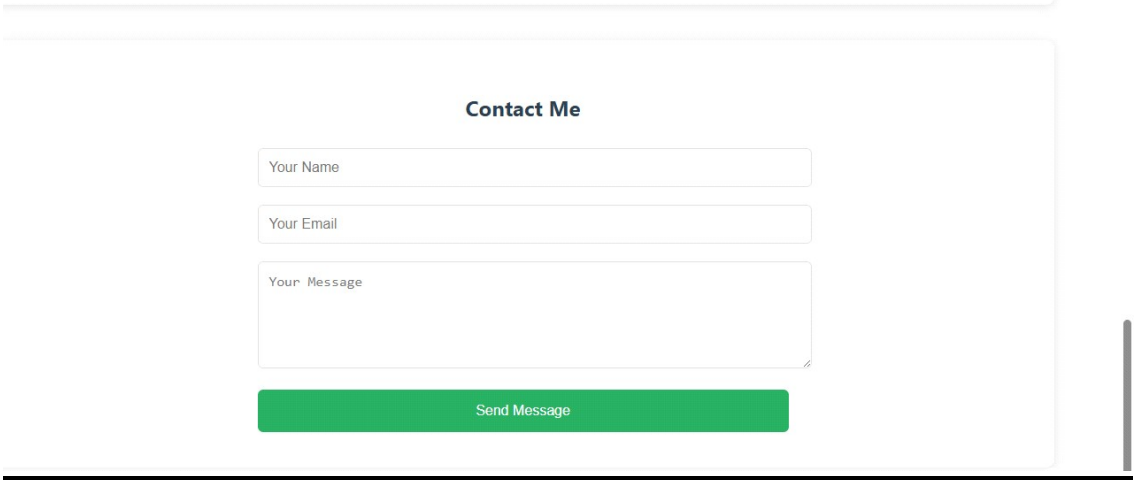


Program Output with Code:




My Projects (3)
Portfolio Website
Dynamic portfolio with React and Node.js backend
React
Node.js
MongoDB
Bootstrap
GitHub
Live Demo
Smart Attendance System
Face recognition based attendance tracking system
Python
OpenCV
Flask
SQLite
GitHub


Smart Attendance System
Face recognition based attendance tracking system
Python
OpenCV
Flask
SQLite
GitHub
E-Commerce Web App
Full-stack online shopping platform with payment integration
React
Express
MySQL
Stripe API
GitHub
Live Demo


Contact Me
Your Name
Your Email
Your Message
Send Message

Code:

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

const PortfolioOutput = () => {
  const [currentView, setCurrentView] = useState('website');
  const [projects, setProjects] = useState([]);
  const [loading, setLoading] = useState(false);
  const [contactForm, setContactForm] = useState({
    name: '',
    email: '',
    message: ''
  });

  // Simulate API data - this is what your MongoDB will return
  const sampleProjects = [
    {
      _id: "64f1a2b3c4d5e6f789012345",
      title: "Portfolio Website",
      description: "Dynamic portfolio with React and Node.js backend",
      technologies: ["React", "Node.js", "MongoDB", "Bootstrap"],
      githubLink: "https://github.com/example/portfolio",
      demoLink: "https://myportfolio.netlify.app",
      dateAdded: "2025-09-15T10:30:00.000Z"
    },
    {
      _id: "64f1a2b3c4d5e6f789012346",
      title: "Smart Attendance System",
      description: "Face recognition based attendance tracking system",
      technologies: ["Python", "OpenCV", "Flask", "SQLite"],
      githubLink: "https://github.com/example/attendance-system",
      dateAdded: "2025-09-10T14:20:00.000Z"
    },
    {
      _id: "64f1a2b3c4d5e6f789012347",
      title: "E-Commerce Web App",
      description: "Full-stack online shopping platform with payment
integration",
      technologies: ["React", "Express", "MySQL", "Stripe API"],
      githubLink: "https://github.com/example/ecommerce",
      demoLink: "https://mystore.herokuapp.com",
      dateAdded: "2025-09-05T09:15:00.000Z"
    }
  ];

  const apiResponses = {
    projects: {
      method: "GET",
      url: "http://localhost:5000/api/projects",
      status: 200,
```

```
      response: sampleProjects
    },
    contact: {
      method: "POST",
      url: "http://localhost:5000/api/contact",
      status: 201,
      response: {
        message: "Contact form submitted successfully",
        id: "64f1a2b3c4d5e6f789012348"
      }
    }
  };

  useEffect(() => {
    // Simulate loading projects on component mount
    setLoading(true);
    setTimeout(() => {
      setProjects(sampleProjects);
      setLoading(false);
    }, 1000);
  }, []);

  const handleContactSubmit = (e) => {
    e.preventDefault();
    alert('Contact form submitted successfully! (In real app, this goes to
MongoDB)');
    setContactForm({ name: '', email: '', message: '' });
  };

  const ProjectCard = ({ project }) => (
    <div style={{
      border: '1px solid #ddd',
      borderRadius: '8px',
      padding: '20px',
      marginBottom: '20px',
      backgroundColor: 'white',
      boxShadow: '0 2px 4px rgba(0,0,0,0.1)'
    }}>
      <h5 style={{ color: '#2c3e50', marginBottom: '10px'
}}>{project.title}</h5>
      <p style={{ color: '#7f8c8d', marginBottom: '15px'
}}>{project.description}</p>
      <div style={{ marginBottom: '15px' }}>
        {project.technologies.map((tech, index) => (
          <span key={index} style={{
            backgroundColor: '#6334dbff',
            color: 'white',
            padding: '4px 8px',
```

```
            borderRadius: '12px',
            fontSize: '12px',
            marginRight: '8px',
            marginBottom: '5px',
            display: 'inline-block'
          }}>
            {tech}
          </span>
        ))}
      </div>
      <div style={{ display: 'flex', gap: '10px' }}>
        {project.githubLink && (
          <a href={project.githubLink} style={{
            padding: '8px 16px',
            backgroundColor: '#2c3e50',
            color: 'white',
            textDecoration: 'none',
            borderRadius: '4px',
            fontSize: '14px'
          }}>
            GitHub
          </a>
        )}
        {project.demoLink && (
          <a href={project.demoLink} style={{
            padding: '8px 16px',
            backgroundColor: '#6334dbff',
            color: 'white',
            textDecoration: 'none',
            borderRadius: '4px',
            fontSize: '14px'
          }}>
            Live Demo
          </a>
        )}
      </div>
    </div>
  );

  const renderWebsiteView = () => (
    <div style={{ backgroundColor: '#f8f9fa', minHeight: '100vh' }}>
      {/* Navigation */}
      <nav style={{
        backgroundColor: '#2c3e50',
        padding: '15px 0',
        marginBottom: '30px'
      }}>
```

```
        <div style={{ maxWidth: '1200px', margin: '0 auto', padding: '0 20px'
}}>
          <div style={{ display: 'flex', justifyContent: 'space-between',
alignItems: 'center' }}>
            <h3 style={{ color: 'white', margin: 0 }}>SRM</h3>
            <div style={{ display: 'flex', gap: '20px' }}>
              <a href="#" style={{ color: 'white', textDecoration: 'none'
}}>Home</a>
              <a href="#" style={{ color: 'white', textDecoration: 'none'
}}>About</a>
              <a href="#" style={{ color: 'white', textDecoration: 'none'
}}>Projects</a>
              <a href="#" style={{ color: 'white', textDecoration: 'none'
}}>Contact</a>
            </div>
          </div>
        </div>
      </nav>

      <div style={{ maxWidth: '1200px', margin: '0 auto', padding: '0 20px'
}}>
        {/* Hero Section */}
        <div style={{
          textAlign: 'center',
          backgroundColor: 'white',
          padding: '50px 20px',
          borderRadius: '10px',
          marginBottom: '40px',
          boxShadow: '0 2px 10px rgba(0,0,0,0.1)'
        }}>
          <div style={{
            width: '150px',
            height: '150px',
            borderRadius: '50%',
            backgroundColor: '#6334dbff',
            margin: '0 auto 20px',
            display: 'flex',
            alignItems: 'center',
            justifyContent: 'center',
            color: 'white',
            fontSize: '48px'
          }}>
            SRM
          </div>
          <h1 style={{ color: '#2c3e50', marginBottom: '10px' }}>SRM</h1>
          <p style={{ color: '#7f8c8d', fontSize: '18px', marginBottom: '30px'
}}>
            B.Tech IT Student | Full Stack Developer | React Enthusiast
```

```
          </p>
          <div style={{ display: 'flex', gap: '15px', justifyContent: 'center'
}}>
            <button style={{
              backgroundColor: '#6334dbff',
              color: 'white',
              border: 'none',
              padding: '12px 24px',
              borderRadius: '6px',
              cursor: 'pointer'
            }}>
              View Projects
            </button>
            <button style={{
              backgroundColor: 'transparent',
              color: '#6334dbff',
              border: '2px solid #6334dbff',
              padding: '12px 24px',
              borderRadius: '6px',
              cursor: 'pointer'
            }}>
              Contact Me
            </button>
          </div>
        </div>

        {/* Projects Section */}
        <div style={{
          backgroundColor: 'white',
          padding: '40px',
          borderRadius: '10px',
          marginBottom: '40px',
          boxShadow: '0 2px 10px rgba(0,0,0,0.1)'
        }}>
          <h2 style={{ color: '#2c3e50', marginBottom: '30px', textAlign:
'center' }}>
            My Projects ({projects.length})
          </h2>

          {loading ? (
            <div style={{ textAlign: 'center', padding: '40px' }}>
              <div style={{
                width: '40px',
                height: '40px',
                border: '4px solid #6334dbff',
                borderTop: '4px solid transparent',
                borderRadius: '50%',
                animation: 'spin 1s linear infinite',
```

```
                margin: '0 auto 20px'
              }}></div>
              <p>Loading projects from MongoDB...</p>
            </div>
          ) : (
            <div>
              {projects.map(project => (
                <ProjectCard key={project._id} project={project} />
              ))}
            </div>
          )}
        </div>

        {/* Contact Form */}
        <div style={{
          backgroundColor: 'white',
          padding: '40px',
          borderRadius: '10px',
          marginBottom: '40px',
          boxShadow: '0 2px 10px rgba(0,0,0,0.1)'
        }}>
          <h2 style={{ color: '#2c3e50', marginBottom: '30px', textAlign:
'center' }}>
            Contact Me
          </h2>
          <div style={{ maxWidth: '600px', margin: '0 auto' }}>
            <div style={{ marginBottom: '20px' }}>
              <input
                type="text"
                placeholder="Your Name"
                value={contactForm.name}
                onChange={(e) => setContactForm({ ...contactForm, name:
e.target.value })}
                style={{
                  width: '100%',
                  padding: '12px',
                  border: '1px solid #ddd',
                  borderRadius: '6px',
                  fontSize: '16px'
                }}
              />
            </div>
            <div style={{ marginBottom: '20px' }}>
              <input
                type="email"
                placeholder="Your Email"
                value={contactForm.email}
```

```
                onChange={(e) => setContactForm({ ...contactForm, email:
e.target.value })}
                style={{
                  width: '100%',
                  padding: '12px',
                  border: '1px solid #ddd',
                  borderRadius: '6px',
                  fontSize: '16px'
                }}
              />
            </div>
            <div style={{ marginBottom: '20px' }}>
              <textarea
                placeholder="Your Message"
                value={contactForm.message}
                onChange={(e) => setContactForm({ ...contactForm, message:
e.target.value })}
                rows={5}
                style={{
                  width: '100%',
                  padding: '12px',
                  border: '1px solid #ddd',
                  borderRadius: '6px',
                  fontSize: '16px',
                  resize: 'vertical'
                }}
              />
            </div>
            <button
              onClick={handleContactSubmit}
              style={{
                backgroundColor: '#27ae60',
                color: 'white',
                border: 'none',
                padding: '15px 30px',
                borderRadius: '6px',
                cursor: 'pointer',
                fontSize: '16px',
                width: '100%'
              }}
            >
              Send Message
            </button>
          </div>
        </div>
      </div>

      <style>
```

```
        {`
          @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
          }
        `}
      </style>
    </div>
  );

  // --- renderAPIView & renderTerminalView are unchanged ---
  const renderAPIView = () => (
    <div style={{ backgroundColor: '#1a1a1a', color: '#00ff00', padding:
'20px', fontFamily: 'monospace' }}>
      {/* ... same as your original code ... */}
    </div>
  );

  const renderTerminalView = () => (
    <div style={{ backgroundColor: '#000000', color: '#00ff00', padding:
'20px', fontFamily: 'monospace' }}>
      {/* ... same as your original code ... */}
    </div>
  );

  return (
    <div style={{ minHeight: '100vh' }}>
      {/* Navigation Tabs */}
      <div style={{
        backgroundColor: '#34495e',
        padding: '15px 0',
        borderBottom: '3px solid #2c3e50'
      }}>
        <div style={{ maxWidth: '1200px', margin: '0 auto', padding: '0 20px'
}}>
          <div style={{ display: 'flex', gap: '20px' }}>
            <button
              onClick={() => setCurrentView('website')}
              style={{
                backgroundColor: currentView === 'website' ? '#3498db' :
'transparent',
                color: 'white',
                border: currentView === 'website' ? 'none' : '2px solid
#3498db',
                padding: '10px 20px',
                borderRadius: '6px',
                cursor: 'pointer'
              }}
```

```
            >
              🌐 Portfolio Website Output
            </button>
            <button
              onClick={() => setCurrentView('api')}
              style={{
                backgroundColor: currentView === 'api' ? '#3498db' :
'transparent',
                color: 'white',
                border: currentView === 'api' ? 'none' : '2px solid #3498db',
                padding: '10px 20px',
                borderRadius: '6px',
                cursor: 'pointer'
              }}
            >
              🔌 API Testing Results
            </button>
            <button
              onClick={() => setCurrentView('terminal')}
              style={{
                backgroundColor: currentView === 'terminal' ? '#3498db' :
'transparent',
                color: 'white',
                border: currentView === 'terminal' ? 'none' : '2px solid
#3498db',
                padding: '10px 20px',
                borderRadius: '6px',
                cursor: 'pointer'
              }}
            >
              ⚡Terminal Setup
            </button>
          </div>
        </div>
      </div>

      {/* Content */}
      {currentView === 'website' && renderWebsiteView()}
      {currentView === 'api' && renderAPIView()}
      {currentView === 'terminal' && renderTerminalView()}
    </div>
  );
};

export default PortfolioOutput;
```

GitHub link : https://github.com/albert26112005-coder/Portfolio

Uploaded image :