

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) are a type of web application that combines the best features of web and mobile apps to offer users a seamless and engaging experience. Progressive Web Apps (PWAs) represent a new paradigm in web development, seamlessly blending the strengths of web technologies with the immersive user experience of native mobile apps. By leveraging responsive design principles, service workers for offline functionality, and the Web App Manifest for installation and customization, PWAs deliver fast, reliable, and engaging experiences across devices and network conditions. With features like push notifications, home screen installation, and seamless navigation, PWAs bridge the gap between web and native apps, offering users a frictionless journey and empowering developers to build versatile, accessible, and modern web applications.

Overview of PWAs:

- 1. Responsive Design:** PWAs are designed to be responsive, meaning they can adapt and provide a great user experience across various devices, including desktops, tablets, and smartphones.
- 2. Progressive Enhancement:** PWAs are built using progressive enhancement principles. This means they should work for all users, regardless of the browser or device they use. Advanced features are progressively enhanced for users with modern browsers and devices.
- 3. App-Like Experience:** PWAs aim to provide an app-like experience, including smooth navigation, offline functionality, push notifications, and the ability to be added to the home screen.
- 4. Service Workers:** One of the key technologies behind PWAs is service workers. These are scripts that run in the background and enable features like offline caching, background synchronization, and push notifications.
- 5. Web App Manifest:** The Web App Manifest is a JSON file that contains metadata about the PWA, such as its name, icons, start URL, display mode, theme colors, and more. This manifest file is used by browsers to install the PWA and customize its appearance on the user's device.
- 6. Offline Functionality:** PWAs can work offline or in low-connectivity environments by caching assets and data using service workers. This ensures that users can still access content and perform actions even when they are not connected to the internet.
- 7. Engagement Features:** PWAs can engage users through features like push notifications, which allow developers to send notifications to users even when the PWA is not actively open in their browser.

8. Fast and Reliable: PWAs are designed to be fast and reliable, providing quick load times and smooth interactions to enhance the user experience.

9. Secure: PWAs are served over HTTPS to ensure data security and protect user privacy, especially when dealing with sensitive information or transactions.

10. Cross-Platform Compatibility: PWAs are platform-agnostic and can run on various operating systems and browsers, reducing development effort and reaching a broader audience.

Overall, PWAs combine the reach and accessibility of the web with the capabilities and engagement of native apps, making them a compelling choice for modern web development.

Implementation:

1. Create an HTML file for your ecommerce website that will contain a link to the manifest.json file.

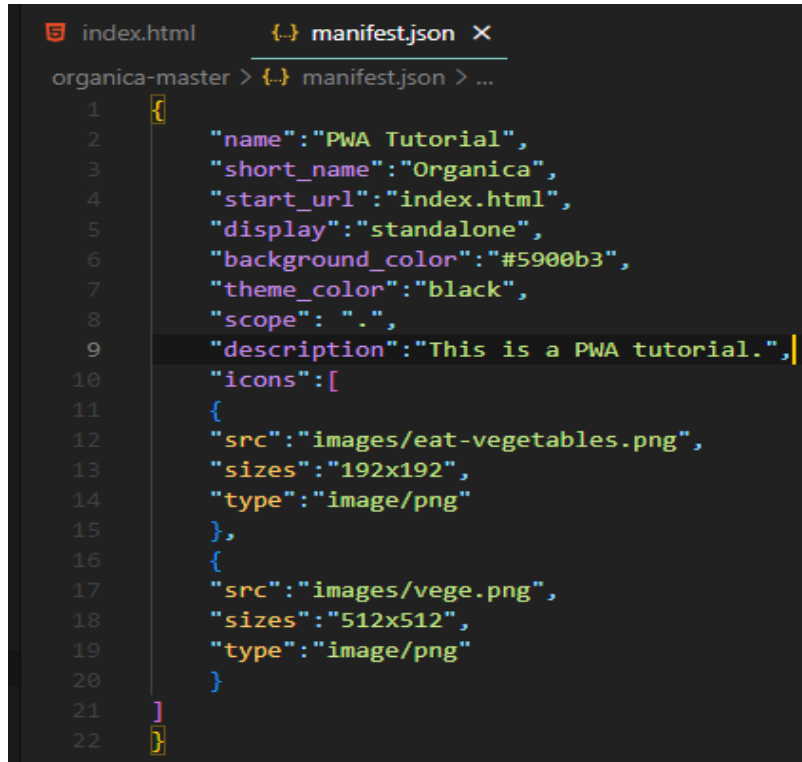
```
<!DOCTYPE html>

<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Organica - Quality organic fruit & vegetables.</title>
  <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">
  <link rel="stylesheet" href="./assets/css/main.css">
  <link rel="stylesheet" href="./assets/css/home.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=Playfair+Display:ital,wght@
0,400;0,700;1,400;1,700&family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">
  <link rel="manifest" href="manifest.json">
</head>

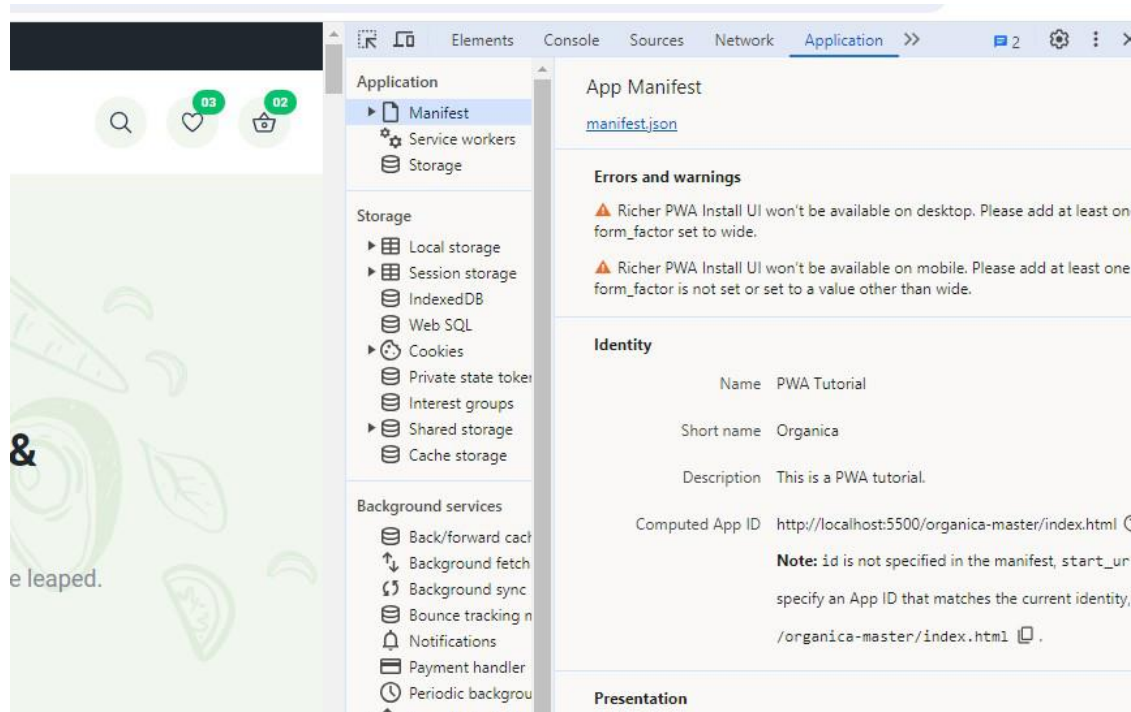
<body id="top">
  <header class="header" data-header>
```

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.

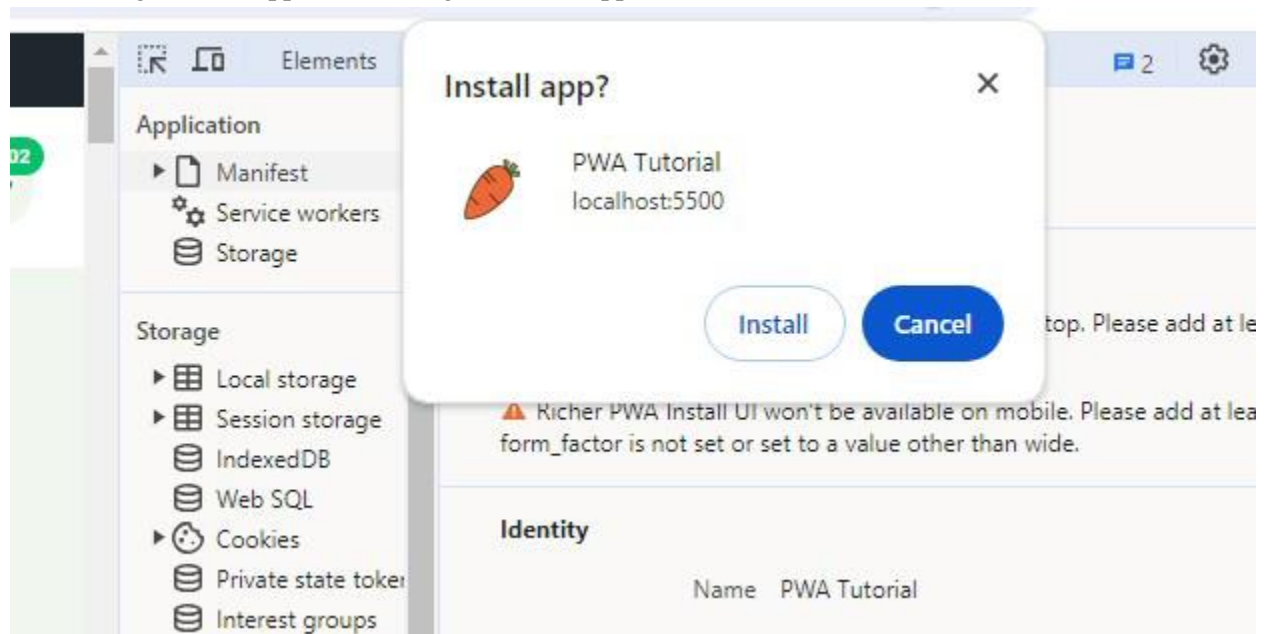
A screenshot of a code editor with a dark theme. The editor shows two tabs at the top: 'index.html' and 'manifest.json'. The 'manifest.json' tab is active. The breadcrumb navigation shows 'organica-master > {manifest.json} > ...'. The code is a JSON object with the following properties: 'name' (PWA Tutorial), 'short_name' (Organica), 'start_url' (index.html), 'display' (standalone), 'background_color' (#5900b3), 'theme_color' (black), 'scope' (.), 'description' (This is a PWA tutorial.), and 'icons' (an array of two icon objects). The first icon has a src of 'images/eat-vegetables.png', size of '192x192', and type of 'image/png'. The second icon has a src of 'images/vege.png', size of '512x512', and type of 'image/png'. Line numbers 1 through 22 are visible on the left side of the editor.

```
1  {
2    "name": "PWA Tutorial",
3    "short_name": "Organica",
4    "start_url": "index.html",
5    "display": "standalone",
6    "background_color": "#5900b3",
7    "theme_color": "black",
8    "scope": ".",
9    "description": "This is a PWA tutorial.",
10   "icons": [
11     {
12       "src": "images/eat-vegetables.png",
13       "sizes": "192x192",
14       "type": "image/png"
15     },
16     {
17       "src": "images/vege.png",
18       "sizes": "512x512",
19       "type": "image/png"
20     }
21   ]
22 }
```

3. Open the index.html file in Chrome navigate to the Application Section in the Chrome Developer Tools. Open the manifest column from the list.



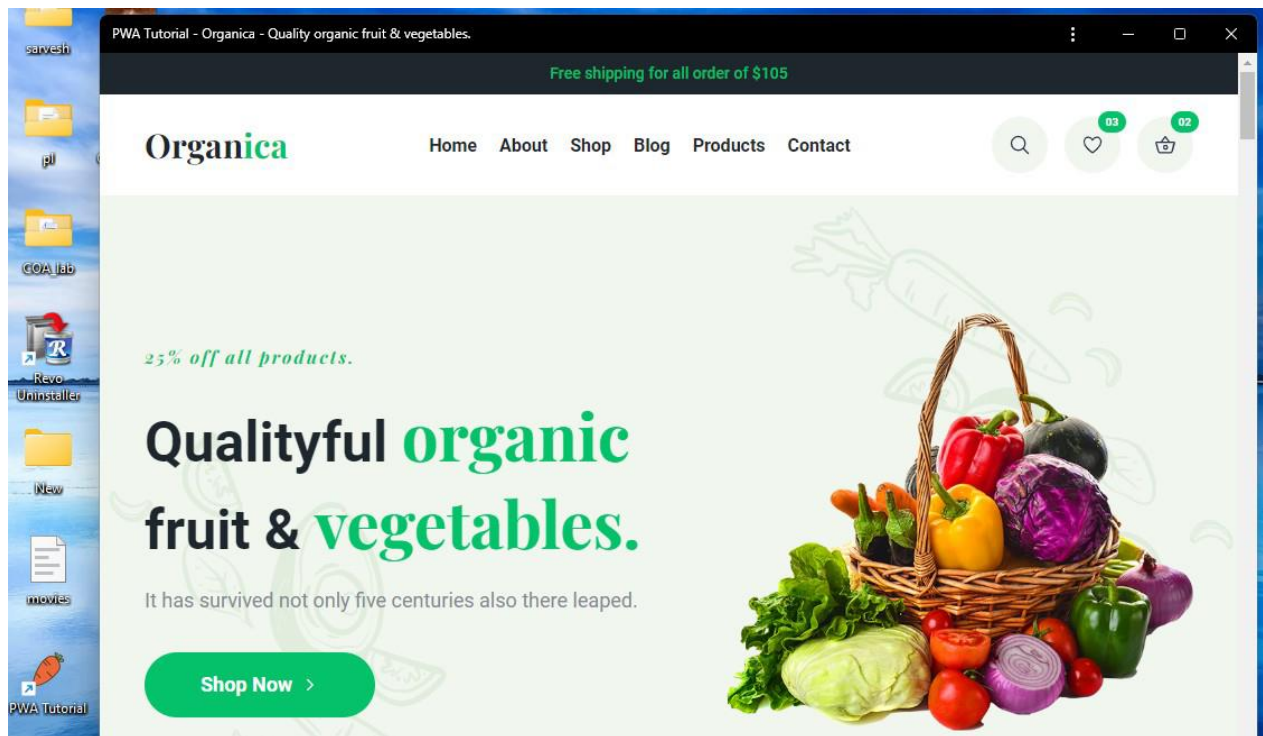
4. Installing the web application using the Install app button



5. Click on the install button to install the application. The application would then be installed, and it would be visible on the desktop.



Final Application:



Conclusion:

Hence we have understood and studied the working of Progressive web applications and the features of Progressive web applications. Also, we have implemented the add to homescreen feature in Progressive web applications on our e-commerce website. By crafting a well-structured manifest.json file with accurate metadata properties such as name, description, icons, and colors, developers can enhance the accessibility and user experience of their PWAs.