

**Name: Sujal.S.Tekwani**

**Class: D15B**

**Roll No:59**

## **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:**

#### **Making a Vite React App a Progressive Web App (PWA)**

A Progressive Web App (PWA) is a type of web application that provides a reliable, fast, and engaging user experience similar to a native mobile app. It leverages modern web technologies, including service workers, caching strategies, and a web app manifest, to enable offline functionality, push notifications, and an installable experience.

In the context of a web application, converting the app into a PWA involves integrating these key components:

#### 1. Service Workers:

- Service workers act as a proxy between the browser and the network, enabling background processes such as caching and offline capabilities.
- They allow the app to load even when there is no internet connection by storing static assets locally.

#### 2. Web App Manifest:

- The `manifest.json` file provides metadata about the app, including its name, icons, theme color, and display mode.
- This file helps the browser recognize the app as installable and controls its behavior when launched.

#### 3. HTTPS Hosting

- A secure environment is required for service workers and manifest to work.
- Always host your PWA over **HTTPS** to ensure integrity and security.

#### **Key Features of a PWA:**

- **Offline Availability:** Allows users to access the app without an internet connection.
- **Fast Performance:** Uses caching strategies to reduce loading time.
- **App-Like Interface:** Can be installed on devices and launched in a standalone mode.
- **Secure & Reliable:** Served over HTTPS to prevent data tampering.
- **Background Sync & Push Notifications:** Keeps users engaged with real-time updates.

## Steps to Make Your Web Application a PWA

### Create a Web App Manifest File

- Go to the root or `public` folder of your project.
  - Create a new file named `manifest.json`.
  - Add metadata such as the app name, icons, theme color, and display mode.
  - This file helps browsers recognize your app as installable and enables the Add to Home Screen feature.
- 

### Link the Manifest in HTML

- Open your `index.html` file.
  - Add a `<link>` tag to reference the `manifest.json` file inside the `<head>` section.
  - Also, include a `<meta>` tag for the theme color.
  - This allows the browser to detect and use the manifest.
- 

### Register a Service Worker

- Create a new file named `service-worker.js` in your project root.
  - Add code to cache static assets and handle fetch events.
  - Service workers enable offline functionality and caching strategies.
- 

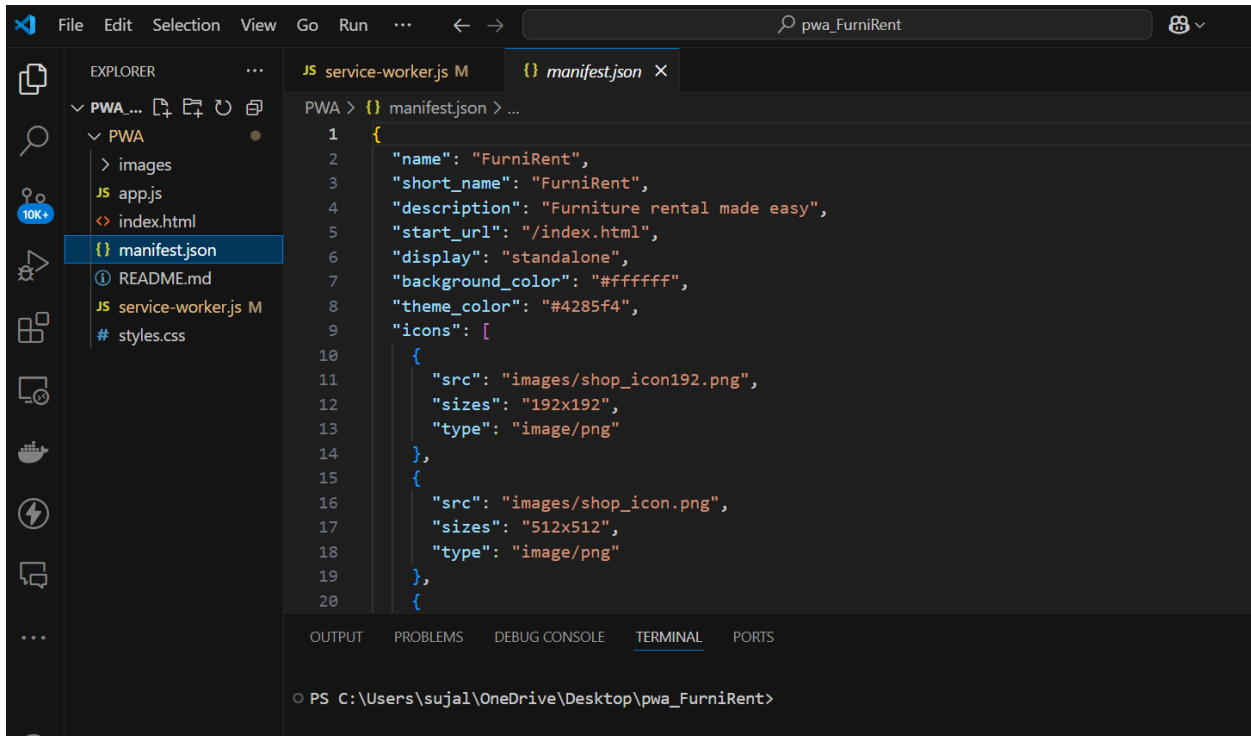
### Add Service Worker Registration Script

- Open your main JavaScript file (e.g., `main.js` or inside a `<script>` tag in `index.html`).
  - Add the logic to register the service worker when the page loads.
  - This ensures the service worker gets activated and starts handling caching.
-

## Serve Over HTTPS

- Host your web app using HTTPS to ensure security.
- Platforms like Netlify, Vercel, or GitHub Pages provide free HTTPS support.
- HTTPS is required for service workers and installability features.

## OUTPUT :

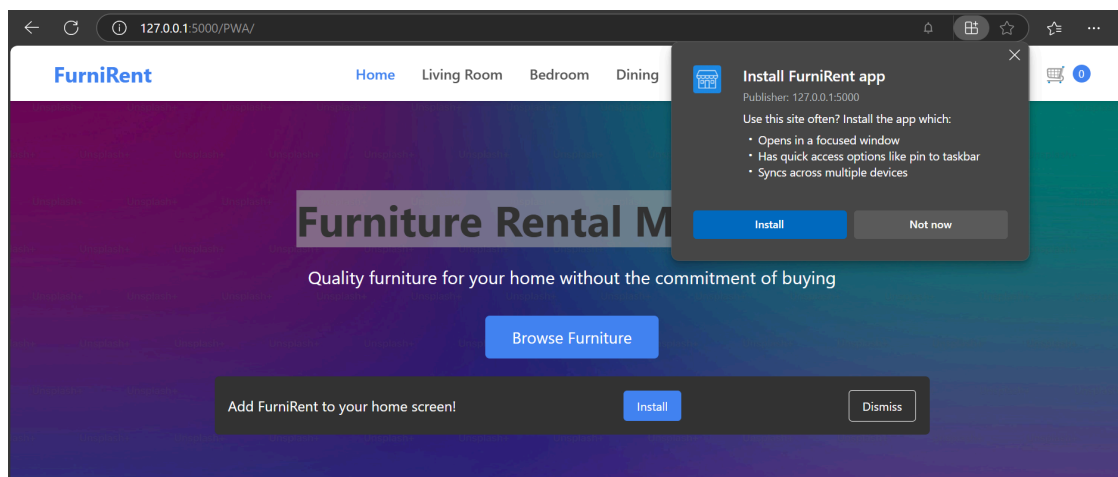
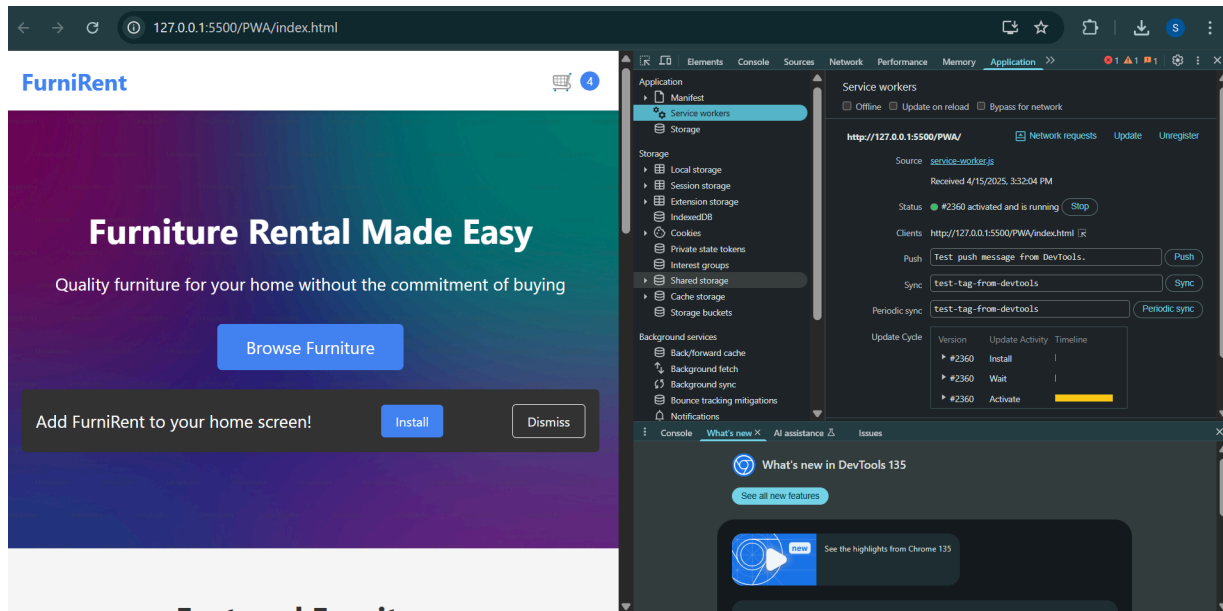


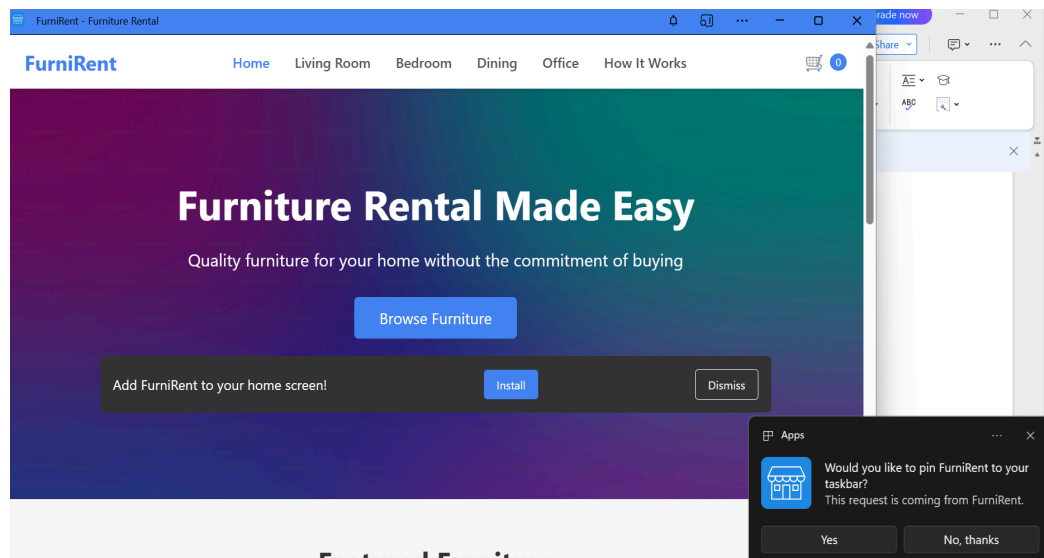
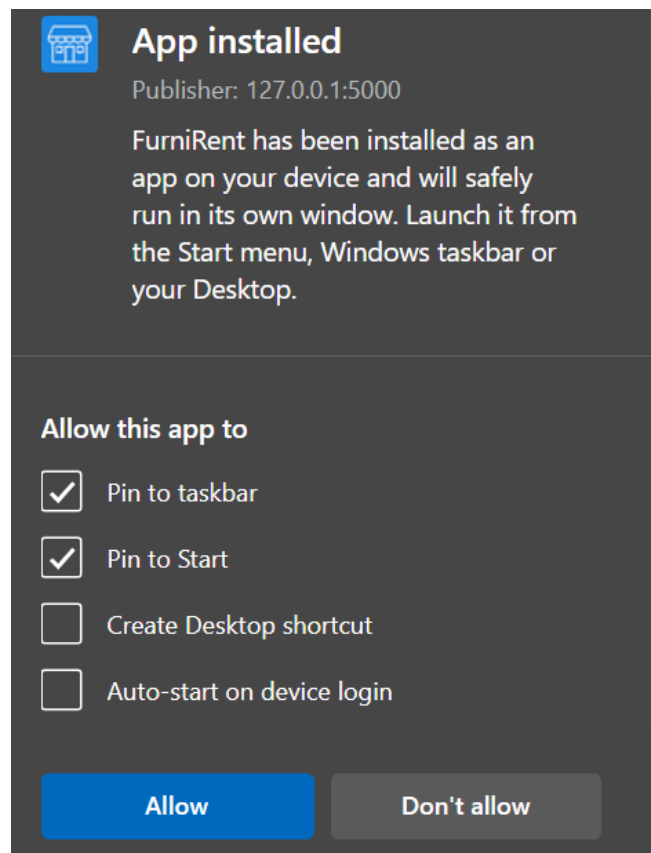
The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left, the main editor in the center, and the Output/Debug Console at the bottom. The Explorer sidebar shows a project named 'PWA\_FurniRent' with a file structure including 'images', 'app.js', 'index.html', 'manifest.json', 'README.md', 'service-worker.js', and 'styles.css'. The 'manifest.json' file is selected and its content is displayed in the main editor. The content is a JSON object defining the PWA's metadata and icons. The Output/Debug Console at the bottom shows the command prompt path: 'PS C:\Users\sujal\OneDrive\Desktop\pwa\_FurniRent>'.

```
1 {
2   "name": "FurniRent",
3   "short_name": "FurniRent",
4   "description": "Furniture rental made easy",
5   "start_url": "/index.html",
6   "display": "standalone",
7   "background_color": "#ffffff",
8   "theme_color": "#4285f4",
9   "icons": [
10    {
11      "src": "images/shop_icon192.png",
12      "sizes": "192x192",
13      "type": "image/png"
14    },
15    {
16      "src": "images/shop_icon.png",
17      "sizes": "512x512",
18      "type": "image/png"
19    },
20    {
```

OUTPUT PROBLEMS DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\sujal\OneDrive\Desktop\pwa\_FurniRent>





## Conclusion:

By following these steps, we successfully converted our Vite React application into a Progressive Web App. This enables offline accessibility, caching for better performance, and an installable app experience on mobile and desktop devices. Implementing PWA features enhances user engagement and makes the app function seamlessly even in limited network conditions.

