**Name: Bhavesh Wadhwani**                    **Class: D15A**                    **Roll no:69**

# Experiment - 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
Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

2. Ease of Access
Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

3. Faster Services
PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the

brands improve the user engagement and retention rate, which eventually adds value to their business.

4. Engaging Approach
As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand. In simpler words, PWAs let you maintain the user engagement and retention rate.

5. Updated Real-Time Data Access
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

The main features are:
Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.

Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

App-like — They behave with the user as if they were native apps, in terms of interaction and navigation.

Updated — Information is always up-to-date thanks to the data update process offered by service workers.

Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.

Searchable — They are identified as "applications" and are indexed by search engines.

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.

Code:
```
import { useEffect, useState } from 'react';
import './App.css';
import { v4 as uuidv4 } from 'uuid';
import Header from './components/Header';
import { ProjectList } from './components/ProjectList';
```

```jsx
import { LogsTable } from './components/LogsTable';

function App() {
const [ projects, setProjects] =
useState(JSON.parse(localStorage.getItem('projects')) || []);
const [ isSidebarVisible, setIsSidebarVisible] = useState(false);
projects.map(project=> console.log(project.state))

useEffect(() => {
localStorage.setItem('projects', JSON.stringify(projects));
},[projects]);

const handleSidebarClose = () => setIsSidebarVisible(false);
const handleSidebarShow = () => setIsSidebarVisible(true);

const handleAddProject = (project) => {
  const projectToBeAdded = {...project, id: uuidv4(), elapsed:0,
runningSince:null,logs:[{Starttime:0,Endtime:0,Description:""}]};
  setProjects([...projects, projectToBeAdded]);
  setIsSidebarVisible(false);
}

const handleUpdateProject = (projectId, updatedProject) => {
  const updatedProjects = projects.map(project => {
    if(project.id === projectId) {
      return{
        ...updatedProject,
        elapsed: project.elapsed,
        runningSince: project.runningSince,
        id: projectId
      }
    }
    return project;
  });
  setProjects(updatedProjects);
}

const handleDeleteProject = (projectId) => {
  const updatedProjects = projects.filter(project => project.id !== projectId);
  setProjects(updatedProjects);
```

```jsx
}

const handleStartTimer = (projectId) => {
  const updatedProjects = projects.map(project => project.id === projectId ?
{...project, runningSince: Date.now()}: project);
  setProjects(updatedProjects);
}
const handleStopTimer = (projectId) => {
  const updatedProjects = projects.map(project => {
    if(project.id !== projectId) return project;

    return {
      ...project,
      elapsed:project.elapsed + (Date.now() - project.runningSince),
      runningSince:null
    }
  });
  setProjects(updatedProjects);
}
  return (
    <>
      <Header
        onAddProject = { handleAddProject }
        onSidebarShow = { handleSidebarShow}
        onSidebarClose = { handleSidebarClose}
        isSidebarVisible = { isSidebarVisible }
      />
      <ProjectList
        onDeleteProject = { handleDeleteProject }
        projects={ projects }
        onSidebarShow= { handleSidebarShow }
        onStartTimer ={ handleStartTimer }
        onStopTimer = { handleStopTimer }
        onUpdateProject={ handleUpdateProject }
      />
      <LogsTable
        projects={projects}
      />
    </>
  );
```
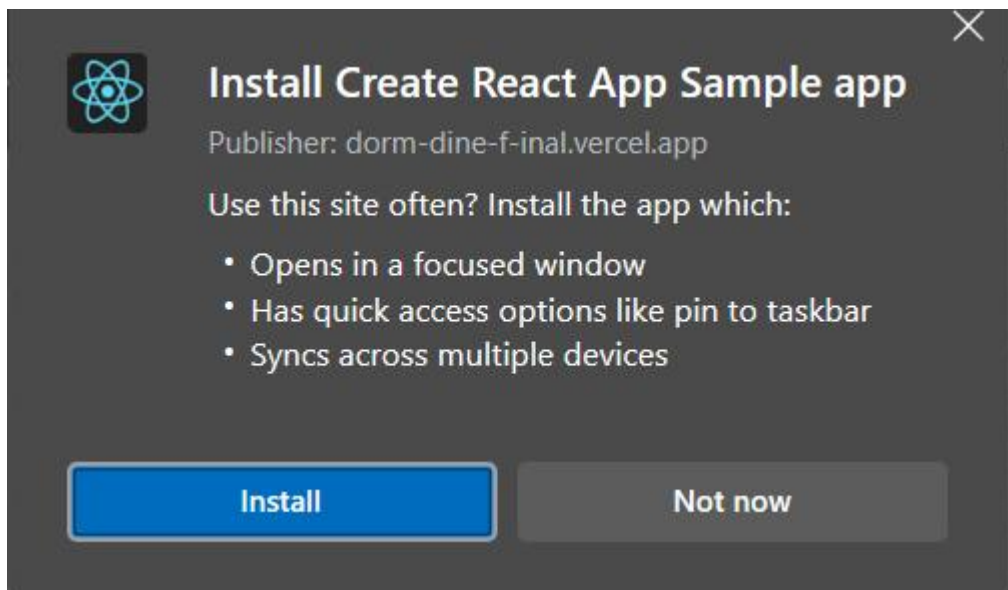
}

export default App;

Open folder in VS code and click go live at bottom right corner

Open your hosted site on Microsoft Edge
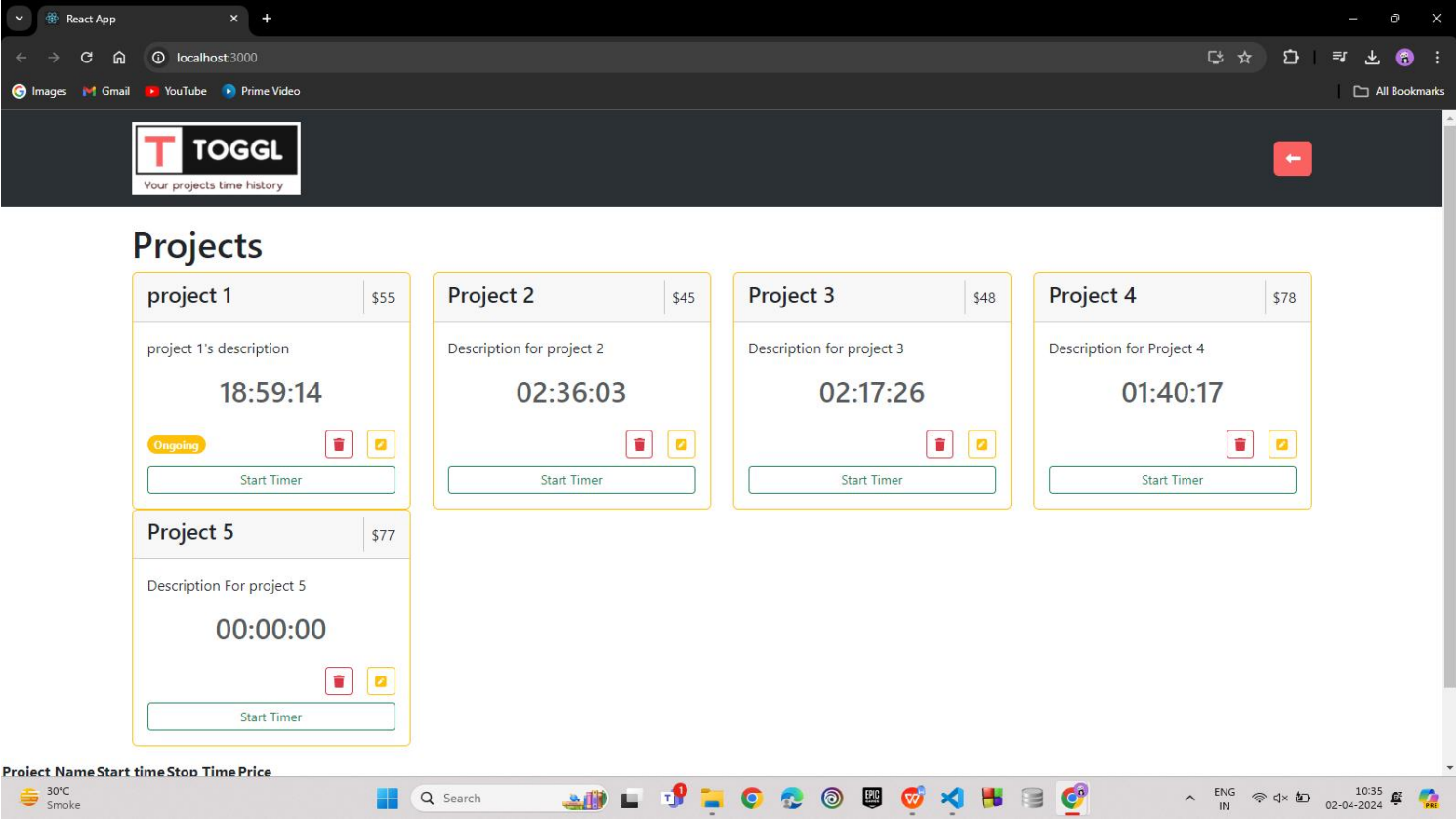
Click on 3 dots
click on Apps
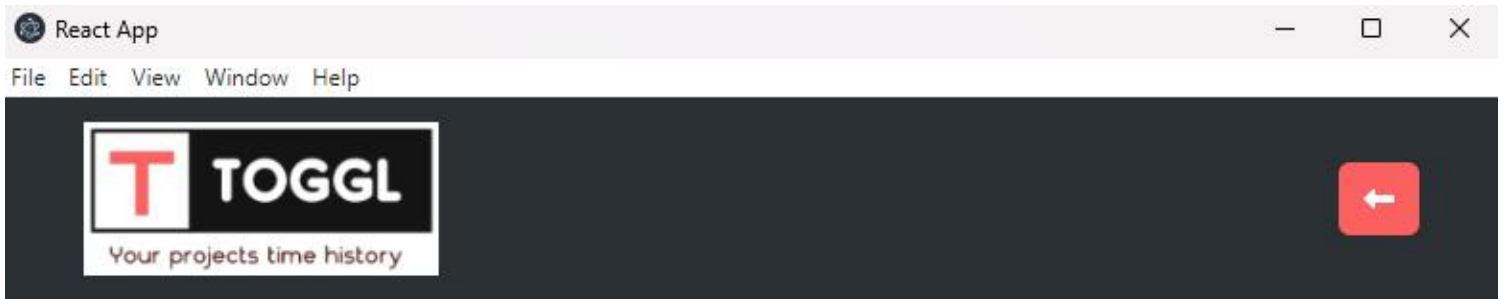select install app option
Click on Install



App icon will appear at the bottom

Output:
   Desktop App Created Successfully.



Open Desktop App:

## Conclusion:

In this experiment, we have successfully created a basic progressive web app of our web page and installed it in our desktop successfully.