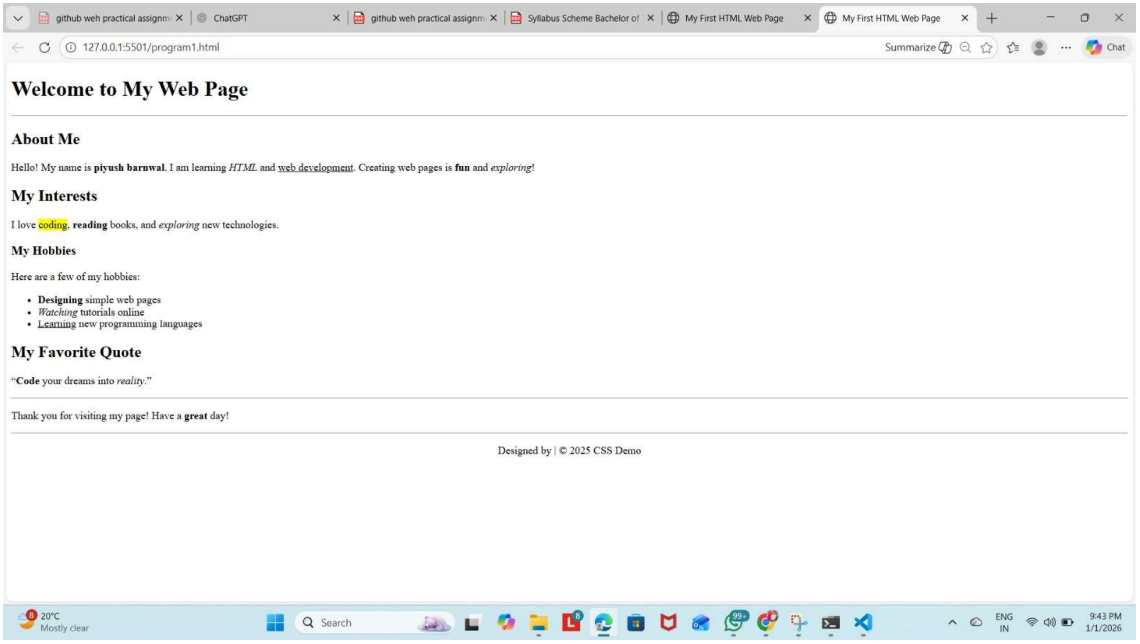
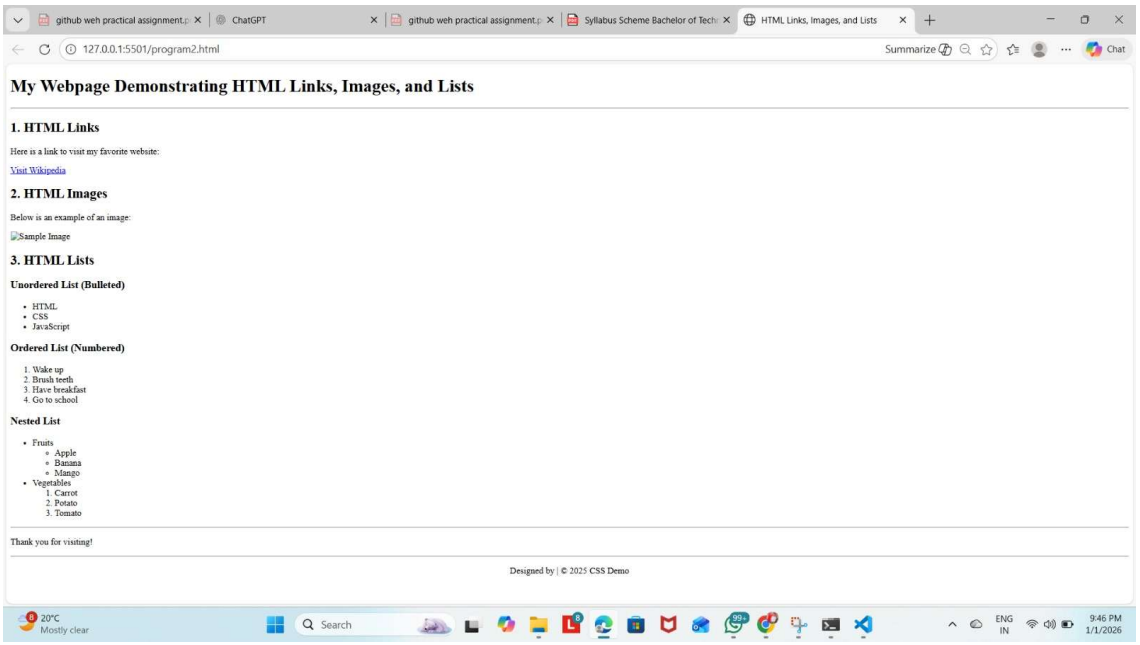
LAB RECORD

UNIT 1

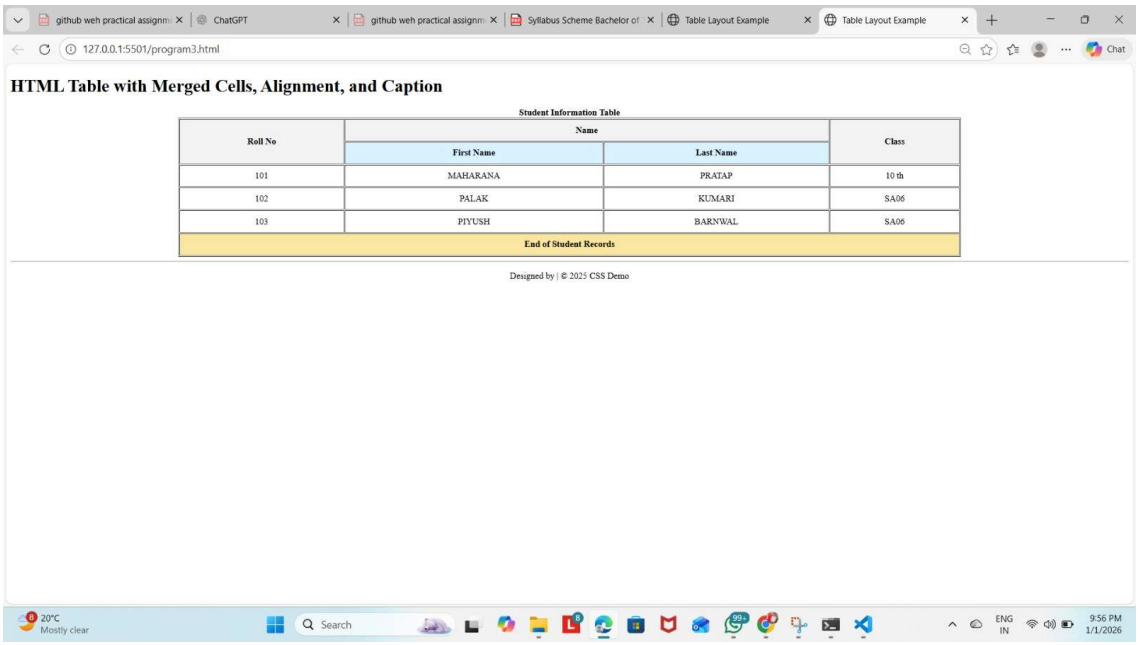
1.Create +C46:C65a basic HTML web page using headings, paragraphs, and text formatting tags.



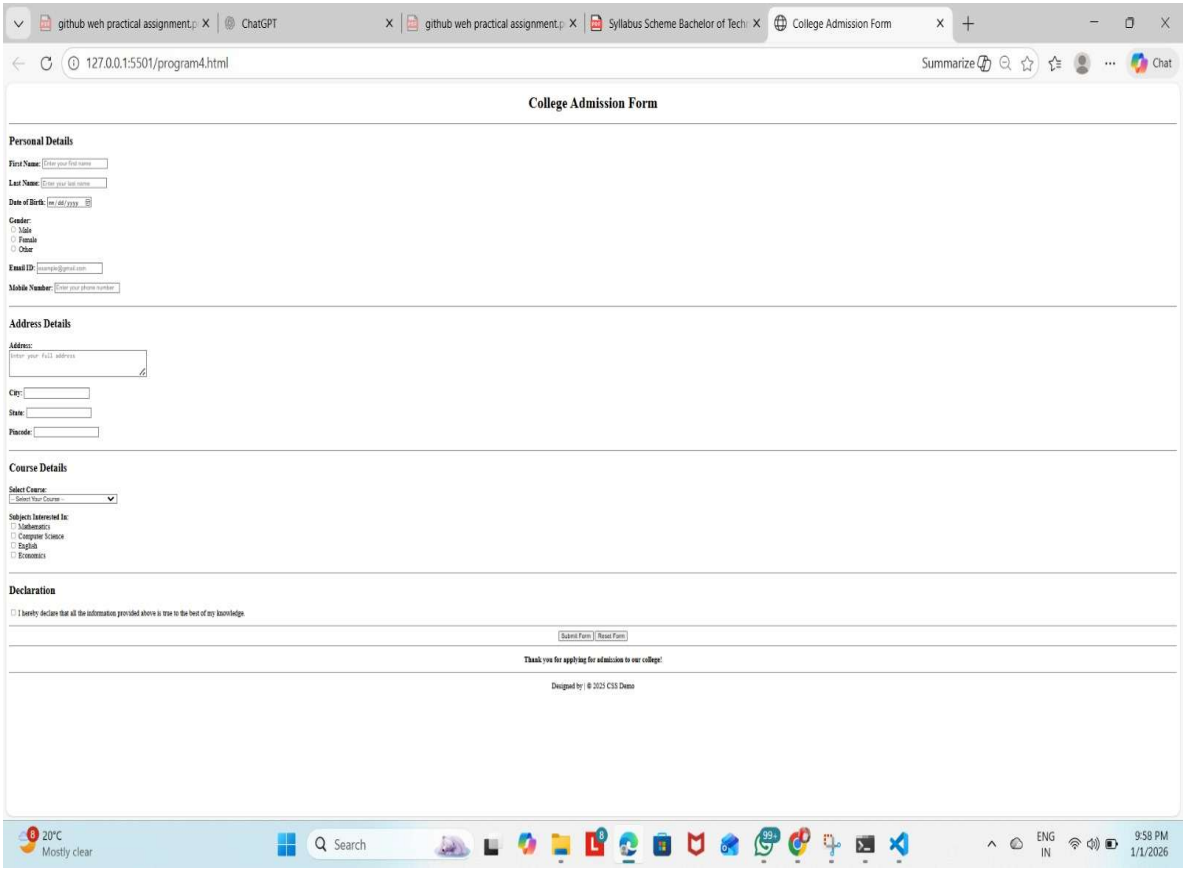
2. Design a webpage demonstrating HTML links, images, and lists (ordered, unordered, nested)



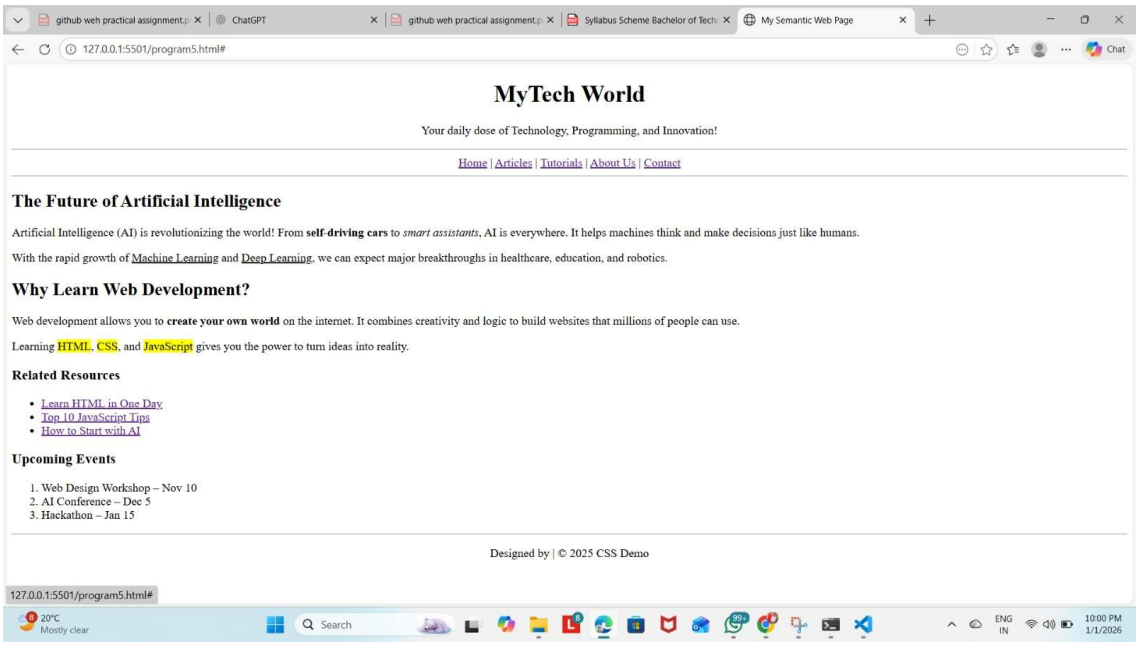
3. Create a table-based layout with merged cells, alignment, and caption using HTML



4. Design a webpage with an HTML form having text input, radio buttons, checkboxes, select menus, and submit/reset buttons



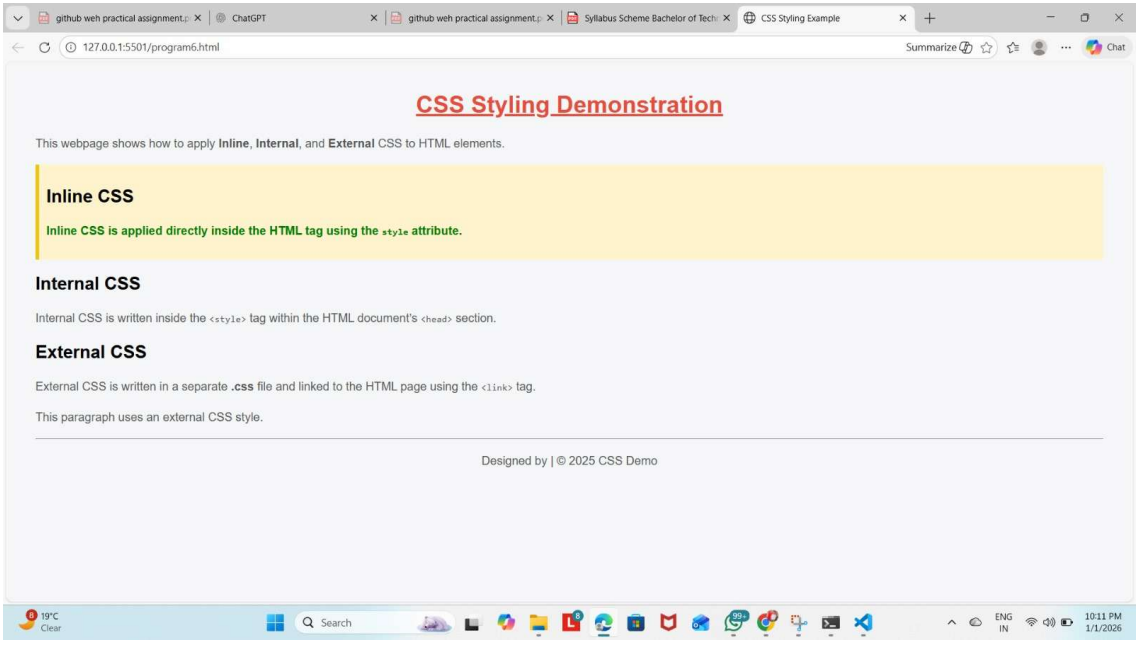
5. Use semantic elements (<article>, <section>, <nav>, <aside>, <footer>) to build a structured web page.



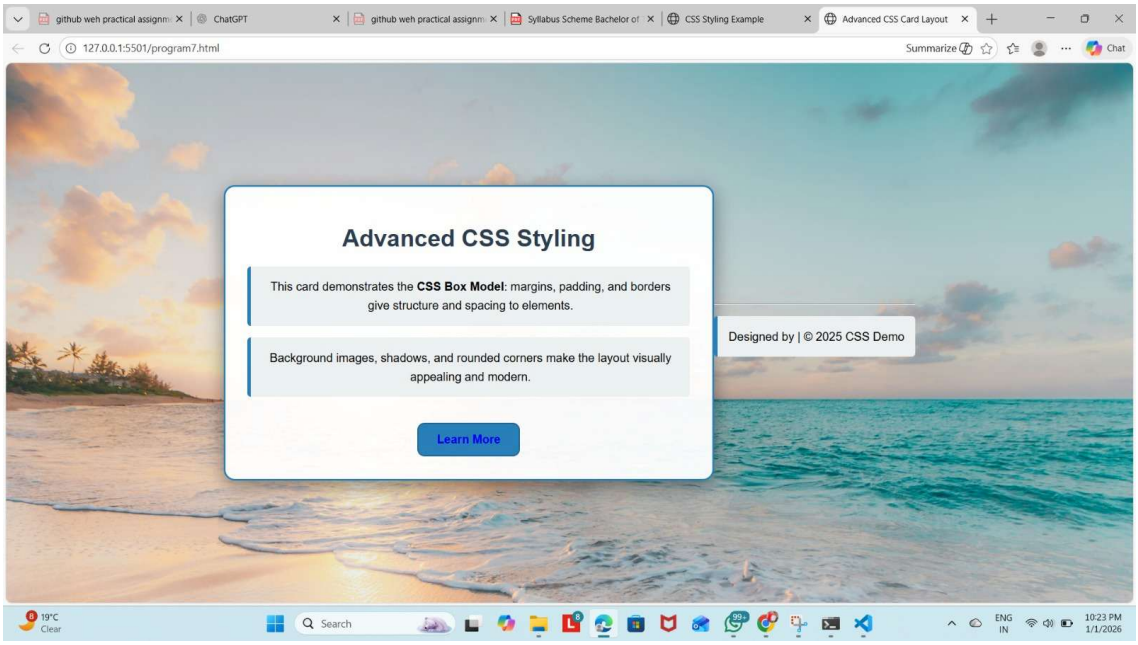
LAB RECORD

UNIT 2

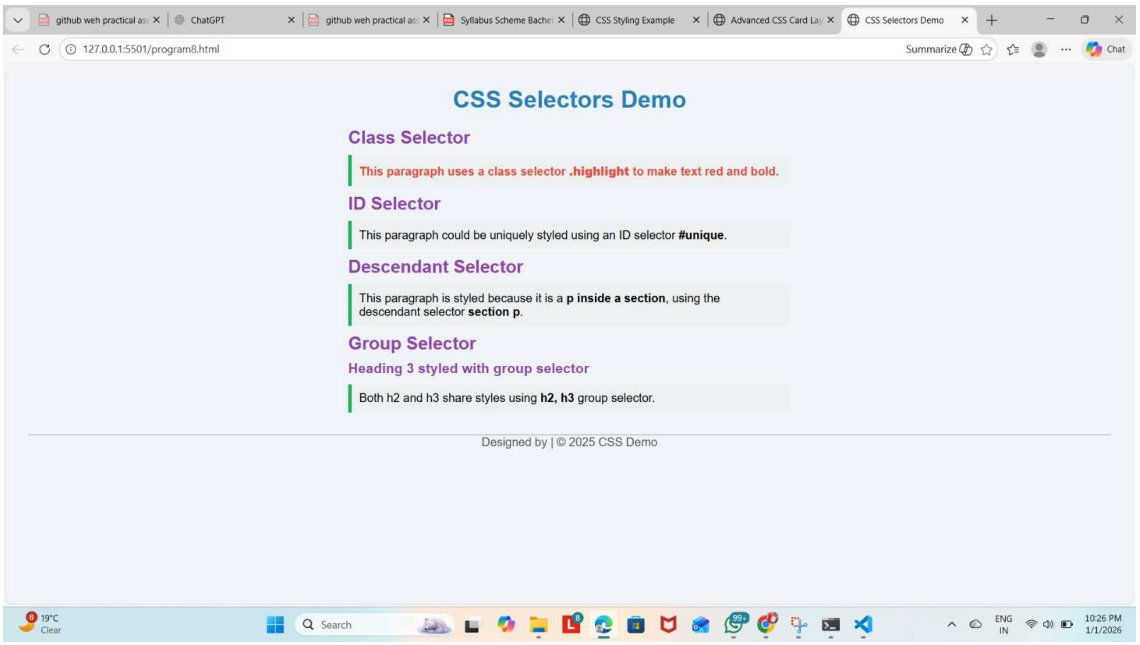
1. Apply Inline, Internal, and External CSS styles to HTML elements.



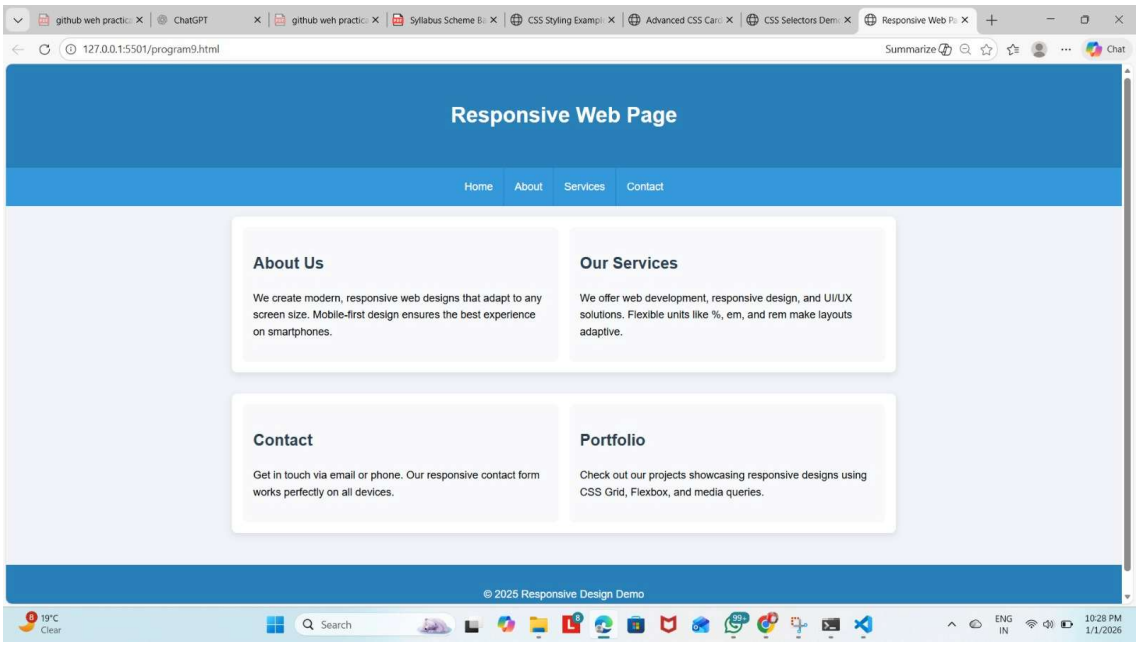
2. Style a web page using advanced CSS: Box Model, background images, borders, margins, and padding



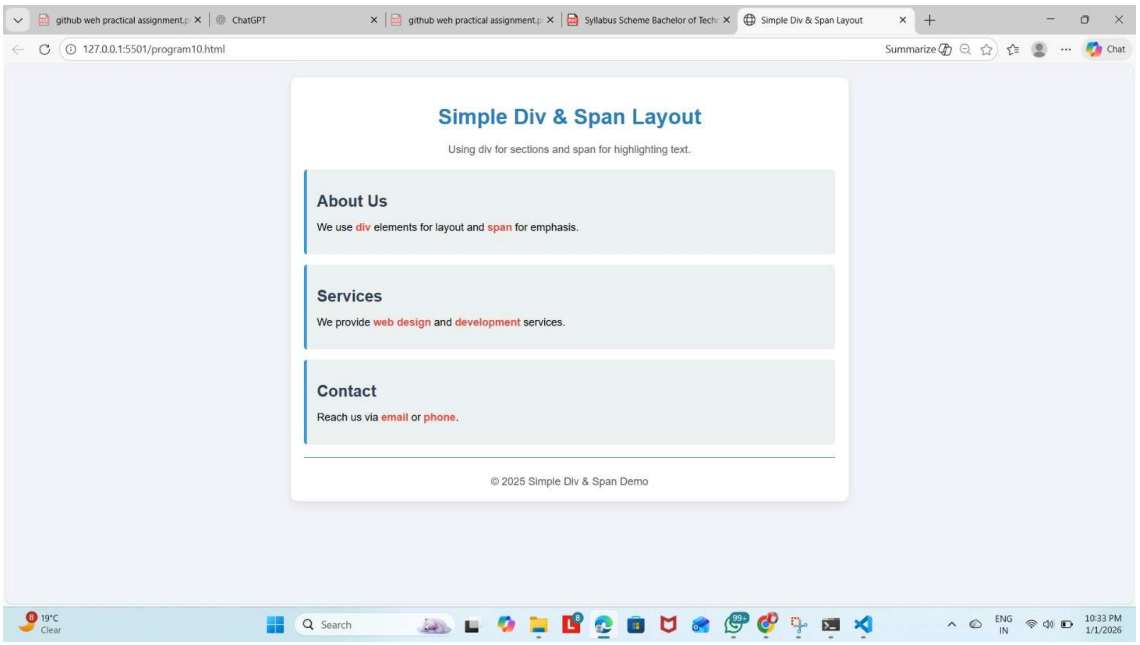
8. Demonstrate the use of different CSS selectors (class, id, descendant, group, universal)



9. Create a responsive web page using media queries and mobile-first design with flexible units



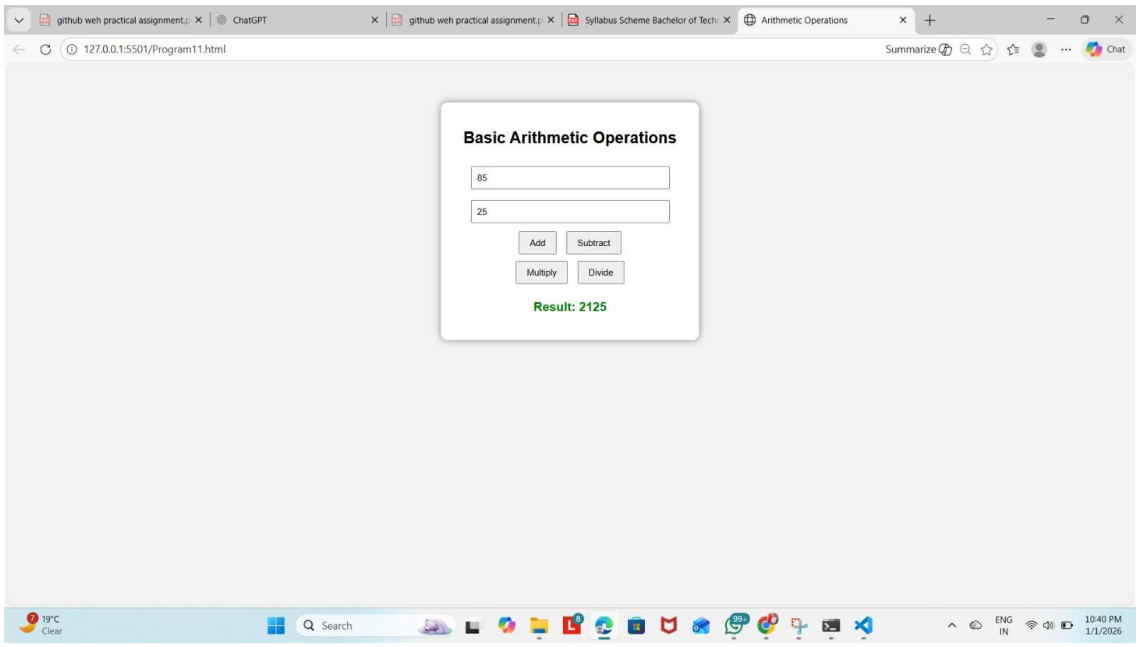
10. Use div and span elements for layout and apply styling with CSS



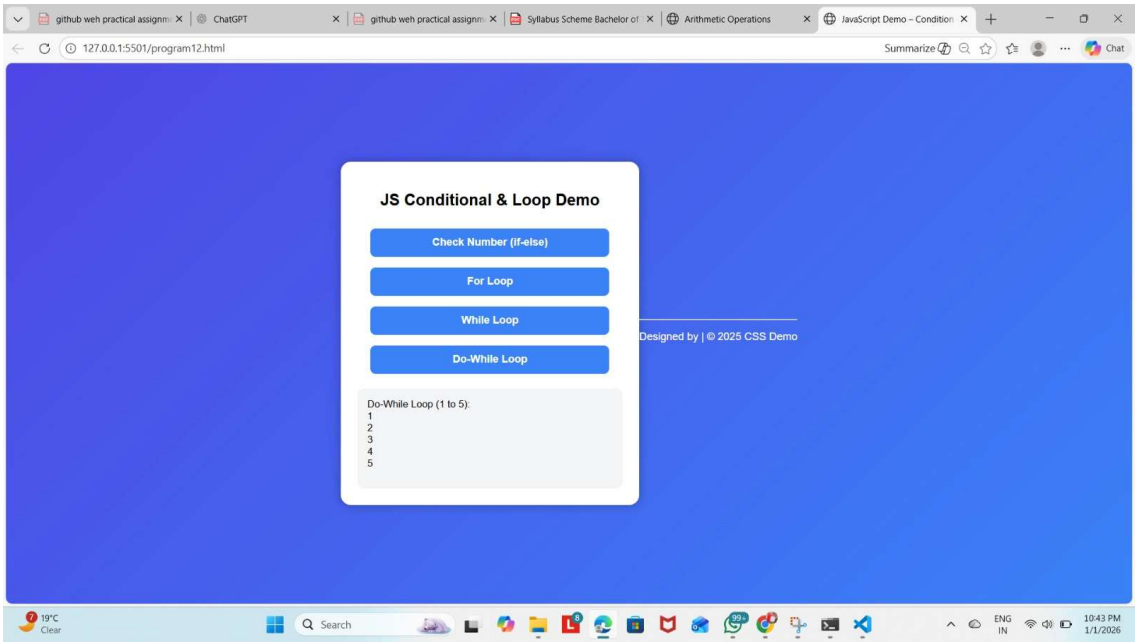
LAB RECORD

UNIT 3

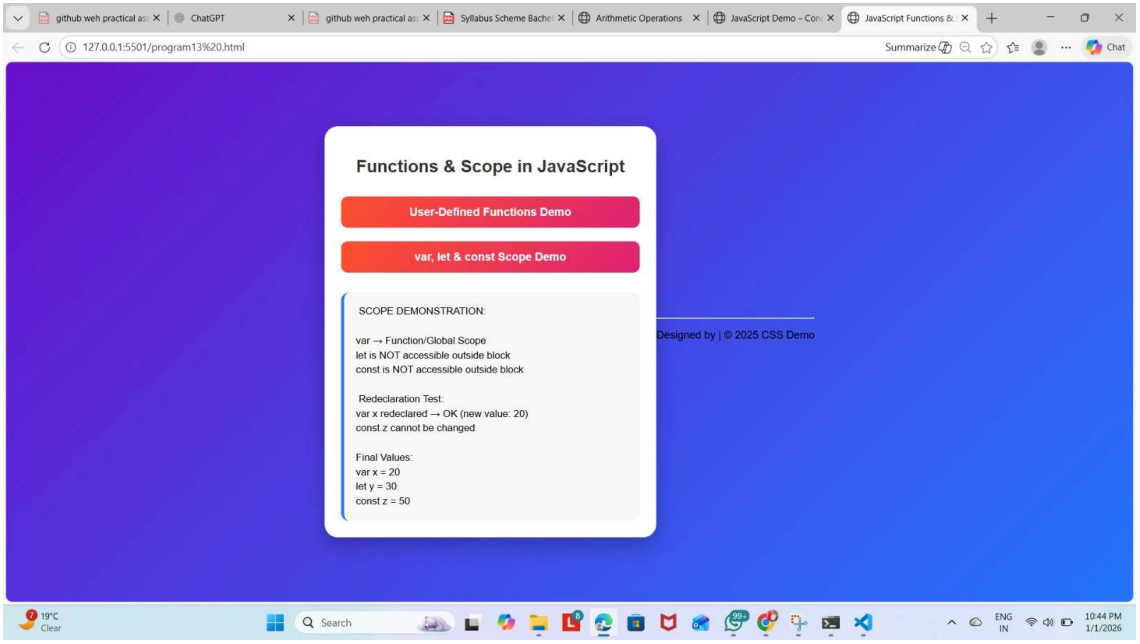
1. Create a JavaScript-enabled web page that performs basic arithmetic operations using input from users



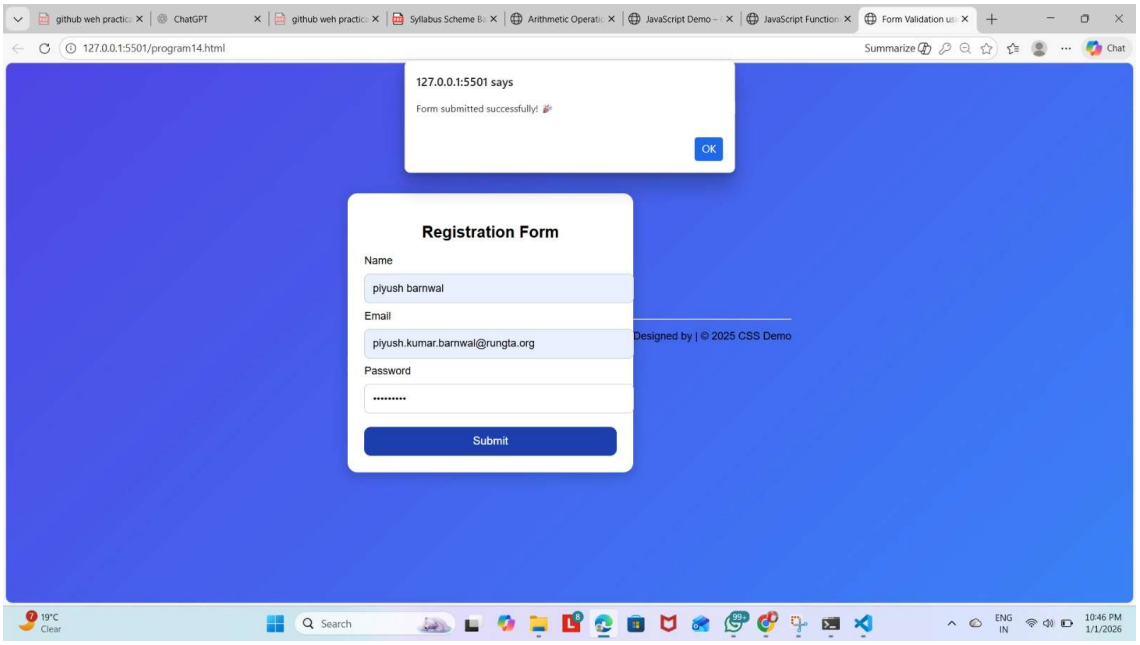
2. **Write JavaScript to demonstrate conditional statements and looping (for, while, do…while)**



3. Create and invoke user-defined functions in JavaScript; use var, let, and const for scope demonstration



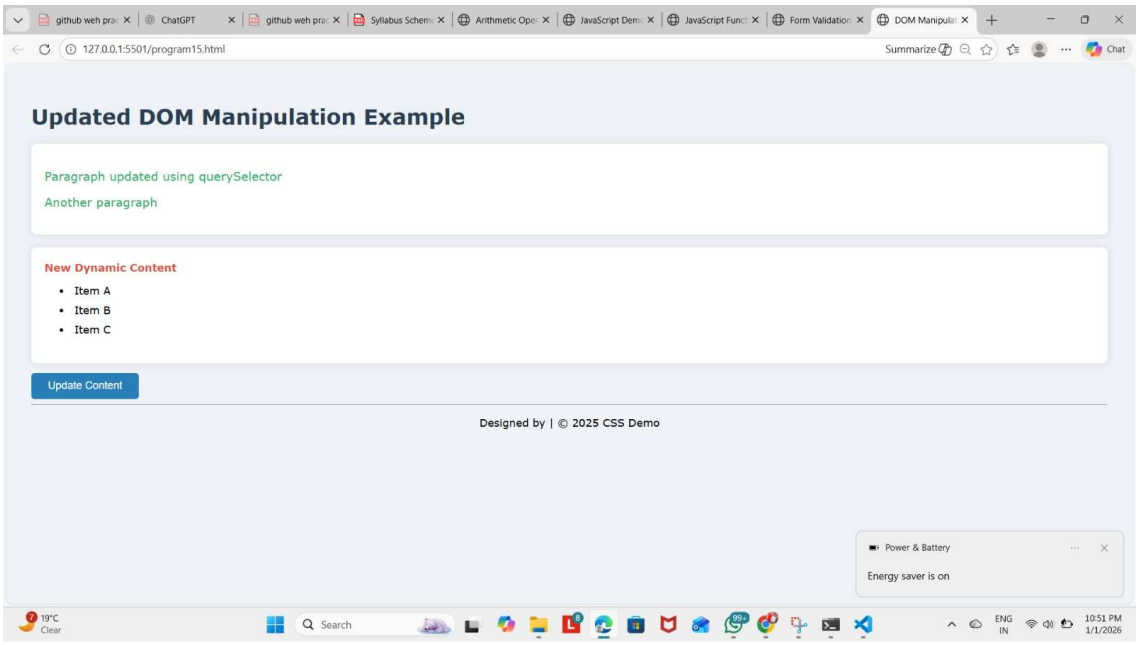
4. Handle HTML form validation using JavaScript events like on submit, on input, and use alert() for feedback.



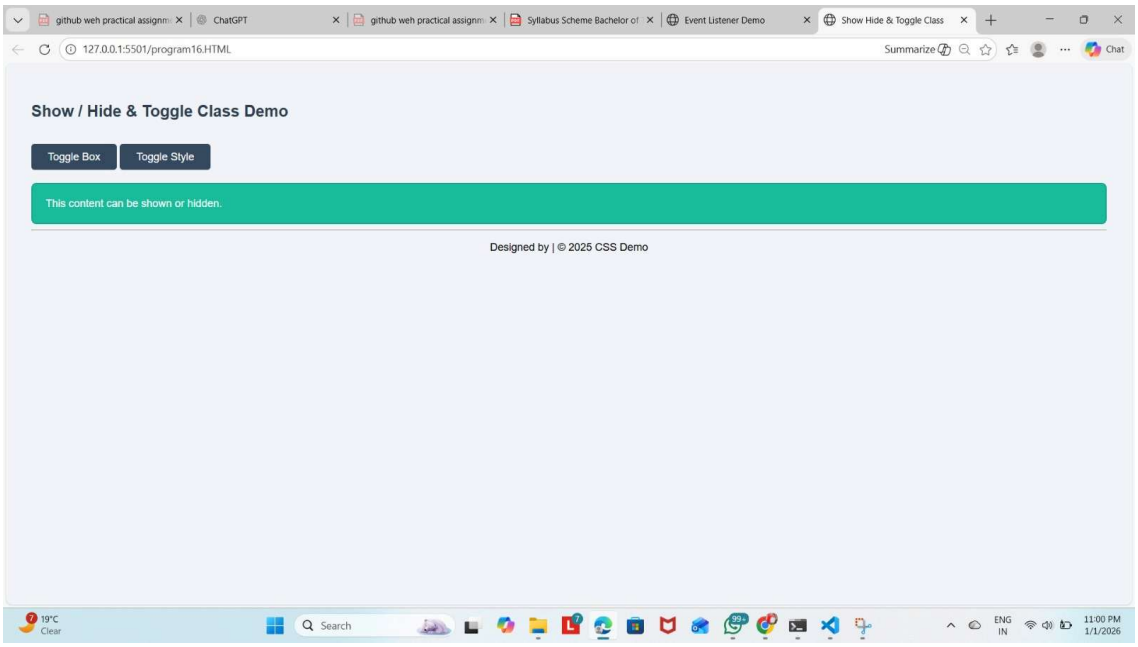
LAB RECORD

UNIT 4

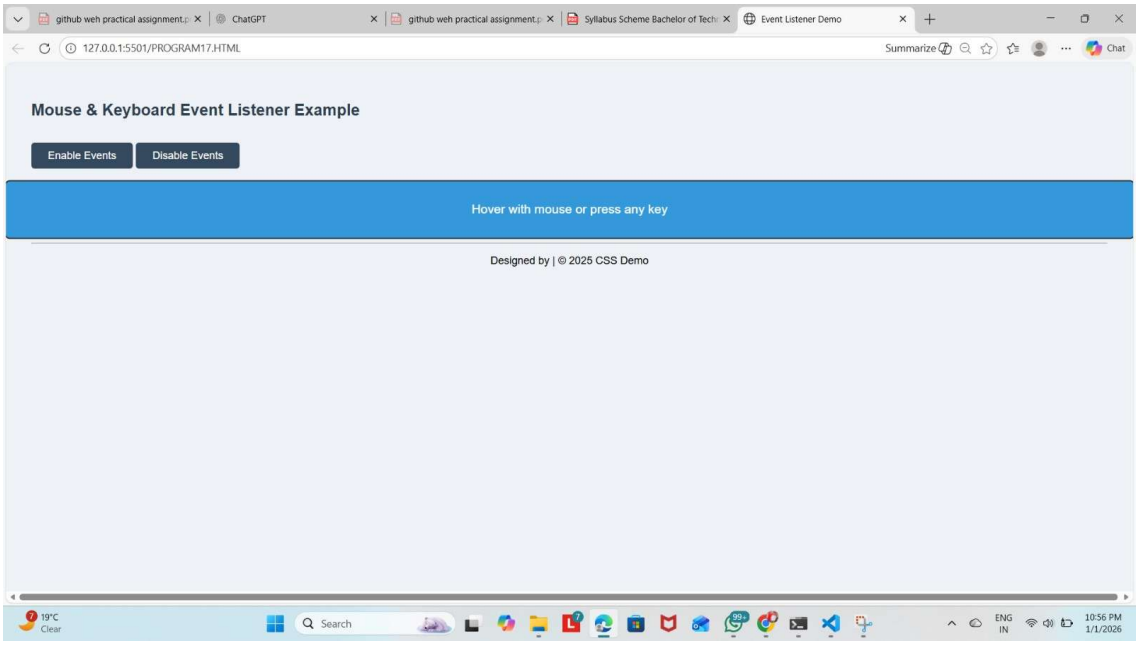
1. Demonstrate DOM manipulation using getElementById(), querySelector(), and innerHTML.



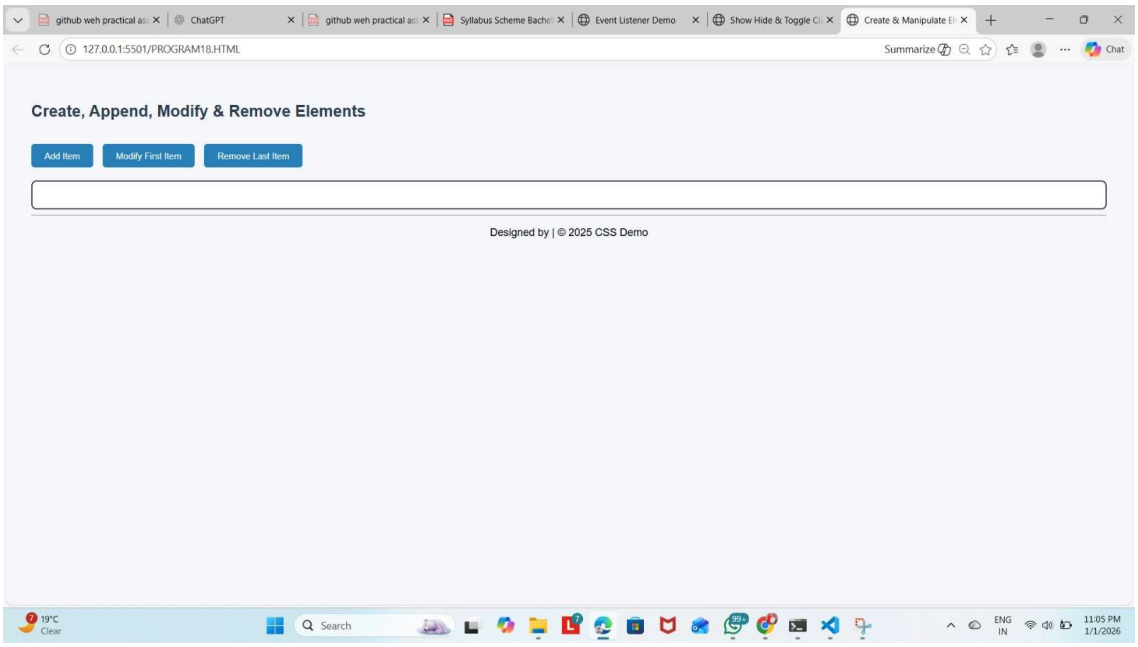
2.Show or hide HTML elements using JavaScript and toggle CSS classes dynamically



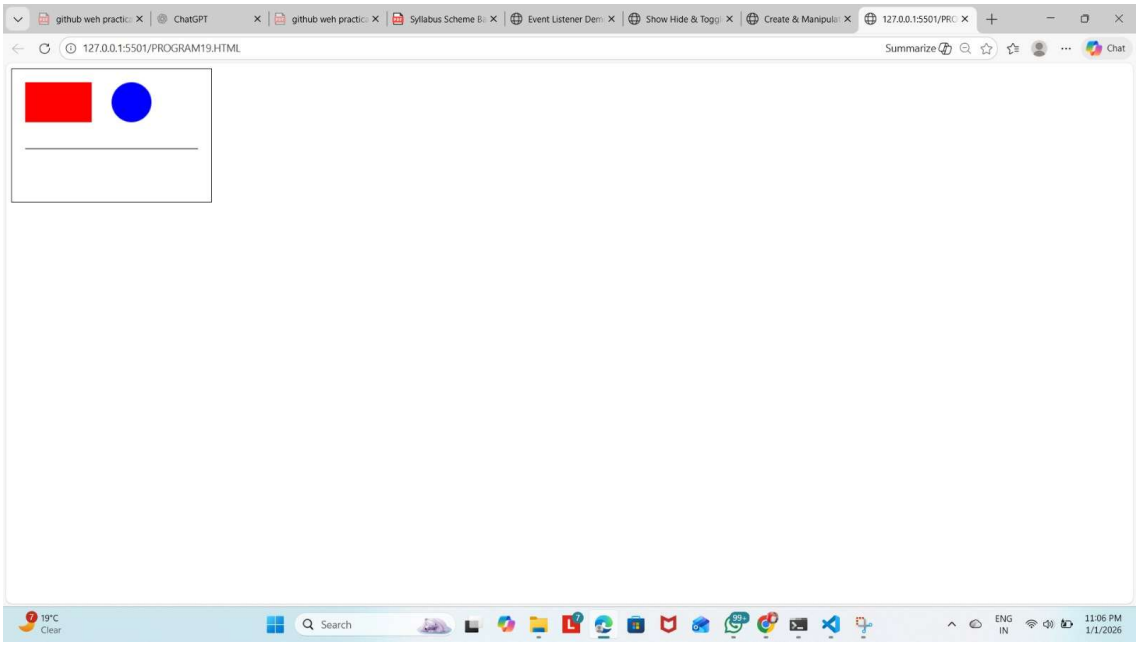
3. Add interactivity using event listeners (addEventListener, removeEventListener) for mouse/keyboard events.



4.Create and manipulate elements using JavaScript: append, remove, or modify child nodes.



5. Create a simple drawing using the <canvas> element: draw shapes and fill colors.



LAB RECORD

UNIT 5

20. Initialize a Git repository, commit changes, and push to GitHub; Deploy a personal portfolio site on GitHub Pages or Netlify.

Initializing a Git Repository, Pushing to GitHub, and Deploying a Personal Portfolio Website

Git and GitHub are widely used tools for version control and code hosting, while platforms like GitHub Pages and Netlify are used to deploy static websites. This process is commonly used to publish personal portfolio websites online.

Part A: Initialize a Git Repository and Commit Changes

Step 1: Create a Project Folder

Create a folder for your personal portfolio website containing files such as:

index.html

style.css

script.js (optional)

Step 2: Open Terminal / Command Prompt

Open the terminal inside the project folder.

Step 3: Initialize Git Repository

bash

Copy code

git init

👉 This command initializes an empty Git repository.

Step 4: Check Repository Status

bash

Copy code

git status

👉 Shows untracked or modified files.

Step 5: Add Files to Staging Area

bash

Copy code

git add .

👉 Adds all project files to the staging area.

Step 6: Commit Changes

bash

Copy code

```
git commit -m "Initial commit of portfolio website"
```

👉 Saves changes with a meaningful message.

Part B: Push Project to GitHub

Step 7: Create a Repository on GitHub

Log in to GitHub

Click New Repository

Enter repository name (e.g., portfolio-website)

Click Create Repository

Step 8: Connect Local Repository to GitHub

bash

Copy code

```
git remote add origin
https://github.com/username/portfolio-website.git
```

Step 9: Push Code to GitHub

bash

Copy code

git branch -M main

git push -u origin main

👉 Project files are now uploaded to GitHub.

Part C: Deploy Portfolio Website Using GitHub Pages

Step 10: Open Repository Settings

Go to GitHub repository

Click Settings → Pages

Step 11: Enable GitHub Pages

Select Branch: main

Select Folder: /root

Click Save

Step 12: Access Live Website

GitHub generates a URL like:

👉 https://username.github.io/portfolio-website/

OR

Part D: Deploy Portfolio Website Using Netlify

Step 10: Login to Netlify

Go to www.netlify.com and log in.

Step 11: Deploy Using Drag & Drop

Click Add new site → Deploy manually

Drag and drop your project folder

Wait for deployment

Step 12: Live Website URL

Netlify provides a live URL like:

👉 https://your-site-name.netlify.app

Advantages of Using GitHub Pages / Netlify

Free hosting for static websites

Easy and fast deployment

Automatic HTTPS

Ideal for personal portfolios

Conclusion

By initializing a Git repository, committing changes, and pushing code to GitHub, developers can manage their projects efficiently. Using GitHub Pages or Netlify, a personal portfolio website can be deployed easily and accessed globally through a live URL.