<u>EXPERIMENT- 9</u>

Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable "add to home screen feature".

Lab Outcome: Understand various PWA frameworks and their requirements.

Theory:

1. ***Differentiate between application, PWA and web view.***

	Application	**PWA (Progressive Web App)**	**Web View**
Platform	Native app	Web-based app that runs in a browser, but can be installed on a device	A component of a native app that displays web content
Installation	Installed from app store	Can be installed from a website or app store	Not installed, accessed through a native app
Accessibility	Accessed from home screen or app drawer	Accessed from home screen or app drawer, with the ability to work offline	Accessed through a native app menu
Functionality	Access to device hardware and features	Limited access to device hardware and features	No access to device hardware and features
Updates	Requires app store update	Updates automatically through a service worker	No updates, uses the web content provided by the native app
Development	Developed natively for each platform	Developed using web technologies such as HTML, CSS, and JavaScript	Developed natively for each platform to display web content
Installation Size	Can be larger due to the need to package native code	Smaller due to being web-based, typically less than 10MB	Not installed, uses the native app to display web content
Performance	Generally faster due to being compiled natively for each platform	Can be slower than native apps due to running in a browser, but can be optimized with techniques such as lazy loading	Similar to the performance of the native app since it uses the same rendering engine

Accessibility	Can use native accessibility features such as VoiceOver on iOS or TalkBack on Android	Accessibility can be more limited due to running in a browser, but can still use features such as ARIA attributes	Accessibility depends on the capabilities of the native app rendering the web view
Discoverability	Listed in app stores and can be promoted through various channels	Can be discoverable through search engines and shared links	Not discoverable, as it is a component of a native app
Cost	Development costs can be higher due to the need to develop for multiple platforms	Development costs can be lower since PWAs are built using web technologies that are familiar to many developers	Development costs can be lower since web views are built using web technologies, but there may be additional costs to integrate them into the native app
Offline Capabilities	Limited offline capabilities since many features require network connectivity	Can work offline to some extent using service workers and caching	No offline capabilities, as it is a component of a native app that requires network connectivity

2. ***Characteristics of PWA.***

Progressive Web Apps (PWA) have several characteristics that distinguish them from traditional web applications. Here are some of the key characteristics of PWA:

1. Progressive Enhancement: PWA is designed to work progressively with all devices and browsers, regardless of their capabilities. This means that even if the user's device or browser does not support all the features of PWA, they will still be able to access the content.
2. App-like Experience: PWA provides a native app-like experience to users by using features such as push notifications, offline support, and an app-like user interface.
3. Responsiveness: PWA adapts to different screen sizes and resolutions, making it suitable for use on any device, including desktops, tablets, and smartphones.
4. Connectivity Independent: PWA can work offline or on low-quality networks by using Service Workers, which allows the app to load and store cached data in the background.
5. Discoverability: PWA can be discovered by search engines and shared through URLs, making them easily accessible to users.
6. Installable: PWA can be installed on the user's device and accessed from the home screen, just like a native app, without requiring the user to go through an app store.
7. Secure: PWA uses HTTPS, which ensures that all communication between the app and the server is secure and encrypted.
8. Easy to Update: PWA updates automatically, without requiring any action from the user, ensuring that the app is always up-to-date.

9. Cross-platform: PWA can be built to work on different platforms, including iOS, Android, and desktop, using a single codebase.
10. Cost-effective: PWA development and maintenance cost are generally lower than that of native apps, making it a cost-effective solution for businesses.
11. Usage of phone features: PWAs have a lot of possibilities to access device features on Android and a few less on iOS. The usage of camera, GPS, or fingerprint scanners in an app-like way enriches the user's experience.

3. ***What is web manifest and HTTP Service worker?***

Web Manifest:

Web Manifest is a JSON file that provides metadata about the Progressive Web App (PWA). The manifest file includes information such as the PWA name, icon, start URL, background color, theme color, etc. It allows the PWA to be added to the user's home screen, launch as a standalone app, and work offline. The manifest file must be included in the HTML document using a link tag with the "rel" attribute set to "manifest".
Here are some key features of the Web Manifest:

1. App name: The name of the PWA that will be displayed on the user's home screen and app drawer.
2. Short name: A shorter version of the app name to be used when there is limited space available.
3. Start URL: The URL that the PWA should open when launched from the home screen.
4. Display mode: The display mode specifies how the PWA should be displayed. The available options are Fullscreen, standalone, minimal-ui, and browser.
5. Icons: The icons array specifies the icons that should be used for the PWA. Different sizes of icons are required to support different devices.

HTTP Service Worker:

A service worker is a JavaScript file that runs separately from the main thread of the browser. It is responsible for intercepting and handling network requests made by the PWA. A service worker allows the PWA to work offline, send push notifications, and perform other background tasks.
The HTTP service worker works in conjunction with the service worker to cache HTTP responses from the server. This caching enables the PWA to work offline and reduce the network usage for subsequent requests. When a network request is made, the service worker intercepts it and checks if there is a cached response available. If a cached response is found, it is returned immediately, without the need for a network request. If there is no cached response available, the service worker performs a network request and caches the response for future use.

Here are some key features of the HTTP service worker:

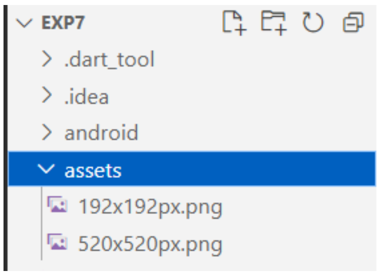
1. Cache API: The cache API is used by the service worker to cache HTTP responses from the server.
2. Network interception: The service worker intercepts network requests made by the PWA and determines whether to return a cached response or perform a network request.
3. Background sync: The service worker can perform background sync tasks, such as sending data to the server when the network is available.
4. Push notifications: The service worker can receive and handle push notifications, even when the PWA is not open in the browser.
5. HTTPS requirement: Service workers and HTTP service workers require HTTPS to ensure the security of the PWA and prevent man-in-the-middle attacks.

Practical:

Step 1: From the previous experiment we will get the website which is responsive.

Step2 : Create icons of 192x192 and 512x512 size.

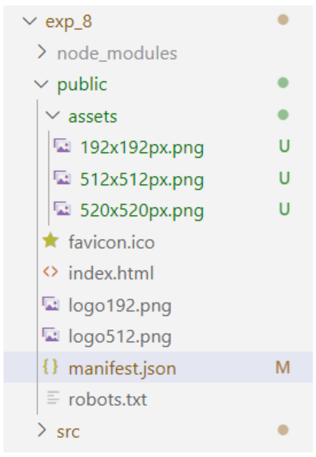
Step 3: Create the assets folder and add icons in the assets folder.



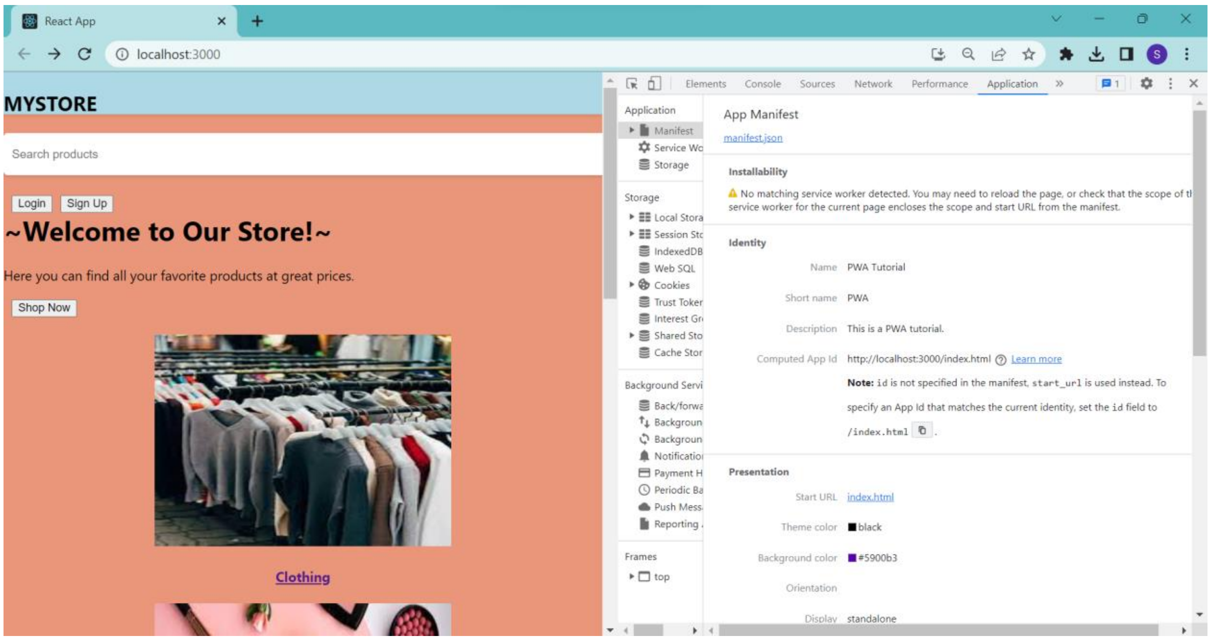
Step 4: Add the manifest.json file.

```
{
  "name":"PWA Tutorial",
  "short_name":"PWA",
  "start_url":"index.html",
  "display":"standalone",
  "background_color":"#5900b3",
  "theme_color":"black",
  "scope": ".",
  "description":"This is a PWA tutorial.",
  "icons":[
  {
  "src":"assets/192x192px.png",
  "sizes":"192x192",
  "type":"image/png"
  },
  {
  "src":"assets/512x512px.png",
  "sizes":"512x512",
  "type":"image/png"
  }
]
}
```

Outcome:



```
exp_8
  node_modules
  public
    assets
      192x192px.png      U
      512x512px.png      U
      520x520px.png      U
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json        M
    robots.txt
  src
```


React App

localhost:3000

MYSTORE

Search products

Login Sign Up

~Welcome to Our Store!~

Here you can find all your favorite products at great prices.

Shop Now

Clothing

Elements Console Sources Network Performance Application

Application: Manifest, Service Wo, Storage

Storage: Local Stora, Session Sto, IndexedDB, Web SQL, Cookies, Trust Toker, Interest Gr, Shared Sto, Cache Stor

Background Servi: Back/forwa, Backgroun, Backgroun, Notificatio, Payment H, Periodic Ba, Push Mess, Reporting

Frames: top

App Manifest

manifest.json

Installability

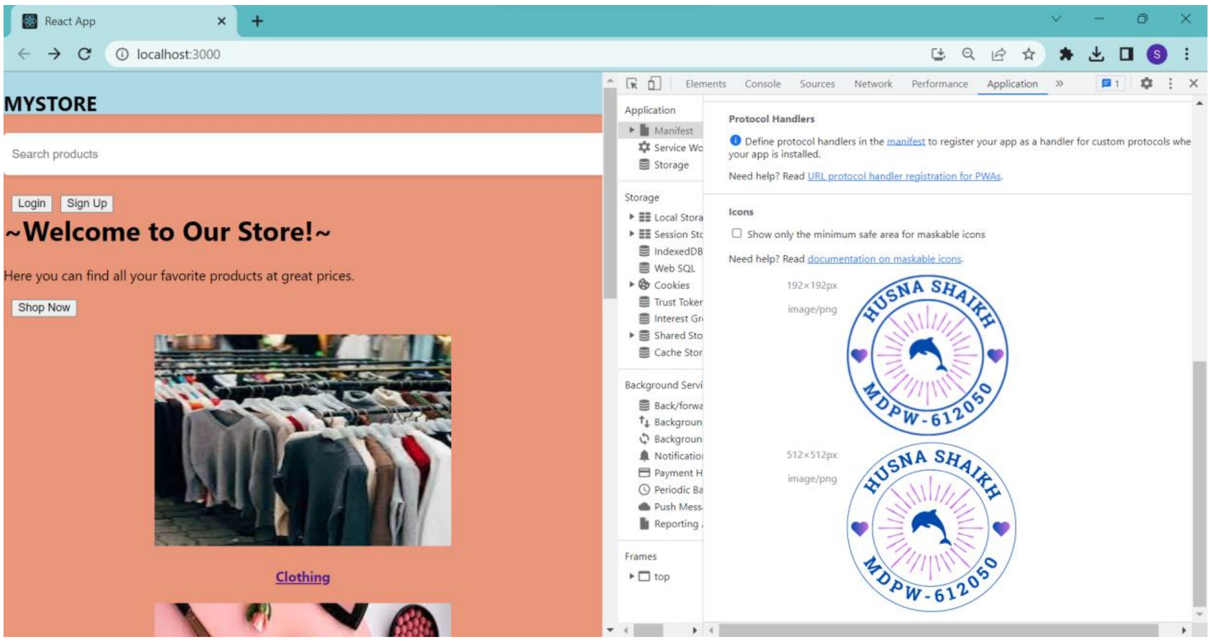
⚠ No matching service worker detected. You may need to reload the page, or check that the scope of t service worker for the current page encloses the scope and start URL from the manifest.

Identity

Name	PWA Tutorial
Short name	PWA
Description	This is a PWA tutorial.
Computed App Id	http://localhost:3000/index.html Learn more
	Note: id is not specified in the manifest, start_url is used instead. To specify an App Id that matches the current identity, set the id field to /index.html .

Presentation

Start URL	index.html
Theme color	■ black
Background color	■ #5900b3
Orientation	
Display	standalone


React App

localhost:3000

MYSTORE

Search products

Login Sign Up

~Welcome to Our Store!~

Here you can find all your favorite products at great prices.

Shop Now

Clothing

Elements Console Sources Network Performance Application

Protocol Handlers

ⓘ Define protocol handlers in the manifest to register your app as a handler for custom protocols whe your app is installed.

Need help? Read URL protocol handler registration for PWAs.

Icons

☐ Show only the minimum safe area for maskable icons

Need help? Read documentation on maskable icons.

192×192px image/png

HUSNA SHAIKH MDPW-612050

512×512px image/png

HUSNA SHAIKH MDPW-612050

Tools used: Visual Studio Code, React, Maifest.json file.

Conclusion:

From this experiment we have learned how to write metadata of our E-commerce PWA in a web app and learned how to view it.