Experiment No : 07

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

Theory:

Progressive Web Apps (PWAs):

Progressive Web Apps (PWAs) are web applications that leverage modern web capabilities to offer a user experience similar to native mobile apps. They are built using standard web technologies like HTML, CSS, and JavaScript but are designed to work on any platform with a standards-compliant browser, including desktop and mobile devices.

Regular web app

Regular web apps, on the other hand, are accessed and used through a web browser on various devices, such as desktops, laptops, tablets, and mobile phones. While they also use web technologies, they may lack some of the advanced features and user experience enhancements found in PWAs.

How Progressive Web Apps (PWAs) differ from regular web apps?

Progressive Web Apps (PWAs) differ from regular web apps in several key aspects, which contribute to providing a more app-like experience for users. Here's how PWAs differ from regular web apps:

1. **Offline Functionality:** PWAs can work offline or with a poor internet connection, thanks to service workers. This enables features like caching of assets and data, allowing PWAs to continue functioning even when the user is offline. Regular web apps typically require an active internet connection to function properly.

2. **App-like User Interface:** PWAs can be installed on the device's homescreen and launched in full-screen mode, giving them a native app feel. They can also utilize features like push notifications, providing a more immersive and engaging user experience compared to regular web apps.

3. **Improved Performance:** PWAs are optimized for performance, offering faster loading times and smoother interactions. Techniques like lazy loading, code splitting, and caching help minimize loading times and reduce data transfer, resulting in a more responsive user experience.

4. **Discoverability and Shareability:** PWAs are discoverable through web search engines and can be shared via URL links. This makes it easy for users to discover and share PWAs with others. Regular web apps may have limited discoverability and shareability compared to PWAs.

5. **Cross-platform Compatibility:** PWAs are designed to work across different platforms and devices, ensuring a consistent user experience. They utilize responsive design principles to adapt to various screen sizes and orientations, making them a versatile solution for reaching users on different devices.

The below steps have to be followed to create a progressive web application:
Step 1: Create an HTML page that would be the starting point of the application. This HTML will contain a link to the file named manifest.json. This is an important file that would be created in the

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Twina Skincare</title>
  <link rel="stylesheet" href="style.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Playball&family=Roboto:ital,wght@0,100;0,300;
0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap" rel="stylesheet">
<link rel="manifest" href="manifest.json">
</head>
<body>
  <header class="pink-header">
    <a href="#home">
      <img src="images/logo.png" alt="Twina Skincare">
    </a>
    <h1 class="logo-text">Twina Skincare</h1>

    <div class="top-nav">
      <input type="text" placeholder="Search...">
      <a href="#" class="button black">Cart</a>
      <a href="#" class="button black">Wishlist</a>
      <a href="#" class="button black">Login</a>
      <a href="#" class="button black">Signup</a>
    </div>
  </header>

  <main>
    <section class="hero" style="background-image: url('images/background-image.png');
background-size: cover; background-position: top;">
    </section>

    <section id="products">
      <h2>Our Skincare Products</h2>

      <div class="product-grid">
        <section class="product">
          <img src="images/lanaige.png" alt="Product 1 Image">
```

```html
    <h3>Lanaige lip Mask</h3>
    <p>A 24hr hydrating lip Mask</p>
    <span>&#8377; rs/999</span>
    <div class="product-actions">
      <button class="add-to-cart">Add to Cart</button>
      <button class="add-to-wishlist">Add to Wishlist</button>
    </div>
  </section>

  <section class="product">
    <img src="images/snailmucin.png" alt="Product 2 Image">
    <h3>Cosrx Snail mucin</h3>
    <p>Get Glass skin in one go.</p>
    <span>&#8377;rs. 1900</span>
    <div class="product-actions">
      <button class="add-to-cart">Add to Cart</button>
      <button class="add-to-wishlist">Add to Wishlist</button>
    </div>
  </section>

  <section class="product">
    <img src="images/paulaschoice.png" alt="Product 3 Image">
    <h3>Paulas choice BHA</h3>
    <p>Super cleanse your skin chemically</p>
    <span>&#8377;rs.2200</span>
    <div class="product-actions">
      <button class="add-to-cart">Add to Cart</button>
      <button class="add-to-wishlist">Add to Wishlist</button>
    </div>
  </section>

  <section class="product">
    <img src="images/cetaphil.png" alt="Product 4 Image">
    <h3>Cetaphil foaming cleanser</h3>
    <p>Get rid of oily Skin</p>
    <span>&#8377; rs.599</span>
    <div class="product-actions">
      <button class="add-to-cart">Add to Cart</button>
      <button class="add-to-wishlist">Add to Wishlist</button>
    </div>
  </section>

  <section class="product">
```

```html
            <img src="images/moisturesurge.png" alt="Product 5 Image">
            <h3>Clinque Moisture Surge</h3>
            <p>Heaven in form of a moisturizer</p>
            <span>&#8377; rs.3700</span>
            <div class="product-actions">
              <button class="add-to-cart">Add to Cart</button>
              <button class="add-to-wishlist">Add to Wishlist</button>
            </div>
          </section>

          <section class="product">
            <img src="images/ordinary.png" alt="Product 6 Image">
            <h3>Ordinary Multipeptide Serum</h3>
            <p>Youthful skin is just a serum away.</p>
            <span>&#8377; rs.999</span>
            <div class="product-actions">
              <button class="add-to-cart">Add to Cart</button>
              <button class="add-to-wishlist">Add to Wishlist</button>
            </div>
          </section>

      </div>

      <a href="#" class="view-all-products">View All Products</a>
    </section>

    <section id="about-us">
      <h2>About Us</h2>
      <p>Welcome to Twina Skincare, your one-stop destination for high-quality skincare
products. We are dedicated to providing our customers with natural and effective skincare
solutions that enhance their beauty and confidence.
      At Twina Skincare, we believe in the power of nature. That's why all our products are
carefully crafted using the finest natural ingredients, sourced ethically from around the
world. Each product is formulated to deliver visible results while nourishing and
rejuvenating your skin.
      Our team of skincare experts is passionate about helping you achieve your skincare
goals. Whether you're looking for anti-aging treatments, acne solutions, or simply want to
pamper your skin, we have the perfect products for you.
      Customer satisfaction is our top priority, and we strive to provide an exceptional
shopping experience. From our user-friendly website to our fast and reliable shipping, we
are committed to making your skincare journey enjoyable and hassle-free.</p>
      Thank you for choosing Twina Skincare. We look forward to helping you discover the
beauty of natural skincare!</p>
```

```
    </section>
  </main>

  <footer>
    <p>&copy; Twina Skincare 2024</p>
  </footer>
</body>
</html>
```

Step 2: Create a manifest.json file in the same directory. This file basically contains information about the web application. Some basic information includes the application name, starting URL, theme color, and icons. All the information required is specified in the JSON format. The source and size of the icons are also defined in this file.

```
{} manifest.json ✕    <> index.html

{} manifest.json > ...
  1
  2    {
  3        "name": "Twina Skincare",
  4        "short_name": "Twina",
  5        "description": "Discover high-quality skincare products at Twina Skincare. We offer
  6        "icons": [
  7          {
  8            "src": "images/logo.png",
  9            "sizes": "512x512",
 10            "type": "image/png"
 11          },
 12          {
 13            "src": "images/ordinary.png",
 14            "sizes": "512x512",
 15            "type": "image/png"
 16          }
 17        ],
 18        "start_url": "/",
 19        "display": "standalone",
 20        "background_color": "#ffc0cb",
 21        "theme_color": "#f7b5b0"
 22    }
 23
```

Step 3:

Create a new folder named images and place all the icons related to the application in that folder. It is recommended to have the dimensions of the icons at least 192 by 192 pixels and 512 by 512 pixels. The image name and dimensions should match that of the manifest file.
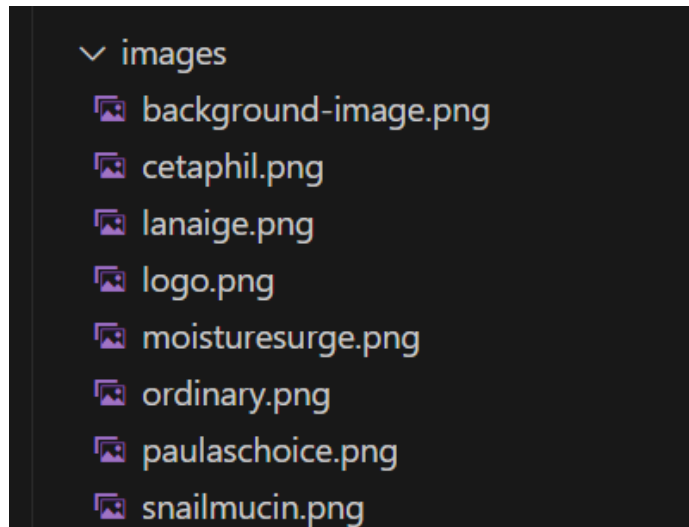
Step 4: Serve the directory using a live server so that all files are accessible.
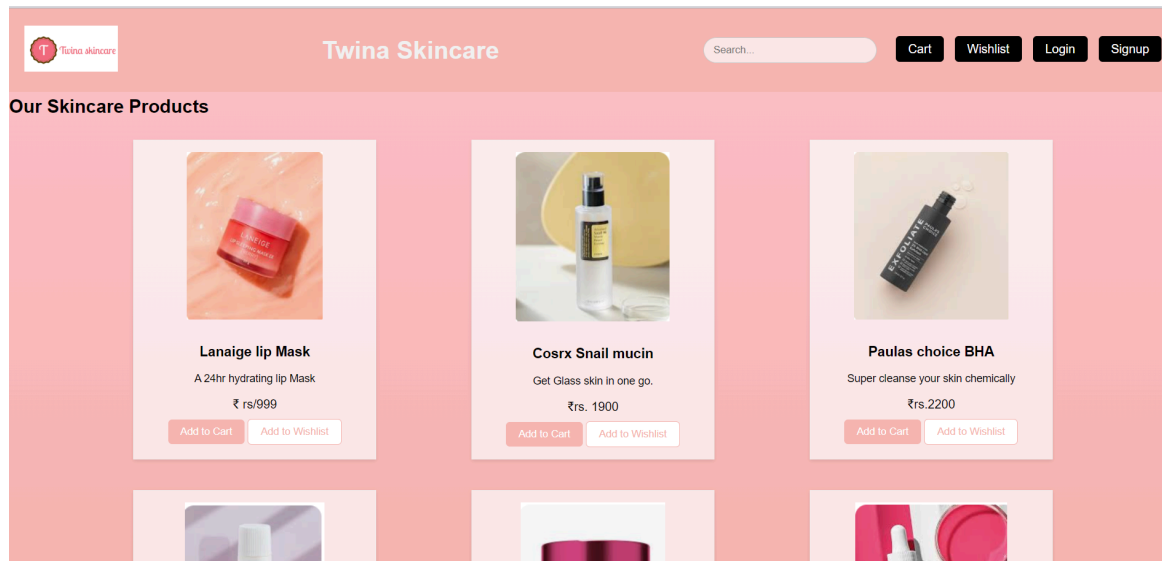
Step 5: Open the index.html file in Chrome navigate to the Application Section in the Chrome Developer Tools. Open the manifest column from the list.

Step 6: Under the installability tab, it would show that no service worker is detected. We will need to create another file for the PWA, that is, serviceworker.js in the same directory. This file handles the configuration of a service worker that will manage the working of the application.

```javascript
// Cache name
const CACHE_NAME = 'twina-skincare-cache-v1';

// Files to cache
const urlsToCache = [
  '/',
  '/index.html',
  '/style.css',
  '/images/logo.png',
  '/images/background-image.png',
  '/images/lanaige.png',
  '/images/snailmucin.png',
  '/images/paulaschoice.png',
  '/images/cetaphil.png',
  '/images/moisturesurge.png',
  '/images/ordinary.png',
];

// Install service worker
self.addEventListener('install', function(event) {
  // Perform installation steps
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
```

```javascript
      })
    );
});

// Fetch cached data
self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request)
      .then(function(response) {
        // Cache hit - return response
        if (response) {
          return response;
        }
        // Clone the request
        const fetchRequest = event.request.clone();
        // Make network request
        return fetch(fetchRequest).then(
          function(response) {
            // Check if we received a valid response
            if (!response || response.status !== 200 || response.type !==
'basic') {
              return response;
            }
            // Clone the response
            const responseToCache = response.clone();
            // Cache the fetched response
            caches.open(CACHE_NAME)
              .then(function(cache) {
                cache.put(event.request, responseToCache);
              });
            return response;
          }
        );
      })
  );
});

// Activate service worker
self.addEventListener('activate', function(event) {
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.filter(function(cacheName) {
```

```
      // Delete outdated caches
      return cacheName !== CACHE_NAME;
    }).map(function(cacheName) {
      return caches.delete(cacheName);
    })
  );
  })
 );
});
```

Step 7: The last step is to link the service worker file to index.html. This is done by adding a short JavaScript script to the index.html created in the above steps. Add the below code inside the script tag in index.html.

```
<script>
  if ('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
      navigator.serviceWorker.register('/service-worker.js')
        .then(function(registration) {
          console.log('ServiceWorker registration successful with scope: ', registration.scope);
        }, function(err) {
          console.error('ServiceWorker registration failed: ', err);
        });
    });
  }
</script>
</body>
```

Installing the application:

Navigating to the Service Worker tab, we see that the service worker is registered successfully and now an install option will be displayed that will allow us to install our app.

Click on the install button to install the application. The application would then be installed, and it would be visible on the desktop.
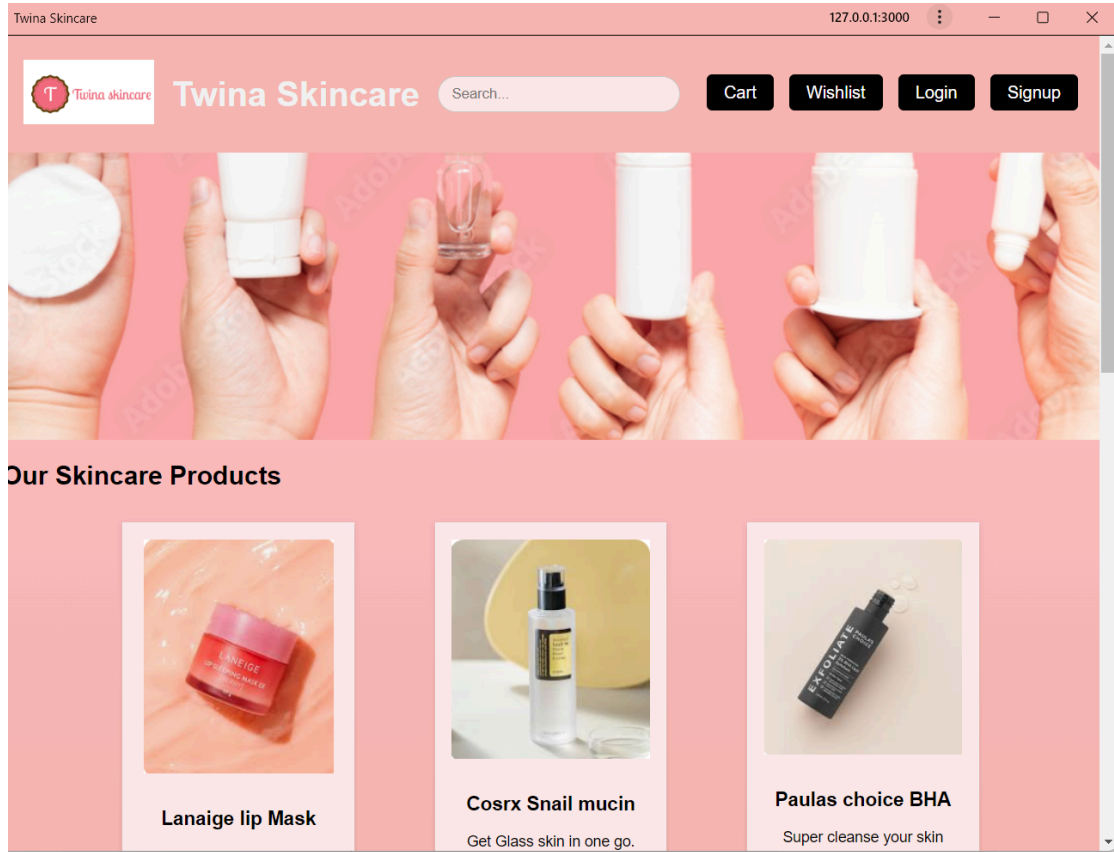
For installing the application on a mobile device, the Add to Home screen option in the mobile browser can be used. This will install the application on the device.

T

Twina Skincare

App

Open

Run as administrator

Open file location

Unpin from taskbar

Pin to Start

Uninstall

Conclusion: We successfully wrote metadata for the PWA, especially for an eCommerce application and enabled "add to homescreen" functionality.