About PWA

**Progressive Web App**

* A progressive web app (PWA) is a type of web application that combines the capabilities of a website and a mobile app, providing an app-like experience to users while being accessible through a web browser. PWAs can be installed on a user's device, work offline, and have access to device features, making them fast, reliable, and engaging.
* A challenge for platform-specific apps is that they are not compatible with multiple platforms and devices, so it's not easy, if even possible, to move an Android app to iOS or an iOS to Windows or ChromeOS without creating a new app from scratch.
* Features: Linkability, Accessible by default, Ubiquitous, Easy to Deploy, Easy to Update
* Progressive Web Apps can also be listed in many app stores such as Google Play Store (for Android), Microsoft Store (for Windows) and Apple AppStore (for iOS, iPadOs and macOS)
* [Read more about PWA](https://web.dev/learn/pwa/progressive-web-apps/)

**Browser and OS Support**

* iOS and iPadOS

Safari (since iOS 11.3), AppStore (since iOS/iPadOS 14, with some limitations), mobile configuration for enterprise distribution.

* Android

Firefox, Google Chrome, Samsung Internet, Microsoft Edge, Opera, Brave, Huawei Browser, Baidu, UCWeb, Play Store (from version 72 with Google Chrome installed, or browsers compatible with TWA), Galaxy Store, Managed Play iframe for enterprise distribution.

**Note:** On iOS and iPadOS, PWAs are only installable if the user is using Safari. That means that users won't install PWAs if they use different browser apps, such as Google Chrome, Firefox, or Microsoft Edge. On both Android and iOS, users can't install PWAs from many in-app browsers, such as Facebook Mobile Browser, Instagram, Google Search App, or Gmail.

**Note:** WebSQL and ApplicationCache these two solutions are deprecated, and you shouldn't use them in your PWA.

**Service Workers**

* A PWA can store its assets in Cache Storage and IndexedDB, now in case of delivering fast and offline experience Service workers come into play, you can serve assets without going to the network and send notifications to a user and a badge to your PWA icon. Content can be upgraded in background and even make your whole PWA work offline.
* [Read more about Service Workers](https://web.dev/learn/pwa/service-workers/)

Setup for basic PWA app:

Git Repository: <https://github.com/Vinayselukar21/PWA-simple-starter>

**Step 1: Initialize a Next.js Project**

* First create a new Next.js project using the following command:

**npx create-next-app next-pwa**

* Then navigate to the project folder and install all the dependencies:

**cd next-pwa**

followed by **npm i**

(you can use any package manager npm, pnpm or yarn)

**Step 2: Add necessary module**

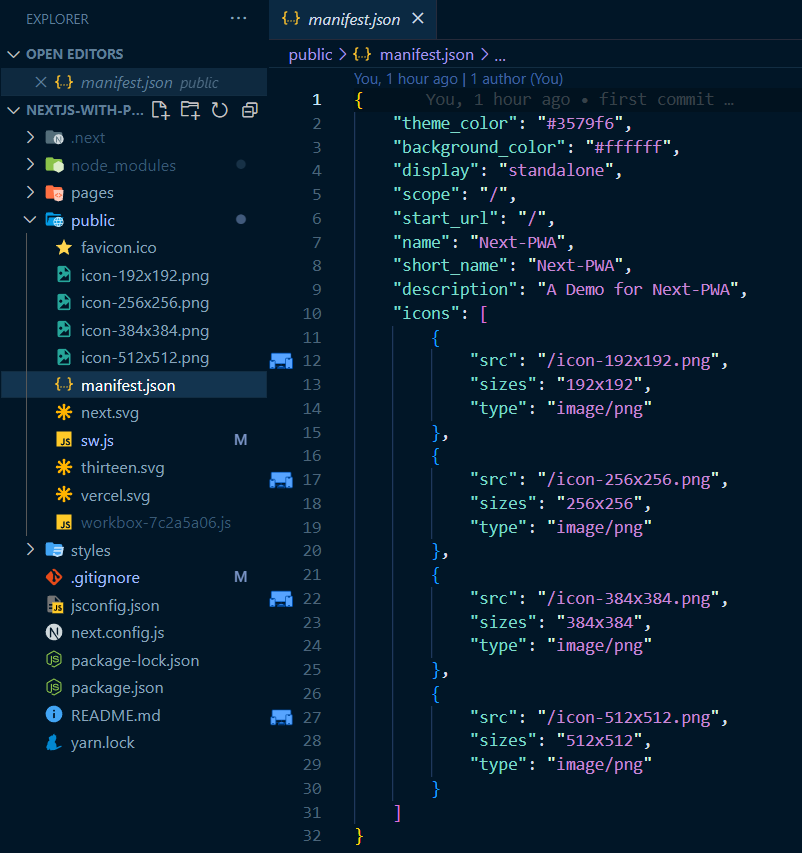
* To add PWA capabilities to your Next.js project we will use **next-pwa** package:

**npm i next-pwa**

Source**:** [next-pwa - npm (npmjs.com)](https://www.npmjs.com/package/next-pwa)

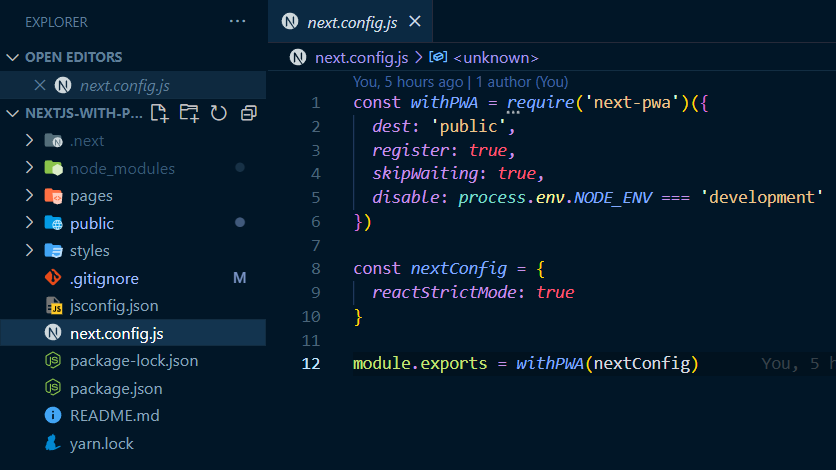
**Step 3: Add icons and Create the Manifest file**

* Create a new file called manifest.json inside the public folder of your project and add the following rules to define your pwa along with icons
* You can also create your manifest.json file for your project from: <https://www.simicart.com/manifest-generator.html/> (With the manifest.json file it will include icons also)

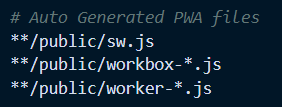


**Step 4: Configure Next.js PWA**

* Configure your next.config.js file to make your project installable with following content:
* This configuration tells **next-pwa** to generate the necessary service worker and store it in the public folder.

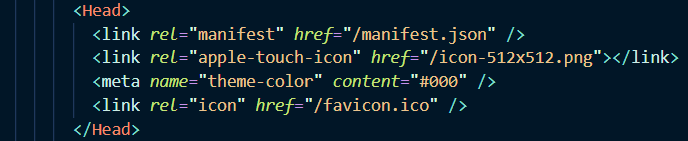


* Now if you use git to manage the source code you will need to add below rules to **.gitignore** file:



Step 5: Update \_document.js to add Head Meta

* Now for letting your project know where the manifest file and the icons are, In the pages folder open \_document.js and add the following code inside the <Head> component which will look like this.

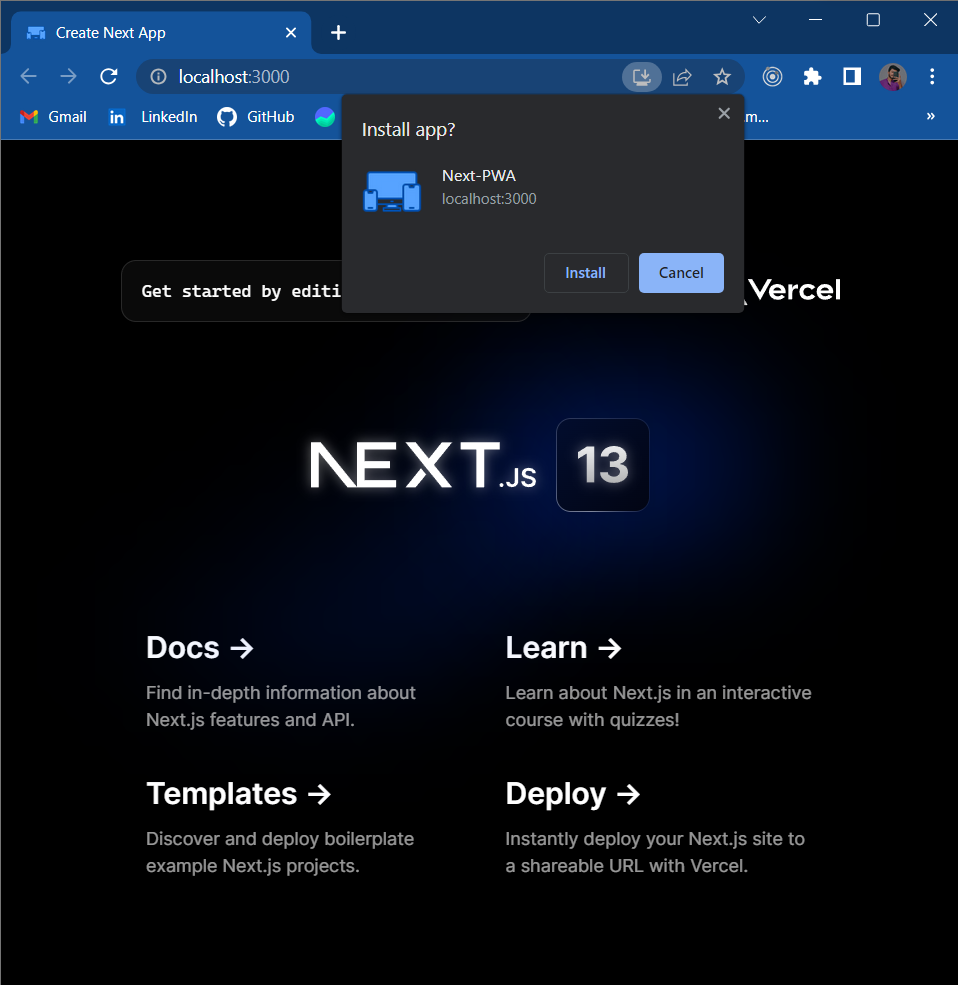


**Step 6: Testing your PWA Locally**

* Once you complete all the steps it is time to test your pwa using command **npm run build**.
* Now in root directory a **.next** folder is created which contains all your build files now start the server by running command **npm start**

**Output:** This is the output you should be able to se

**Link:** [**https://pwa-simple-starter.vercel.app/**](https://pwa-simple-starter.vercel.app/)

Desktop View Mobile View

Reference Document: [Learn PWA (web.dev)](https://web.dev/learn/pwa/)