

Experiment 10: Design and Develop a multi-page, responsive portfolio website / blog showcasing their skills, projects, and contact information.

Learning Objective: Student should be able to synthesize and apply all concepts from the previous nine experiments—including semantic HTML, external CSS, Bootstrap forms and grid, JavaScript validation, and jQuery manipulation—to design, develop, and deploy a complete, multi-page responsive portfolio and blog website for my group.

Tools: Notepad, Google Chrome, Git, GitHub, Command Prompt

Theory:

This capstone experiment represents the **integration and deployment phase** of a complete web development project. It moves beyond individual skills and focuses on **system design**, requiring you to act as a full-stack developer (on the client-side).

You will be combining all the separate modules into a single, cohesive application. Key engineering considerations include:

- **Modularity:** Reusing components, such as a consistent <header> and <footer> across all pages.
- **User Journey:** Planning the logical flow for a visitor (e.g., a recruiter). A typical journey: Homepage → Projects Page → Contact Page.
- **Data Management (Static):** The “blog” will be static, meaning each post is its own HTML content, not pulled from a database. Interactivity will be simulated with jQuery.
- **Full Lifecycle:** This experiment covers the entire development lifecycle:
 1. **Design:** Planning the pages and layout.
 2. **Development:** Writing the code.
 3. **Testing:** Validating forms, checking for broken links, and testing responsiveness.
 4. **Deployment:** Publishing the final site to a live URL.

Result and Discussion:

My Portfolio

Your Name

Master of Engineering Student at University of Mumbai
Specializing in Web Development, Cloud Computing, and Machine Learning.

About Me

Welcome to my personal portfolio. I am a dedicated and passionate Computer Engineering graduate student with a strong foundation in software development and a keen interest in building scalable, efficient, and user-friendly web applications. This website showcases my journey, my projects, and my technical insights.

My Technical Skills

Front-End	Back-End	Other
HTML5 & CSS3 JavaScript (ES6+) Bootstrap & jQuery React.js	Python (Django, Flask) Node.js (Express) Java (Spring) RESTful APIs	Git & GitHub SQL (MySQL, PostgreSQL) Docker & AWS Agile Methodologies

© 2025 Parth Mehta. All rights reserved.

My Projects

Project 1

GadgetGalaxy E-Commerce

A responsive static e-commerce site built with HTML, CSS, Bootstrap, and JavaScript. Features a product catalog and a fully validated checkout form.

[View Code](#)

Project 2

Real-Time Chat App

A full-stack chat application built with Node.js, Express, and Socket.io. Allows multiple users to communicate in real-time in different rooms.

[View Code](#)

Project 3

Sentiment Analysis Bot

A machine learning model built with Python and Scikit-learn, deployed as a REST API using Flask. Analyzes the sentiment of user-provided text.

[View Code](#)

© 2025 Parth Mehta. All rights reserved.



GitHub Dashboard | ParthRMehta/gadget-galaxy: M | Blog - Portfolio | Welcome to GadgetGalaxy | +

parthmehta.github.io/gadget-galaxy/MyPortfolio/blog.html

My Portfolio | Home | Projects | **Blog** | Contact

Blog & Articles

Understanding Client-Side vs. Server-Side Rendering

Posted on 28 October 2023

In web development, two primary approaches exist for rendering content: Client-Side Rendering (CSR) and Server-Side Rendering (SSR). This post explores the fundamental differences...

Client Side Rendering (CSR) is the approach where content is rendered in the browser using JavaScript. When a user requests a page, the server sends a minimal HTML file with a JavaScript bundle. The browser then executes this JavaScript, fetches data from APIs, and builds the HTML on the fly. This is the default for Single Page Applications (SPA) built with frameworks like React, Vue, or Angular.

Pros of CSR:

- **Rich Interactivity:** User interactions feel incredibly fast and responsive after the initial load.
- **Faster Subsequent Navigation:** Clicking links only swaps out components, it doesn't require a full page reload from the server.

Cons of CSR:

- **Slow Initial Load:** The user sees a blank white screen until the large JavaScript bundle is downloaded, parsed, and executed.
- **Poor SEO:** Search engine crawlers may struggle to index content that is created by JavaScript, although this is improving.

Server-Side Rendering (SSR), on the other hand, involves the server generating the full HTML for a page in response to a user's request. When the browser receives the HTML, it's ready to be displayed immediately. The browser then downloads the JavaScript, which "hydrates" the page, attaching event listeners and making it interactive.

Pros of SSR:

- **Fast Initial Page Load:** Users see content almost instantly (a fast "First Contentful Paint").
- **Excellent SEO:** Search engines can easily crawl the fully-formed HTML content.

Cons of SSR:

- **Slower Page-to-Page Navigation:** Every new page requires a full roundtrip to the server to generate new HTML.
- **Higher Server Load:** The server is doing the rendering work for every user request.

Today, many developers use hybrid approaches like **Static Site Generation (SSG)** or **Incremental Static Regeneration (ISR)** to get the best of both worlds.

GitHub Dashboard | ParthRMehta/gadget-galaxy: M | Contact - Portfolio | Welcome to GadgetGalaxy | +

parthmehta.github.io/gadget-galaxy/MyPortfolio/contact.html

My Portfolio | Home | Projects | Blog | Contact

Contact Me

Have a question or want to work together? Fill out the form below.

Your Name

Your Email

Subject

Message

Send Message



The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Contact - Portfolio" and displays a contact form titled "Contact Me". The form includes fields for "Your Name", "Your Email", "Subject", and "Message", along with a "Send Message" button. A green success message at the top of the form area says "Thank you! Your message has been sent." Below the form, there is a copyright notice: "© 2025 Parth Mehta. All rights reserved."

GitHub page displaying MyPortfolio pages.

Learning Outcomes: The student should have the ability to plan, integrate, test, and deploy a complete, multi-page web application, managing the interplay between structure (HTML), style (CSS/Bootstrap), and interactivity (JS/jQuery) in a collaborative group setting.

Course Outcomes: Upon completion of the course students will have successfully synthesized all course concepts to produce a professional, portfolio-worthy, multi-page responsive website, fulfilling the core objective of the course.

Conclusion:

This capstone experiment successfully integrates all the skills from the course into a single, practical, and high-value project. The result is not just a series of exercises but a tangible, live website that can be used professionally to showcase the group's technical abilities. This demonstrates a complete understanding of the front-end development workflow.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				