COLLEGE CODE : 9509

COLLEGE NAME:HOLYCROSS ENGINEERING COLLEGE

DEPARTMENT:CSE

STUDENT NM-ID: ac62f0521eb1c36ff94a355b46b2b4a

Roll No:950923104043

Date: 15.09.2025

Completed the project named as Phase 2

TECHNOLOGY PROJECT NAME:IBM-FE-Protfolio website

Submitted by,

Name: J.Rose Vincy

Mobile No: 8056498511

Phase 2 — Portfolio Website

1. Tech Stack Selection

Choosing the right stack ensures scalability, maintainability, and performance.

Frontend:
- Framework: React.js (or Next.js for server-side rendering and SEO benefits)
- Styling: Tailwind CSS or SCSS for responsive and modern UI
- Animations: Framer Motion for smooth transitions

Backend:
- Framework: Node.js with Express.js (REST API)
- Alternative: Next.js API routes (if backend is minimal)

Database:
- MongoDB (NoSQL for flexible project/portfolio data storage)
- Alternative: PostgreSQL (if relational structure is needed)

Hosting / Deployment:
- Frontend: Vercel / Netlify
- Backend + DB: Render / Railway / MongoDB Atlas

Version Control & Collaboration:
- Git + GitHub

2. UI Structure / API Schema Design

UI Structure (Pages & Sections):
1. Home / Landing Page – Introduction, headline, CTA
2. About Me – Bio, skills, education
3. Projects – Showcasing portfolio projects with images, descriptions, and links
4. Experience / Resume – Work history, certifications
5. Contact – Form + social links

API Schema Design (Example JSON):

Projects API:

{
"id": "p001",
"title": "Portfolio Website",
"description": "A personal portfolio showcasing my projects and skills.",
"techStack": ["React", "Node.js", "MongoDB"],
"githubLink": "https://github.com/user/portfolio",
"liveDemo": "https://portfolio.vercel.app"
}

User API:

{
"id": "u001",
"name": "John Doe",
"role": "Full Stack Developer",
"skills": ["JavaScript", "React", "Node.js", "Python"],
"email": "johndoe@email.com"
}

3. Data Handling Approach

Frontend: Fetch data via REST API using Axios/Fetch
State Management: Context API or Redux Toolkit for handling global state

Backend:
- CRUD operations for Projects, Skills, and User Profile
- Data validation using Joi or Yup

Database: Store structured data (projects, users, messages)

4. Component / Module Diagram

[Frontend (React)]
|-- Home Component
|-- About Component
|-- Projects Component
|-- Contact Component
|-- Navbar / Footer
|
v
[Backend (Node.js / Express)]
|-- User Module
|-- Projects Module
|-- Contact Module
|
v
[Database (MongoDB)]
|-- users collection
|-- projects collection
|-- messages collection

5. Basic Flow Diagram

User -> Browser (React UI)
-> API Request (Fetch/Axios)
-> Backend (Express.js)
-> Query Database (MongoDB)
<- Return JSON Response
<- Render Data on UI

Contact Form Example Flow:
User submits → Request hits backend → Backend validates & saves to DB → Email notification → Success message displayed.