COLLEGE CODE: 9504

COLLEGE NAME: Dr.G.U.POPE COLLEGE OF ENGINEERING

DEPARTMENT: CSE

STUDENT NM-ID: 9312D5145FBDD048FB642D48275CE68D

ROLL NO.: 22

COMPLETED THE PHASE III

“PORTFOLIO WEBSITE”

SUBMITTED BY,

NAME: Malaiarasi M

MOBILE NO.:8940496383

MVP Implementation

Project Setup:

1. Project Title & Description

   - Name of the project.
   - One or two lines about what it does.

2. Tools & Tech Stack

   - Programming languages (Python, Java, JS, etc.)
   - Frameworks (React, Django, Node.js, etc.)
   - Database (MySQL, MongoDB, etc.)
   - Other tools (Git, VS Code, Docker, etc.)

3. Folder / File Structure

Show how your project files are arranged.
Example (React app):

```
my-app/
├── src/
│   ├── components/
│   ├── pages/
│   └── App.js
├── public/
└── package.json
```

4. Installation / Setup Instructions

Explain step by step how to run your project.

Example:

Clone repo

git clone https://github.com/username/project.git

Go to folder

cd project

Install dependencies

npm install

Start project

npm start

5. Initial Configurations
   - Environment setup (Node.js version, Python venv, etc.)
   - API keys or .env usage (without exposing secrets).
   - Any database setup (create tables, migrations).
6. Basic Flow Diagram (Optional but good)

Show how data flows or modules connect at the start.

Project Setup – Weather App

- Tech Stack: React, OpenWeather API, Tailwind CSS
- Folder Structure: Organized into components, pages, and services.

- Installation: npm install, then npm start.
- Config: API key stored in .env.
- Flow: User enters city → API call → Weather data displayed.

Core Features Implementation

WhattoInclude in"Core FeaturesImplementation"

1. List the Core Features

- Write down the key things your project can do.
  Example (E-Commerce App):

- User login & signup
- Product catalog browsing
- Add to cart & checkout
- Payment integration

2. Explain Each Feature

For each core feature, explain how you built it (tools, logic, libraries).
Example:

- User Authentication: Implemented using Firebase Auth with JWT for secure login.
- Cart Management: Used Redux to handle state and keep items even after refresh.

3. Show Snippets / Screenshots

- Add small code snippets (not the full code).
- Or screenshots of working features.

Example (Weather App – Fetch Weather Feature):

```
// Fetch weather data using API
async function getWeather(city) {
 constresponse=awaitfetch(

` https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}`
 );
 const data = await response.json();
 return data;
}
```

4. Workflow / Flow Diagram (Optional)

- Show how the feature works step by step.
  Example:
  User → Enter City → API Call → Get Data → Show Weather

Example (Portfolio Content)

Core Features Implementation – Weather App

1. Search City & Display Weather:

- Implemented using OpenWeather API.
- Fetches live weather data and shows temperature, humidity, and conditions.

2. Recent Search History:
   - Stored searched cities in localStorage.
   - Displays last 5 searched cities for quick access.
3. Responsive UI:
   - Built with Tailwind CSS.
   - Works smoothly on both mobile and desktop.

Data Storage (Local State / Database)

1. Local State (Frontend Apps)

If your project is frontend-heavy (React, Angular, Vue):

- Mention where you store temporary data (e.g., state variables, context, Redux).
- Explain how state updates affect the UI.

Example (React Weather App):

- Used useState to store user input and weather data.
- Used localStorage to save last 5 searched cities for persistence.

2. Database (Backend Apps)

If your project has backend/database:

- Mention which database you used (SQL / NoSQL).
- Show a simple schema / table structure.
- Explain how CRUD operations (Create, Read, Update, Delete) are implemented.

Example (E-Commerce App with MongoDB):


```
{
 "user": {
  "id": "001",
  "name": "John",
  "email": "john@email.com"
 },
 "product": {
  "id": "P101",
  "title": "Laptop",
  "price": 55000
 },
 "cart": {
  "userId": "001",
  "products": ["P101"]
 }
```

3. Hybrid Approach

Some projects use both local state & database.
Example:

- Local state → To manage temporary data like form inputs, toggle states, UI filters.
- Database → To persist long-term data like user accounts, orders, or reports.

Example (Portfolio Content)

Data Storage – Weather App

- Local State:
  - o Managed weather data and search input using React useState.
  - o Stored recent city searches in localStorage for persistence.
- Database (Optional Extension):
  - o Designed a MongoDB schema to store user preferences (favorite cities).
  - o API endpoints allow saving/retrieving cities for each user.

Data Storage – E-Commerce App

- Local State:

- Used Redux for cart management, ensuring items persist across pages.

- Database:
  - Used MongoDB to store users, products, and orders.
  - Implemented CRUD operations via Express.js API.

Testing Core Features

1. Testing Methods Used

- Manual Testing → Checking each feature by hand.
- Automated Testing → Using frameworks (e.g., Jest, PyTest, Mocha).
- Unit Testing → Testing small pieces of code (functions, components).
- Integration Testing → Testing how different modules work together.
- End-to-End (E2E) Testing → Simulating real user actions.

2. Test Cases / Scenarios

Write test cases for your core features.

Example (Weather App):

- Input a valid city → Weather data should display.

- Input an invalid city → Show “City not found” message.
- Recent searches should be saved in localStorage.

Example (E-Commerce App):

- Add product to cart → Cart count increases.
- Checkout with valid payment → Order confirmation shown.
- Checkout without login → Redirect to login page.

3. Code Snippets (Optional)

Show a small test example.

Example (React Weather App – Jest Test):

```
test("fetches weather data for valid city", async () => {
  const data = await getWeather("London");
  expect(data.name).toBe("London");
});
```

Example (Python – PyTest):

```
def test_add_numbers():
    assert add(2, 3) == 5
```

4. Bug Fixes & Improvements

Briefly mention:

- What bugs you found during testing.

- How you fixed them.

Example (Portfolio Content)

Testing Core Features – Weather App

- Methods: Performed unit testing using Jest for API functions and manual testing for UI behavior.
- Test Cases:
    - Valid city shows correct weather.
    - Invalid city displays error message.
    - Recent search history persists in localStorage.
- Sample Test: Wrote Jest tests to check if API fetch returns correct city data.
- Result: Fixed a bug where app crashed on empty input by adding input validation.

Testing Core Features – E-Commerce App

- Methods: Manual + automated testing (Mocha + Chai).
- Test Cases:
    - Cart updates correctly when adding/removing products.
    - Login required for checkout.
    - Order saved in database after payment success.

. Bug Fix: Fixed issue where cart items were not persisting after page refresh (solved using Redux + localStorage).

Version Control (GitHub)

1. Repository Setup

- Mention that you created a GitHub repo for your project.
- Add a screenshot or link to the repository.

Example:

- Repo Link: github.com/username/weather-app

2. Branching Strategy

- Explain how you managed branches.
- Example:
    - o main → Stable production-ready code
    - o dev → Development branch
    - o feature/* → Feature-specific branches

3. Commit Practices

- Show that you used meaningful commit messages.
- Example:
    - o feat: add weather API integration
    - o fix: handle invalid city error

- style: improve UI responsivenes

4. Collaboration (if team project)

- Mention Pull Requests (PRs), Code Reviews, and Merging process.

5. GitHub Features Used

- Issues → For tracking bugs.
- Projects/Boards → For task management.
- Actions (CI/CD) → For automated builds/tests.

Example (Portfolio Content)

Version Control (GitHub) – Weather App

- Created a GitHub repository to manage codebase.
- Followed a branching strategy with main for stable releases and dev for active development.
- Used meaningful commit messages (e.g., feat: add search history feature).
- Managed issues and tasks using GitHub Projects board.
- Implemented Pull Requests for reviewing code before merging into main.

Version Control (GitHub) – E-Commerce App

- All code maintained in a GitHub repository: github.com/username/ecommerce-app

- Branches used:
    - main → Production-ready
    - feature/cart → Cart functionality
    - feature/payment → Payment integration
- Used GitHub Actions for automated testing on every push.
- Collaborated with teammates via Pull Requests & Code Reviews.