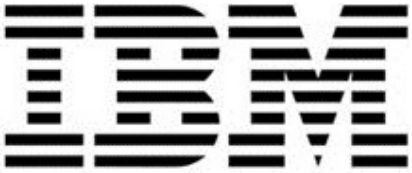


COLLEGE CODE: 9504

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

DEPARTMENT: CSE

STUDENT NM-ID: 22D4F7A708B89F78ABC4416E9A31BCD2

ROLL NO.: 26

COMPLETED THE PHASE I
“PORTFOLIO WEBSITE”

SUBMITTED BY,

NAME: Navitha U

MOBILE NO. : 9047627313

Users & Stakeholders

"Users & Stakeholders" usually means identifying who will use a system/product and who has an interest or influence over it.

Users

These are the people who directly interact with the system/product.
Examples:

- **End Users** – the primary people using it (students, employees, customers).
- **Administrators** – manage, configure, and maintain the system.
- **Support Staff** – help troubleshoot or provide assistance.

Stakeholders

These are people or groups who may not directly use the system but are **affected by it or have influence over it**.

Examples:

- **Project Sponsor / Management** – fund and approve the project.

- **Clients / Customers** – those for whom the system is developed.
- **Regulatory Bodies** – ensure compliance with laws and standards.
- **Developers / Designers** – build and maintain the system.
- **Investors** – expect returns or benefits from the system.

A quick way to remember:

- Users = direct interaction.
- Stakeholders = direct + indirect interest (influence, investment, approval).

Quick Table for Assignment

Category	Examples	Role
Users	Student, Teacher, Admin, Customer	Directly use the system
Stakeholders	Sponsor, Client, Regulator, Developer, Investor	Have interest/influence, may not use directly

Example program:

```
<!DOCTYPE html>
<html>
<head>
  <title>Users & Stakeholders</title>
</head>
<body>
  <h2>=== USERS ===</h2>
  <ul id="usersList"></ul>

  <h2>=== STAKEHOLDERS ===</h2>
  <ul id="stakeholdersList"></ul>

  <script>
    const users = ["Student", "Teacher", "Administrator",
"Customer"];
    const stakeholders = ["Project Sponsor", "Client",
"Regulatory Body", "Developer", "Investor"];

    const usersList =
document.getElementById("usersList");
```

```
    const stakeholdersList =
document.getElementById("stakeholdersList");

    users.forEach(u => {
      let li = document.createElement("li");
      li.textContent = u + " (Direct system interaction)";
      usersList.appendChild(li);
    });

    stakeholders.forEach(s => {
      let li = document.createElement("li");
      li.textContent = s + " (Has interest/influence in
system success)";
      stakeholdersList.appendChild(li);
    });
  </script>
</body>
</html>
```

Output:

=== USERS ===

- Student (Direct system interaction)
- Teacher (Direct system interaction)
- Administrator (Direct system interaction)
- Customer (Direct system interaction)

=== STAKEHOLDERS ===

- Project Sponsor (Has interest/influence in system success)
- Client (Has interest/influence in system success)
- Regulatory Body (Has interest/influence in system success)
- Developer (Has interest/influence in system success)
- Investor (Has interest/influence in system success)

User stories

What are User Stories?

👉 A **user story** describes a feature from the **user's point of view**.
The common format is:
"As a [type of user], I want [goal] so that [benefit]."

Example:

- *As a student, I want to register online so that I can access my courses easily.*
- *As an admin, I want to see analytics so that I can track system usage.*

◇ **How to Show User Stories in a Portfolio**

You can create a **section** like this:

<section>

<h2>User Stories</h2>

Here are a few user stories that guided my project designs:

As a student, I want to register online so that I can access my courses anytime.

As a teacher, I want to upload assignments so that students can submit their work digitally.

As an admin, I want to generate reports so that I can monitor overall performance.

</section>

Why Add User Stories?

- Shows **design thinking** (you care about the user).
- Proves you understand **requirements gathering**.
- Helps recruiters/clients see your **real-world approach**.

MVP FEATURES(Minimum Viable Product)

brief bio typically includes:

BRIEF BIO:

1.Your name:

Example: ”John”.

2. Profession/title:

Example: Web Developer.

3. Relevant experience/skills:

Example: 3+ years of experience in front-end development.

4. Achievements/accomplishments:

Example: Successfully developed and deployed multiple web applications.

SKILLS :

Technical Skills: Proficiency in specific tools, technologies, or programming languages

Soft skills: Team work & communication

Experience: Hands-on work experience in a specific field or industry, showcasing skills and achievements.

Soft skills:

Programming languages:

COMMUNICATION:

PROJECT SHOECASE:

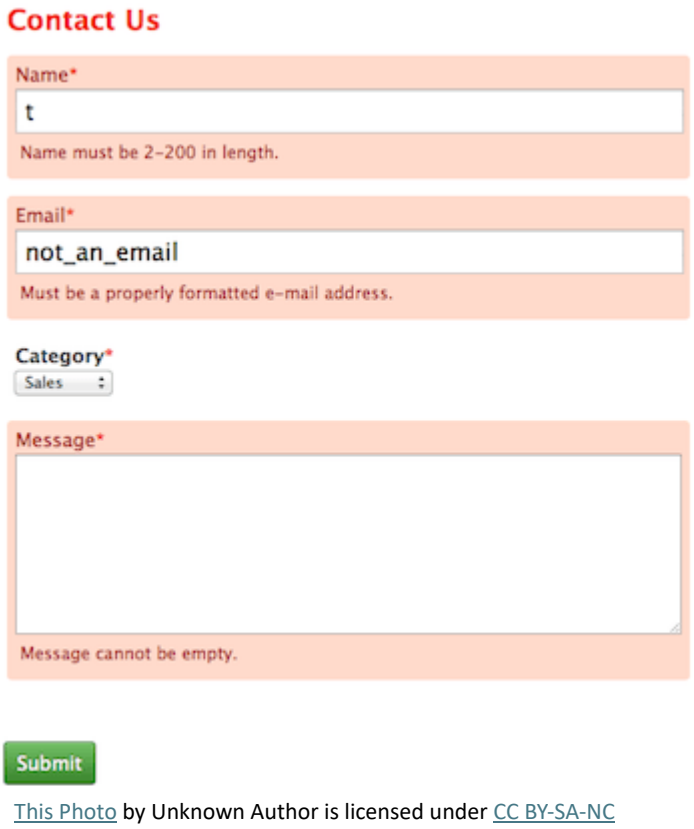
1. Project Images/Screenshots: Visual representation of your projects.

2. Brief Project Descriptions: Short summaries of each project.

3. Technologies Used: List of programming languages, frameworks, and tools used.

CONTACT PAGE:

1. Contact Form: A simple form for visitors to send messages or inquiries.



2. Email Address: Display your email address for direct contact.

3. Social Media Links: Links to your professional social media profiles (e.g.,

).

Portfolio Website Wireframe

A wireframe is a visual representation of a website's layout and structure. Wireframe for a portfolio website:

Header:

Logo: [Your Name/Logo]
Navigation Menu:
About
Projects
Contact

Hero Section:

Background Image/Video:
A visually appealing background that showcases your work or personality
Headline:
A brief introduction to who you are and what you do.
Call-to-Action (CTA) Button:
Encourages visitors to learn more about you.

About Section

Profile Picture: A photo of yourself.
Bio: A brief description of your background, skills, and experience.
. Skills: A list of your technical skills and areas of expertise.

Projects Section

Project Grid:A grid showcasing your projects, each with:
Project Image : A screenshot or image representing the project.
Project Title: The title of the project.
Project Description: A brief summary of the project.

Contact Section

Contact Form: A form that allows visitors to send you a message.
Contact Information: Your email address, phone number, and other relevant contact details.

Footer

Social Media Links: Links to your social media profiles.
Copyright Information: A copyright notice with your name and year.

Portfolio Website API Endpoint List

List of potential API endpoints for a portfolio website:

User Endpoints

1. GET /api/users: Retrieve a list of users (if applicable).
2. GET /api/users/{id}: Retrieve a specific user's information.
3. POST /api/users: Create a new user account.
4. PUT /api/users/{id}: Update a user's information.
5. DELETE /api/users/{id}: Delete a user account.

Project Endpoints

1. GET /api/projects:Retrieve a list of projects.
2. GET /api/projects/{id}: Retrieve a specific project's details.
3. POST /api/projects: Create a new project.
4. PUT /api/projects/{id}: Update a project's details.
5. DELETE /api/projects/{id}: Delete a project.

Contact Endpoints

1. POST /api/contact: Send a message or inquiry.

Authentication Endpoints

1. POST /api/login: Authenticate a user and return a token.
2. LPOST /api/logout: Invalidate a user's token.

Other Endpoints

1. GET /api/skills: Retrieve a list of skills or technologies.
2. GET /api/experience: Retrieve a list of work experience or education.

API Endpoint Considerations

✨Authentication and authorization should be implemented to secure sensitive endpoints.

✨Input validation and sanitization should be performed to prevent security vulnerabilities.

✨ API endpoints should be designed to follow RESTful principles and use meaningful HTTP status codes.

✨API documentation should be provided to facilitate usage and integration.

Acceptance criteria

Acceptance criteria are a set of specific, measurable, and testable conditions that a product or feature must meet to be considered complete and accepted by stakeholders and users. They clarify requirements, align teams on what "done" looks like, and serve as a basis for testing, ensuring that the delivered work fulfills expectations and achieves business goals. Acceptance criteria should be written clearly and concisely, ideally before development begins, and can be structured in various formats, such as "Given-When-Then" scenarios.

Why Acceptance Criteria Are Important

Clear Communication:

They serve as a shared understanding between the development team, product managers, and stakeholders, eliminating ambiguity about requirements.

Quality Assurance:

Acceptance criteria define the scope of what needs to be tested, enabling testers to verify that the product meets the specified functional and non-functional requirements.

Reduced Rework:

By clearly defining what is expected upfront, the criteria help prevent misunderstandings and the need for costly rework later in the development process.

Alignment with User Needs:

They focus on delivering positive results for the end-user, ensuring that the final product satisfies customer expectations and business goals.

Key Characteristics of Effective Acceptance Criteria

Testable:

Each criterion must be verifiable through a clear pass or fail outcome.

Clear and Concise:

They should be easy to understand and avoid jargon or overly technical language.

Specific:

Criteria should be precise and well-defined, leaving no room for interpretation.

Customer-Focused:

They should be written from the user's perspective, focusing on the positive customer outcomes.

Developed Early:

It's best practice to define acceptance criteria before development begins to set expectations and guide the team.

Examples of Acceptance Criteria Formats

Scenario-Based (Given-When-Then):

This format details the conditions under which a feature should perform.

*Given: the user is on the login page, When they enter a valid username and password, Then they should be redirected to their dashboard.

Checklist Format:

A simple list of specific conditions that must be met.

*User successfully logs in with valid credentials.

*User's profile information is displayed correctly on the dashboard.