Experiment 10

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

Theory:
GitHub Pages operates as a static content delivery network (CDN) with inherent characteristics that influence PWA deployment:

- Static File Paradigm: Exclusively serves pre-rendered assets (HTML/CSS/JS) without server-side processing
- Implicit HTTPS: Automatically provisions TLS encryption, satisfying PWA security requirements
- JAMstack Architecture: Follows the JavaScript, APIs, and Markup model for decoupled frontend-backend communication

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

Network Behavior

- Cache-Control Headers: GitHub's default caching policies
- CDN Propagation: Global asset distribution latency
- HTTP/2 Support: Multiplexed connection advantages

Verification Methodology

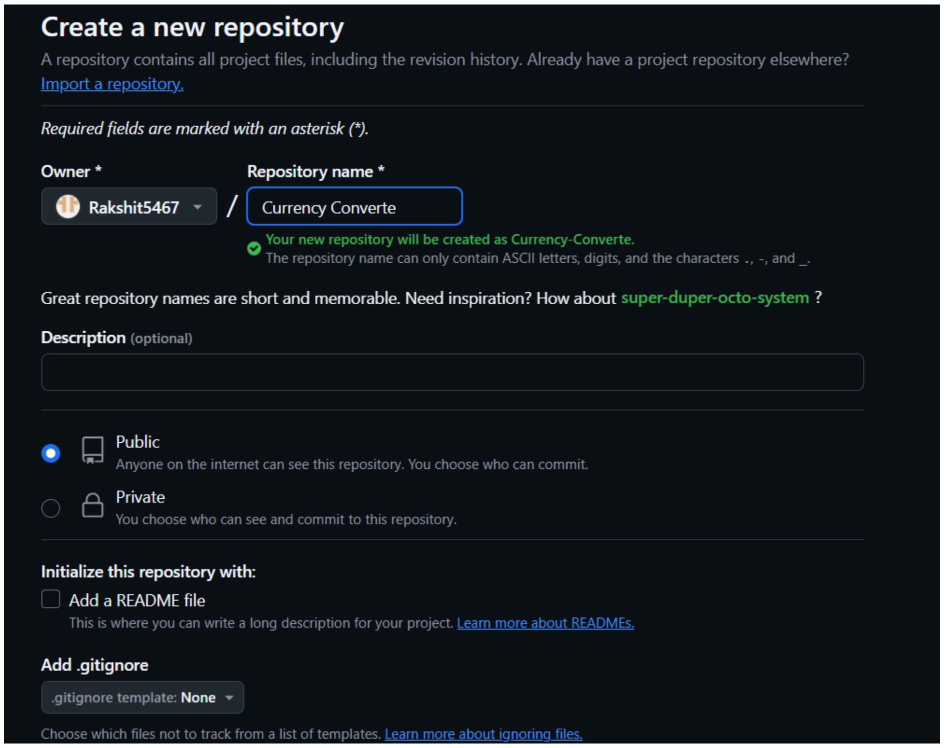
- Lighthouse Auditing: PWA compliance scoring
- Offline Simulation: Network throttling tests
- Cross-Browser Validation: Feature support matrix

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.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * Rakshit5467 / Repository name * Currency Converte

Your new repository will be created as Currency-Converte.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about super-duper-octo-system ?

Description (optional)

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:

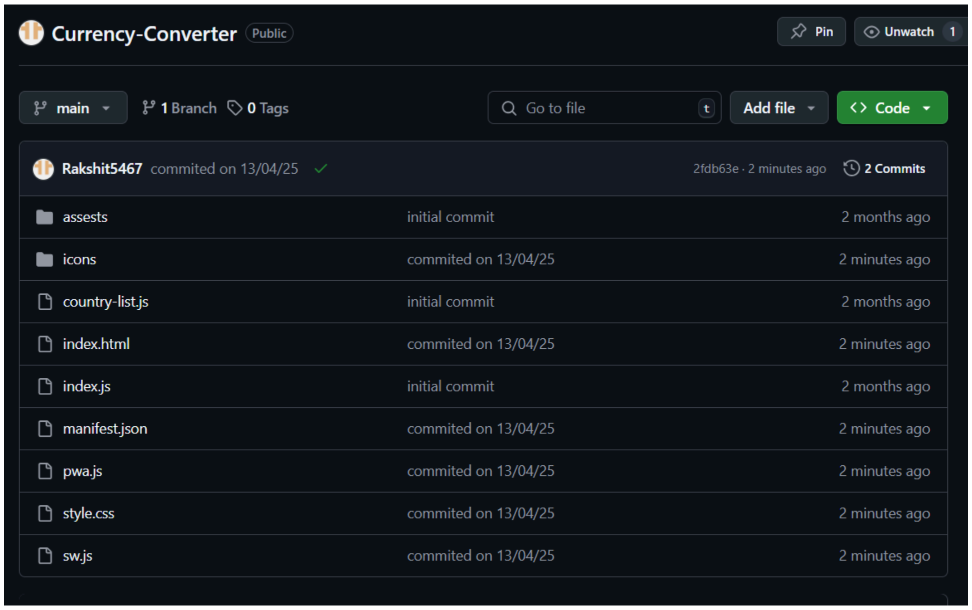
☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

2. Commit all the files to the repository



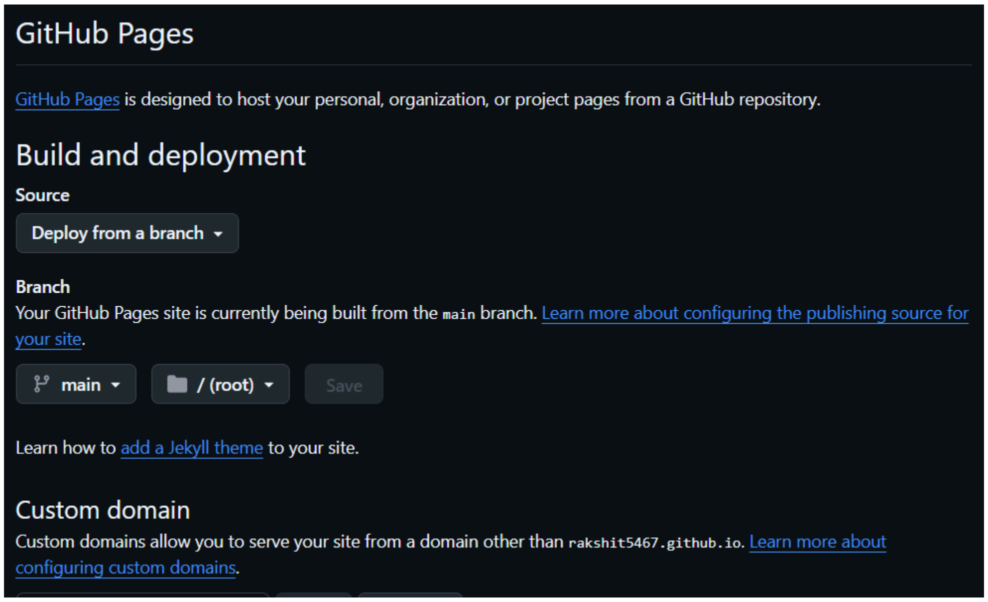
Currency-Converter Public

main · 1 Branch · 0 Tags

Rakshit5467 commited on 13/04/25 · 2fdb63e · 2 minutes ago · 2 Commits

assests	initial commit	2 months ago
icons	commited on 13/04/25	2 minutes ago
country-list.js	initial commit	2 months ago
index.html	commited on 13/04/25	2 minutes ago
index.js	initial commit	2 months ago
manifest.json	commited on 13/04/25	2 minutes ago
pwa.js	commited on 13/04/25	2 minutes ago
style.css	commited on 13/04/25	2 minutes ago
sw.js	commited on 13/04/25	2 minutes ago

3. Navigate to Settings -> Pages
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

Your GitHub Pages site is currently being built from the `main` branch. Learn more about configuring the publishing source for your site.

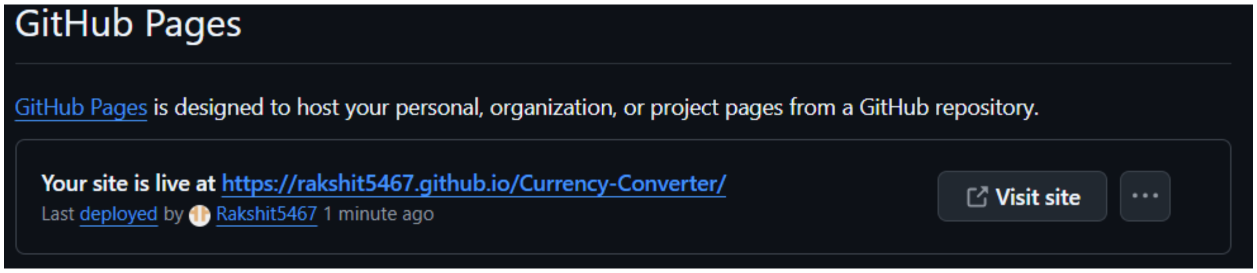
main · / (root) · Save

Learn how to add a Jekyll theme to your site.

Custom domain

Custom domains allow you to serve your site from a domain other than `rakshit5467.github.io`. Learn more about configuring custom domains.

6. 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://rakshit5467.github.io/Currency-Converter/

Last deployed by Rakshit5467 1 minute ago

Github link: https://github.com/Rakshit5467/Currency-Converter.git

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.