Name : Shivam.R.Prajapati **Div: D15C** **Roll No: 41**

Aim: To study and implement deployment of Ecommerce PWA to GitHub Pages.

Theory:

Progressive Web Applications (PWAs) represent a modern approach to building web applications that combine the accessibility of websites with the rich user experience of native apps. Hosting these applications effectively is crucial to ensure performance, availability, and security. GitHub Pages offers a free and straightforward platform to host static web applications, including PWAs, making it a popular choice for developers worldwide.

What are GitHub Pages?

GitHub Pages is a static site hosting service integrated with GitHub repositories. It allows developers to publish web content directly from a repository without requiring any server-side infrastructure.

●Features of GitHub Pages:

○ Free and Public Hosting: GitHub Pages provides free hosting for open-source projects.

○ Automatic HTTPS: All sites hosted via GitHub Pages are served over HTTPS, ensuring security.

○ Branch or Folder Deployment: Content can be deployed from specific branches (like main or gh-pages) or designated folders (docs/).

○ Custom Domains: Supports connecting custom domain names.

○ Integrated with Git Version Control: Easy deployment and versioning through Git.

Understanding Progressive Web Applications (PWAs)

A Progressive Web Application (PWA) is a web app enhanced with modern web capabilities that enable it to provide a user experience similar to native applications.

●Characteristics of PWAs:

○ **Responsive:** Works well on any device and screen size.

○ **Offline Capabilities:** Uses service workers to cache resources and work without an internet connection.

○ **Installable:** Users can install the app to their home screen, bypassing app stores.

○ **Secure:** Served over HTTPS, which is mandatory for many PWA features.

○ **Discoverable:** Indexable by search engines.

○ **Re-engageable:** Supports push notifications.

●**Core Components:**

● Manifest file (manifest.json): Describes the app's appearance, icons, and launch parameters.

● Service Worker: Background script that manages caching and offline behavior.

● Static assets: HTML, CSS, JavaScript files forming the app's UI and logic.

Why Host PWAs on GitHub Pages?

GitHub Pages offers a compelling hosting option for PWAs because:

● Static Hosting Compatibility: PWAs are typically built into static files after a build process. GitHub Pages serves these static files efficiently.

● HTTPS Support: Service workers require HTTPS for security reasons. GitHub Pages automatically provides HTTPS, enabling full PWA functionality.

● Cost-Effective and Easy Deployment: GitHub Pages provides free hosting with minimal configuration, ideal for developers, learners, and small projects.

● Seamless Integration with Git: Hosting from a GitHub repository enables easy version control and continuous deployment workflows.

General Steps to Deploy a PWA to GitHub Pages

Step 1: Prepare the Production Build

Use your framework's build tool to generate a production-ready version of the app. For

example:

```
npm run build
```

This generates a folder (build/, dist/, or equivalent) containing optimized static files.

Step 2: Configure Paths

Ensure paths in your PWA (such as those in manifest.json and asset references) are relative or correctly set according to the GitHub Pages URL, which usually includes your GitHub username and repository name.

Step 3: Setup GitHub Repository

Create or use an existing GitHub repository for your project.

Step 4: Choose Deployment Branch

GitHub Pages can deploy from the repository's main branch, a gh-pages branch, or a docs folder. For PWAs, it is common to deploy from the gh-pages branch.

Step 5: Deploy Static Files

There are multiple ways to deploy:

- Manual deployment: Commit and push your build files to the deployment branch.
- Automated deployment: Use tools like gh-pages npm package to deploy automatically:

```
npm install gh-pages --save-dev
```

Add deployment scripts in your package.json:

```
"scripts": {

"predeploy": "npm run build", "deploy": "gh-pages -d build"

}
```

Then run:

npm run deploy

Step 6: Access Your PWA

Visit your PWA at: https://<username>.github.io/<repository-name>/

Important Considerations for PWAs on GitHub Pages

- **Routing Issues:**

Since GitHub Pages is static, client-side routing (e.g., React Router) can cause 404 errors when users refresh or access deep-linked URLs. Solutions include:

- **Using hash-based routing.**
- **Adding a 404.html fallback redirect to the index page.**
- **Service Worker and Cache Management**

Careful configuration of service worker caching is important to ensure users get the latest updates without stale content.

- **Asset Paths and Manifest Configuration**

Paths to assets like icons in manifest.json must reflect the GitHub Pages URL path structure.

- Limitations

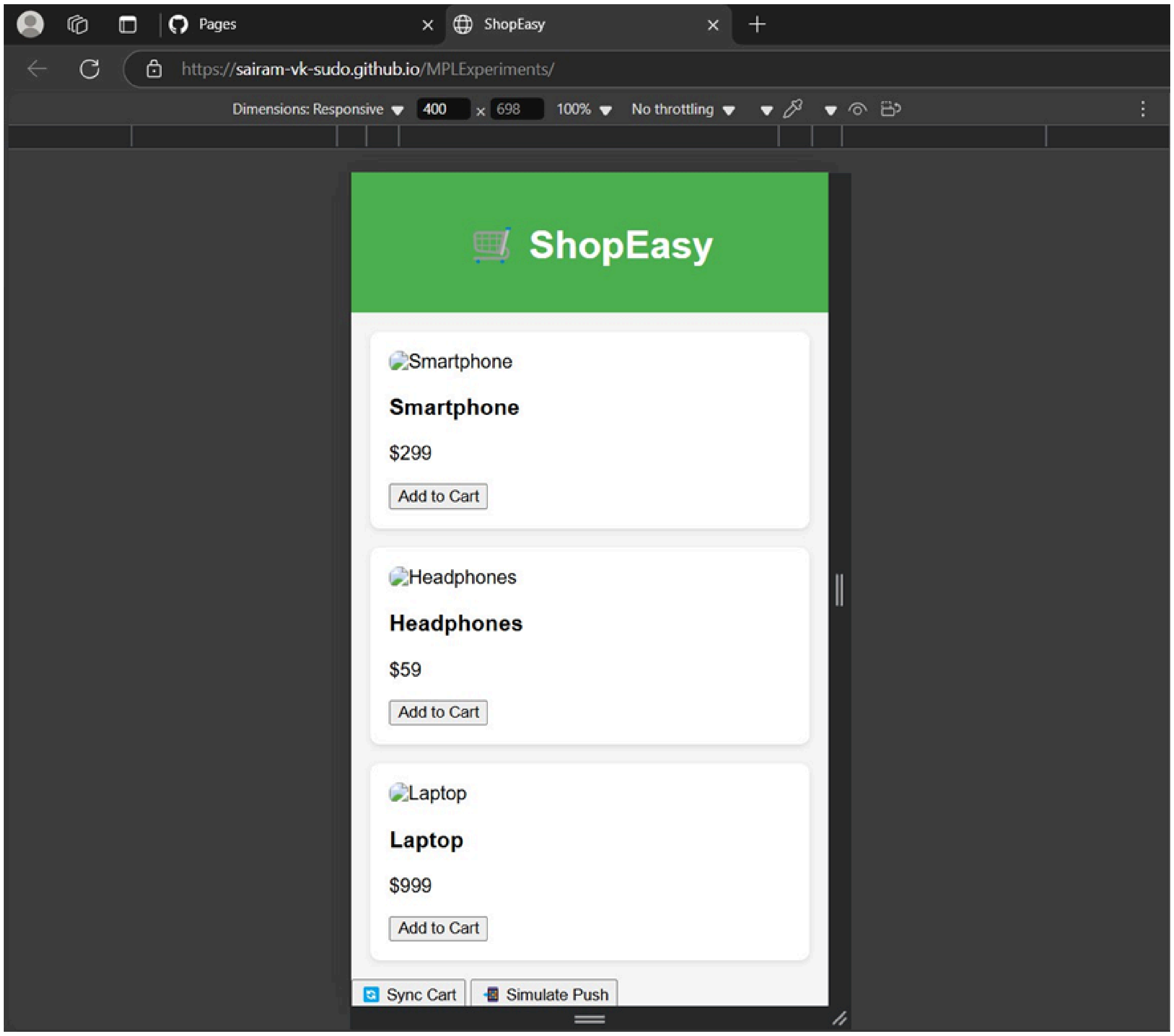
○ No server-side code execution (e.g., no Node.js backend).

○ Not suitable for dynamic backend-powered apps unless combined with APIs.

Code:https://github.com/Kingmaker-2/PWA-10.git

Output:

Name : Shivam.R.Prajapati **Div: D15C** **Roll No: 41**



Hosted App

Conclusion: Hosting Progressive Web Applications on GitHub Pages offers a practical, cost-effective, and secure solution for deploying modern web apps. GitHub Pages provides free static hosting with automatic HTTPS, which is essential for enabling core PWA features such as service workers and offline functionality. Its seamless integration with Git simplifies deployment and version control, making it accessible for developers of all skill levels. While there are some limitations due to its static nature—such as the inability to run server-side code and potential routing challenges—these can often be managed with proper configuration. Overall, GitHub Pages is an excellent platform for delivering fast, reliable, and secure PWAs to a global audience with minimal setup and maintenance effort.