Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

Theory:

Introduction to Progressive Web Apps (PWAs)

A Progressive Web App (PWA) is a modern web application that offers an experience similar to native mobile applications. PWAs combine the best features of the web and mobile apps, including offline access, responsive design, fast loading times, and installation capabilities. One key feature of PWAs is the “Add to Homescreen” functionality, which allows users to install the web app directly onto their device’s home screen, launching it in standalone mode like a native app.

Importance of the Web App Manifest

To enable the “Add to Homescreen” feature, a PWA must include a Web App Manifest file, typically named manifest.json. This JSON file provides essential metadata about the application. It includes information such as the application’s name, short name, icons, start URL, display mode, background color, and theme color. When a browser detects the manifest file and a registered service worker, it considers the web app installable and prompts the user with an option to add it to their home screen.

Structure and Role of the Manifest File

The manifest file plays a critical role in defining how the app appears when installed. The name and short_name fields define how the app will be labeled on the device. The start_url specifies the page that will open when the app is launched. The display property is often set to "standalone" to make the app open in a fullscreen-like mode without browser UI elements. The theme_color and background_color enhance the user interface experience by defining the color schemes of the title bar and the background screen. The icons array provides different image sizes so that devices can select the appropriate icon resolution for their screen.

Integration with HTML and Service Worker

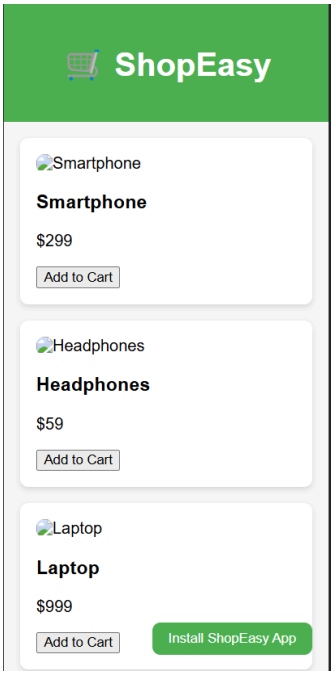
In the HTML file, the manifest is linked using the <link rel="manifest" href="manifest.json"> tag in the <head> section. The theme color is also defined using a <meta name="theme-color"> tag to align the browser’s UI with the app's branding. Alongside the manifest, a service worker script must be registered to provide offline support and caching. The service worker handles the installation of resources and intercepts network requests to serve cached content when offline.

Handling the Install Prompt

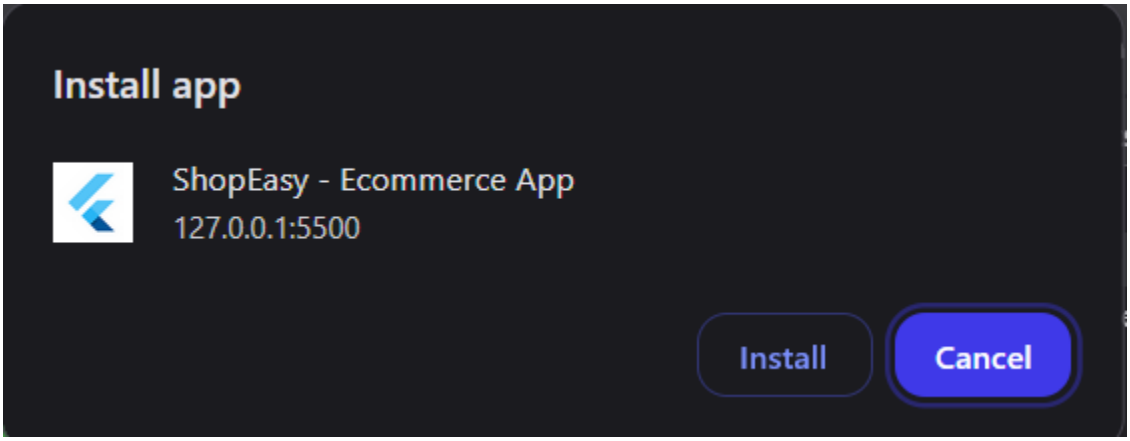
Modern browsers emit a beforeinstallprompt event when the app meets the installation criteria. This event can be captured and used to display a custom install button. When the user clicks the install button, the app calls prompt() on the stored event, allowing the user to install the app. The response to the prompt is then handled to check whether the user accepted or dismissed the installation.

Code: https://github.com/Sairam-Vk-sudo/mplExp27/tree/main/7

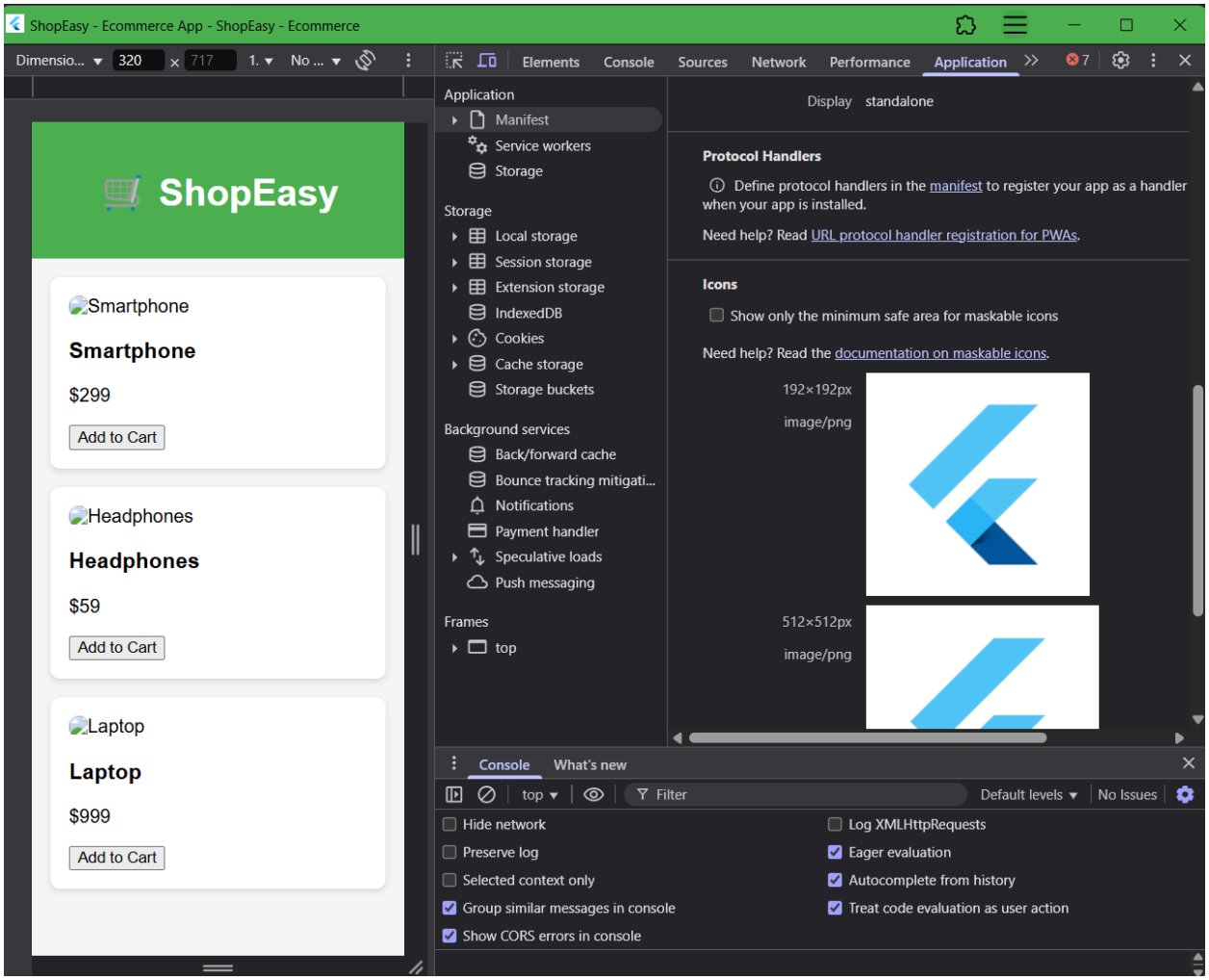
Output:



Install App Button



Chrome prompt to install app



Installed app running

Conclusion: Implementing a manifest file and handling the install prompt are essential steps in transforming a traditional web application into a Progressive Web App. This setup not only enables the “Add to Homescreen” feature but also improves user engagement and accessibility by offering an app-like experience on any device. Through proper metadata configuration and service worker integration, web apps like the Ecommerce PWA "ShopEasy" can offer a seamless, installable experience to users.