

## **Experiment No. 7**

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

### **THEORY**

#### **Regular Web App**

A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.

#### **Progressive Web App**

Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience
2. Ease of Access
3. Faster Services
4. Engaging Approach
5. Updated Real-Time Data Access
6. Discoverable
7. Lower Development Cost

### **CODE:-**

#### **//Manifest.json :**

```
{  
  "name": "Language TutorI",  
  "short_name": "PWA",  
  "start_url": "index.html",  
  "display": "standalone",  
  "background_color": "#5900b3",  
  "theme_color": "black",  
  "scope": ".",  
  "description": "This is a PWA tutorial.",  
  "icons": [
```

```

        {
          "src": "images/icon-192x192.png",
          "sizes": "192x192",
          "type": "image/png"
        },
        {
          "src": "images/icon-512x512.png",
          "sizes": "512x512",
          "type": "image/png"
        }
      ]
    }
  }

//serviceworker.js
var staticCacheName = "pwa";
self.addEventListener("install", function (e) {
  e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
      return cache.addAll(["/"]);
    })
  );
});
self.addEventListener("fetch", function (event) {
  console.log(event.request.url);
  event.respondWith(
    caches.match(event.request).then(function (response) {
      return response || fetch(event.request);
    })
  );
});
}

//Index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta
    name="description"
    content="Web site created using create-react-app"
  />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<meta name="apple-mobile-web-app-status-bar" content="#aa7700">
<meta name="theme-color" content="black">

```

```

<!-- Manifest File link -->
<link rel="manifest" href="manifest.json">
  <title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>

<script>
  // Register the Service Worker
  async function registerSW() {
    // Check if browser supports Service Worker
    if ('serviceWorker' in navigator) {
      try {
        // Register the Service Worker named 'serviceworker.js'
        await navigator.serviceWorker.register('serviceworker.js');
      }
      catch (e) {
        // Log error message if registration fails
        console.log('SW registration failed');
      }
    }
  }
</script>

</html>

```

## OUTPUT :

//Open folder in VS code and run the website.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

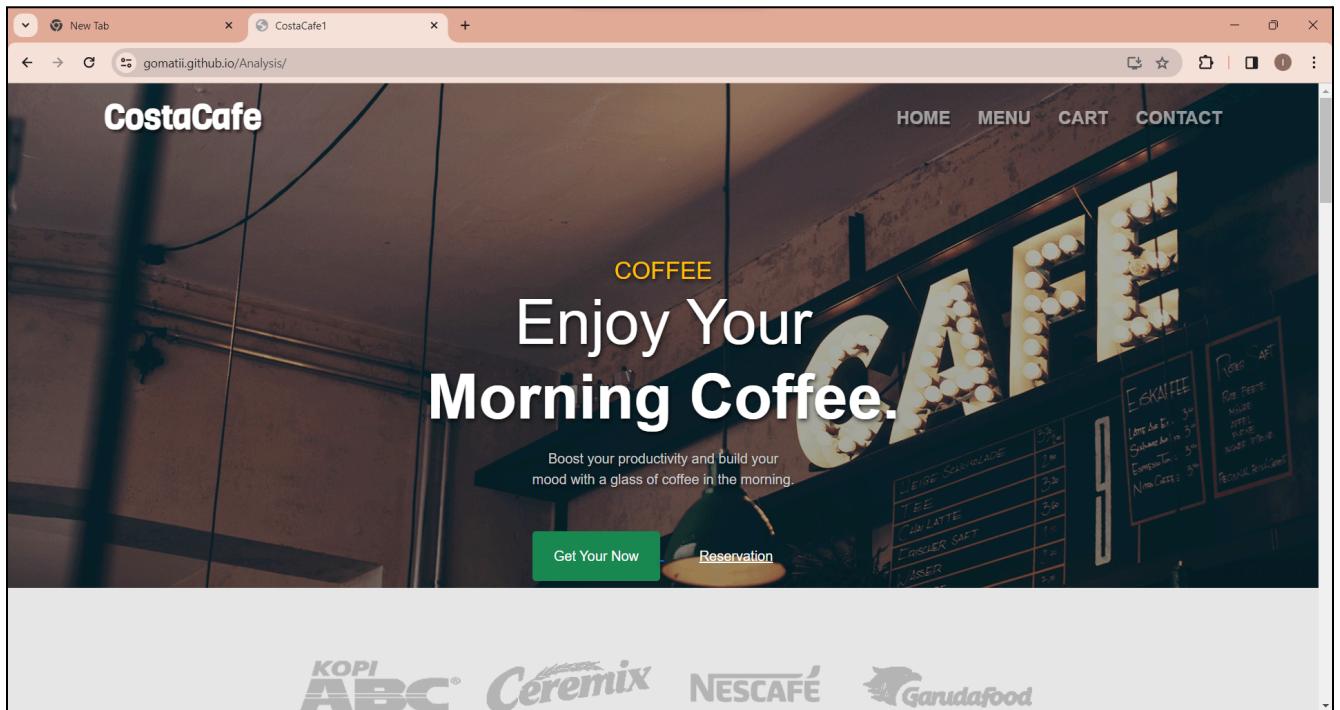
Local:          http://localhost:3000
Compiled successfully!

You can now view firebase-auth in the browser.

Local:          http://localhost:3000
On Your Network:  http://192.168.1.100:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

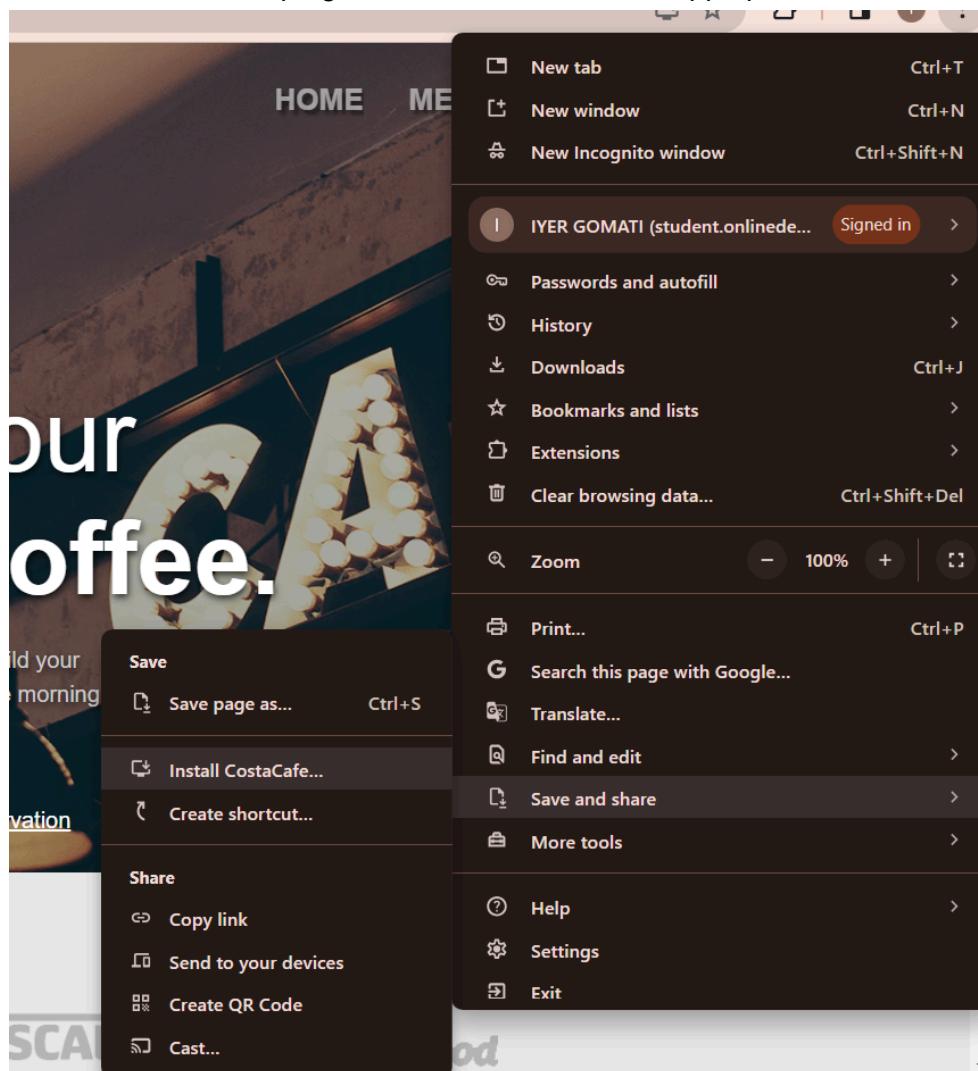
```



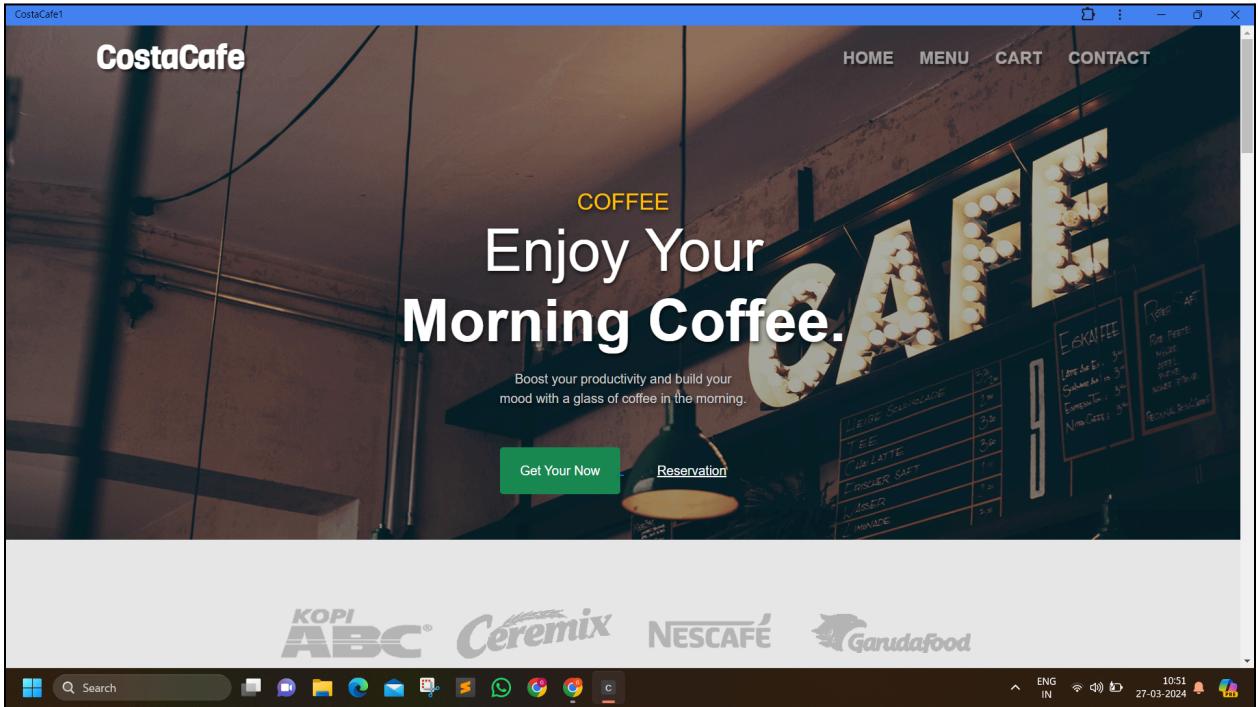
//open Developer tools options -> Applications

A screenshot of the Chrome Developer Tools Application tab, showing the configuration for the CostaCafe PWA. The left sidebar lists various application-related settings like Manifest, Service workers, Storage, and Background services. The right panel shows the "Identity" section with the app name set to "CostaCafe" and a computed App ID of "https://gomati.github.io/Analysis/index.html". The "Presentation" section includes fields for the Start URL ("index.html"), Theme color ("#ffd31d"), and Background color ("#333"). The "Protocol Handlers" section is currently empty. At the bottom, there's an "Easter egg" note mentioning a hidden feature behind a colorful emoji.

//click on 3 dots on top right corner of browser from app option install this site as an app



//App icon will appear at the bottom



### Conclusion:-

Hence, we learnt how to write a metadata of our E-commerce website PWA in a Web App Manifest File to enable add to homescreen feature.