

Experiment 10

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

Theory:

GitHub Pages: GitHub Pages allows developers to freely host and easily publish public web pages directly from their GitHub repositories. It offers seamless deployment—just edit, push your changes, and your website goes live. Some of its key features include blogging support with Jekyll, custom URLs, and an automatic page generator. Compared to Firebase, GitHub Pages is favored for being completely free, easy to set up, and integrated directly with GitHub, making it an excellent choice for developers already using the platform. It is used by well-known companies like Lyft, CircleCI, and HubSpot, and is listed in 775 company tech stacks and 4,401 developer stacks. The platform offers several advantages such as a familiar interface, out-of-the-box Jekyll support, and the ability to use custom domains by adding a CNAME file and updating DNS records. However, it does have some limitations—your site's code will be public unless you use a paid private repository, HTTPS support for custom domains is not yet available, and plugin support in Jekyll can be somewhat limited.

Github: <https://github.com/brijeshforcollege/PWA-App>

PWA Architectural Requirements

For successful deployment, the PWA must conform to GitHub Pages' constraints through:

- Client-Side Rendering: Implementation of CSR patterns using frameworks (React, Vue) or vanilla JS
- External API Consumption: Reliance on CORS-enabled endpoints for dynamic data fetching
- Asset Path Resolution: Absolute URL management for resources in repository subdirectories

Service Worker Implementation

The service worker lifecycle must account for: Scope Determination: Worker registration path relative to repository root Cache Strategy Selection: Network-first for API calls, cache-first for static assets

Versioned Asset Management:

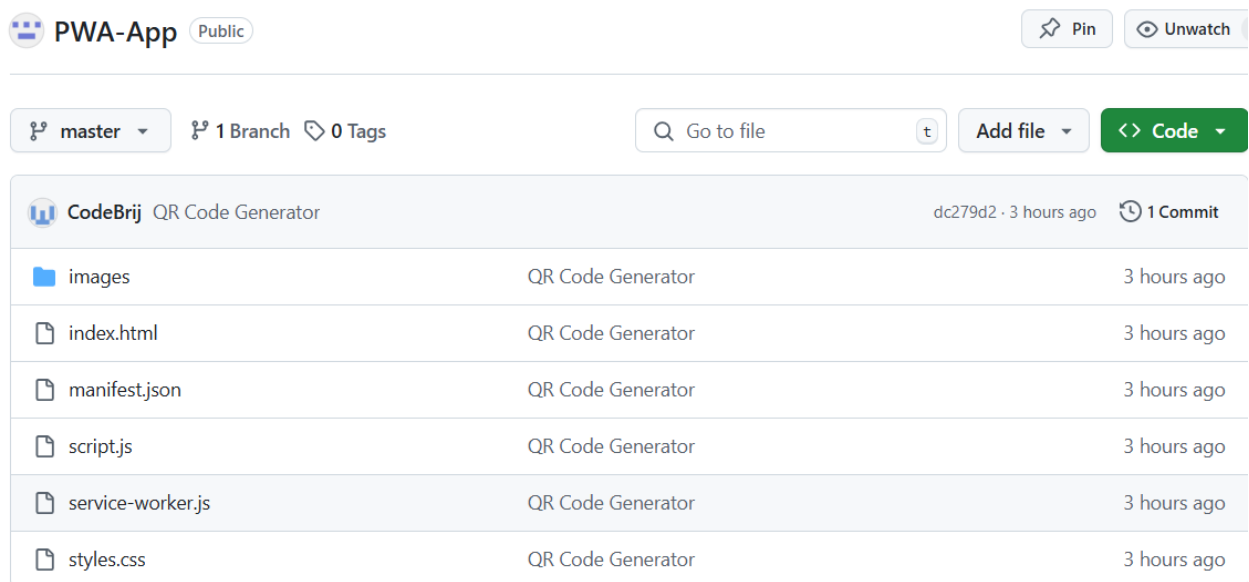
Cache busting through content hashing Deployment Process Model

1. Repository Configuration
 - Branch selection (main/gh-pages)
 - Build artifact directory specification
 - Custom domain mapping (optional)
2. Build Optimization
 - Static asset compilation
 - Route manifest generation
 - Resource preloading
3. Hosting Activation.
 - GitHub Pages service enablement Automated CI/CD pipeline integration
4. Network Behavior

- Cache-Control Headers: GitHub's default caching policies
 - CDN Propagation: Global asset distribution latency
 - HTTP/2 Support: Multiplexed connection advantages
5. Verification Methodology Lighthouse Auditing:
- PWA compliance scoring
 - Offline Simulation: Network throttling tests
 - Cross-Browser Validation: Feature support matrix
6. Limitations and Boundary Conditions
- No Server-Side Logic: Restricted to client-executed
 - JavaScript Subdirectory Challenges: Path resolution in nested repositories
 - Build File Size Limits: 1GB repository soft cap

Steps:

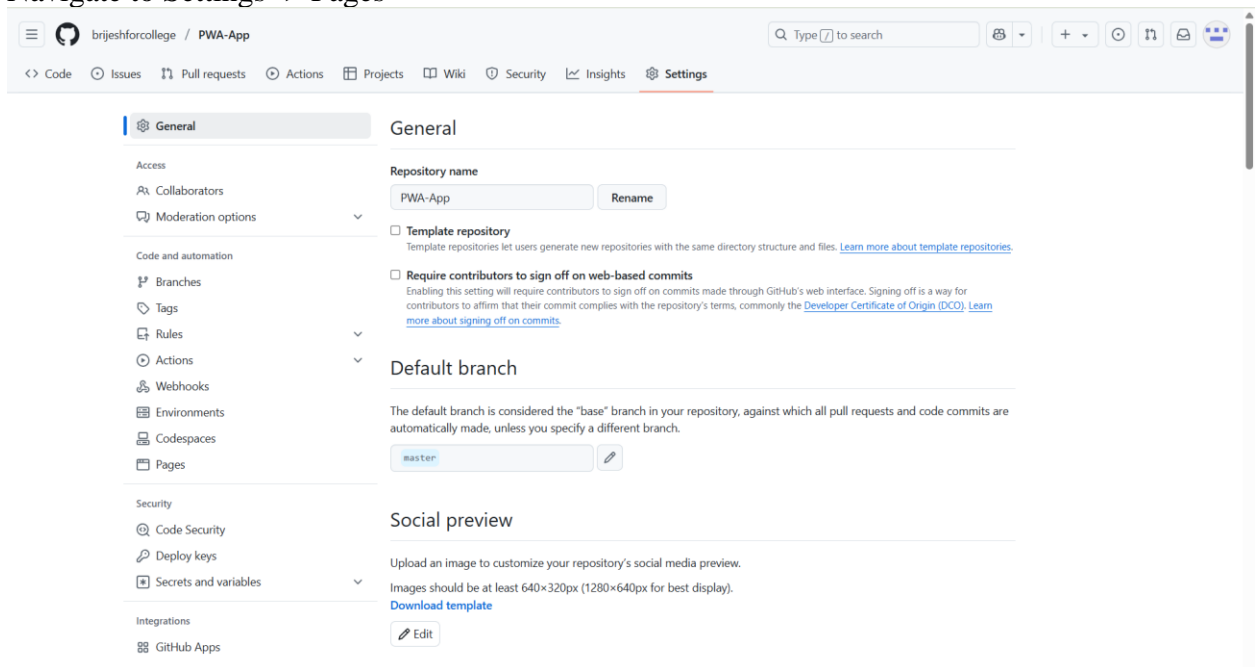
- 1) Create a new github repository for the project
- 2) Commit all the files to the repository



The screenshot shows the GitHub repository page for 'PWA-App', which is public. At the top, there are buttons for 'Pin' and 'Unwatch'. Below this, the repository is shown to be on the 'master' branch with 1 branch and 0 tags. A search bar and 'Add file' button are present. The file list shows the following files, all committed 3 hours ago by 'QR Code Generator':

File Name	Committed By	Time
images	QR Code Generator	3 hours ago
index.html	QR Code Generator	3 hours ago
manifest.json	QR Code Generator	3 hours ago
script.js	QR Code Generator	3 hours ago
service-worker.js	QR Code Generator	3 hours ago
styles.css	QR Code Generator	3 hours ago

- 3) Navigate to Settings -> Pages



The screenshot shows the 'Settings' page for the 'PWA-App' repository. The 'General' tab is selected. The 'Repository name' is 'PWA-App'. The 'Default branch' is 'master'. The 'Social preview' section shows an option to upload an image to customize the repository's social media preview. The left sidebar contains a navigation menu with the following items:

- General
- Access
- Collaborators
- Moderation options
- Code and automation
- Branches
- Tags
- Rules
- Actions
- Webhooks
- Environments
- Codespaces
- Pages
- Security
- Code Security
- Deploy keys
- Secrets and variables
- Integrations
- GitHub Apps

- 4) Select source as “Deploy from a branch” and branch as the branch where your complete pwa app is.

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Build and deployment

Source

Deploy from a branch ▾

Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

None ▾

Save

Visibility

GitHub Enterprise

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise. You can try GitHub Enterprise risk-free for 30 days. [Learn more about the visibility of your GitHub Pages site.](#)

Start free for 30 days

- 5) After saving, wait for some time, and then reload the page. You will see the link for the hosted website

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://brijeshforcollege.github.io/PWA-App/>

Last [deployed](#) by  [brijeshforcollege](#) now

 Visit site

...

Conclusion:

This experiment successfully demonstrated the deployment of a Progressive Web App (PWA) to GitHub Pages, establishing it as an effective zero-cost hosting solution for production-ready PWAs. Through careful configuration of build settings, proper asset path management, and service worker optimization, we verified that GitHub Pages can fully support core PWA features including offline functionality, HTTPS security, and installability. The deployment process highlighted the importance of static site architecture considerations while proving that GitHub's infrastructure adequately meets the technical requirements for hosting performant, secure progressive web applications with reliable global CDN distribution.