Roll No:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY
ACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA lab

COURSE CODE	ITL604
EXPERIMENT NO.	07
EXPERIMENT TITLE	To create a responsive user interface using Jquery Mobile/ Material UI / Angular UI /React UI for your application
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	08/03/2022
SUBMISSION DATE	15/03/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION	PRESENTATION	UNDERSTANDING	TOTAL MARKS
4	4	7	15
NAME& SIGN. OF FACULTY	Prof Dr. S. M. Patil		

Aim: To create a responsive user interface using bootstrap

Theory:

What is a responsive website?

A responsive website means that the site can be opened on multiple screen sizes that are relevant to all kind of devices that we use daily. The ability to showcase your website to a wide variety of people with diverse devices means improved user experience.

How to Build a Responsive Bootstrap Website?

Awesome! With all the background knowledge equipped, we are now ready to go with the tutorial itself. Let's begin.

Note: You can use a simple text editor and a browser to get the website ready. However, if you want to explore how it will function as a real website, then we recommend you to check out free VPS such as Amazon Web Services, and get your website online. This way you can check its behavior on different systems or browsers.

Basic Setup

Step 1: Viewport and initial scale

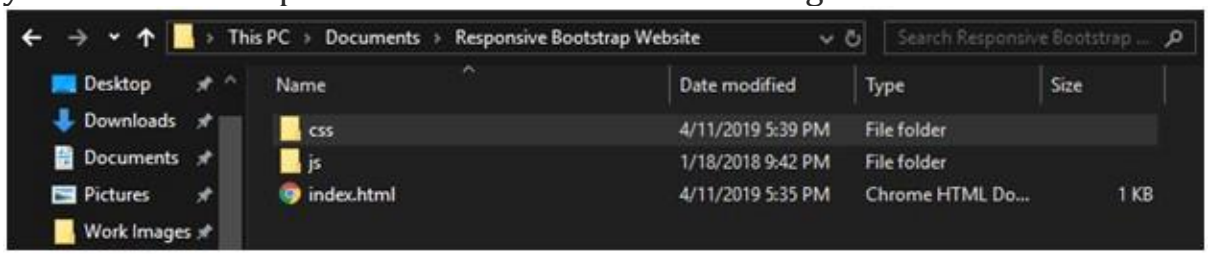
The first step is to set up a responsive Bootstrap properly. It can be done by simply putting the code in your web pages.

```
<meta name=”viewport” content=”width=device-width, initial-scale=1.0">
```

The above code defines a meta tag that tells the browser to set the width of the website according to the device width. It also set scaling to 1 which equates to default website. In short, we tell the browser that we are going to build a responsive website.

Step 2: Hooking up Bootstrap

Now, it is time to link Bootstrap libraries. It can be done by simply using the code below. To make the code work, you also need to download Bootstrap to your development folder and extract it. It will create two folders within your main development folder as shown in the image below.



Now use the following code and put it in the index.html file below the viewpoint meta tag.

Starting the Building Process

Depending on what hosting solution you are using, whether it be Amazon S3, a dedicated or shared hosting plan, or even through your own in-house hosting, as long as you are using these same tools and commands, you should be fine in setting this process up. A responsive website can be divided into five major components. As we are building a responsive website, the main components are as follows.

1. Navigation
2. Information area
3. Contents section
4. Right Sidebar
5. Footer.

Step 1: Getting the navigation up

The navigation stays at the top of the website. Apart from that, it can also contain other vital information such as the website name. To ensure that it works, you need to use the “navbar” class as shown in the code below.

As you can see, have also used other classes such as fixed-top, navbar-light, bg-light and so on. These control the text nature, its background and so on.

Now, let’s add more code.

code defines a container div which we can use it to store and manipulate the navigation content.

Most of the above code is self-explanatory. We create branding and menu items using predefined classes. We also highlighted the Home menu item so that people know on which page they are currently browsing. Other than that we also used collapse navbar-collapse. It makes menu stack when viewed in smaller screens.

Step 2: Getting the Information area up

Putting the required information on your website is easy thanks to the jumbotron class. To make sure we keep things organized, we will be using the div tag.

Let's look at the code below to get a better understanding.

The other two key classes here is the display-2 and btn btn-primary btn-lg classes. They create big content on the website center.

Step 3: Getting the content ready

It's now time to create the content section for the responsive website. To ensure that we make this section responsive as well, we are going to use the col-md-8 mb-4, col-md-4 mb-4 and so on. As you can see, we are using the Flexbox-based bootstrap grid system.

The col-md-* is the basis of the whole structure. It is a two-column layout which will be triggered once the website finds a medium sized screen. Similarly, col-xs-* will be triggered if a smaller screen is encountered. The choice of the classes is all upon you as you need to decide which screen sizes to target.

The only thing that you need to keep in mind is that the total column supported by bootstrap is 12. This means that you need to only put the classes that sum up to 12. If not done correctly, the code will not perform as intended. Furthermore, you can also use the row class within the columns and nest them using a div element. Let's take a look at the code below to get a proper understanding.

Step 4: Getting the Right Sidebar Ready

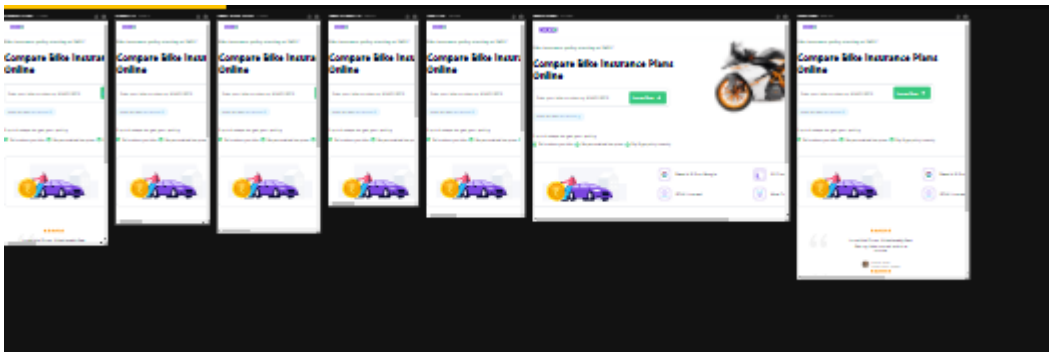
Awesome, we have built a significant part of the website. Now, it is time to get the right sidebar ready. In our website's case, we will be listing the vertical navigation list on the sidebar. To do it, all you need to do so is use the code below.

Step 5: Footer

The last step is to get the footer ready. Footer provides business information or any other copyright information. To build the booter, simply use the code below.

Result: -

Responsive viewer extension of chrome browser results



Conclusion: For making progressive web application , it is very important to make our existing website responsive because every device has different size and for better UI experience we should make website responsive using different library available over internet.

Roll No:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY
ACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA

COURSE CODE	ITL604
EXPERIMENT NO.	08
EXPERIMENT TITLE	To write meta data of your application PWA in a Web App manifest file to enable" Add to Home screen features
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	15/03/2022
SUBMISSION DATE	29/03/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION		PRESENTATION		UNDERSTANDING		TOTAL MARKS	
	4		4		7		15
NAME& SIGN. OF FACULTY		Prof Dr. S. M. Patil					

Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to home screen feature”.

Theory:

The web app manifest is a JSON file that tells the browser about your Progressive Web App and how it should behave when installed on the user's desktop or mobile device. A typical manifest file includes the app name, the icons the app should use, and the URL that should be opened when the app is launched.

Manifest files are supported in Chrome, Edge, Firefox, UC Browser, Opera, and the Samsung browser. Safari has partial support.

Create the manifest file

The manifest file can have any name, but is commonly named manifest.json and served from the root (your website's top-level directory). The specification suggests the extension should be .webmanifest, but browsers also support .json extensions, which may be easier for developers to understand.

manifest.json > [ ] icons > {} 3

```
{
    "name": "GO Insaurance",
    "short_name": "GO Insaurance",
    "icons": [
      {
        "src": "/static/favicon/acko-logo-48x48.png",
        "sizes": "48x48",
        "type": "image/png",
        "density": 1.0
      },
      {
        "src": "/static/favicon/acko-logo-96x96.png",
        "sizes": "96x96",
        "type": "image/png",
        "density": 2.0
      },
      {
        "src": "/static/favicon/acko-logo-192x192.png",
        "sizes": "192x192",
        "type": "image/png",
        "density": 3.0
      },
      {
        "src": "/static/favicon/acko-logo-256x256.png",
        "sizes": "256x256",
        "type": "image/png",
        "density": 4.0
      },
      {
        "src": "/static/favicon/acko-logo-512x512.png",
        "sizes": "512x512",
        "type": "image/png",
        "density": 5.0
      }
    ],
    "start_url": "/",
    "display": "standalone",
    "orientation": "portrait",
```

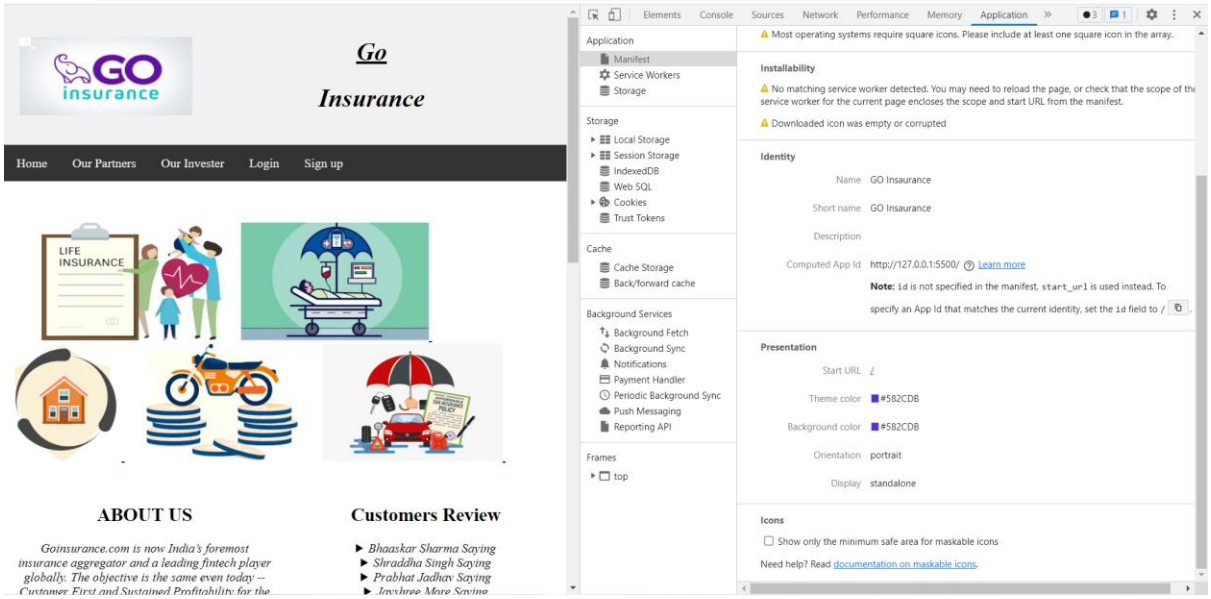
This is a fairly straight forward file, describing our application, its colors and icons. We also define which URL (relative to the URL the app is loaded from) should be opened if the application is installed to a device.

Save the file, and move over to index.html. Just before </head>, insert the following link:

<link rel="manifest" href="./manifest.json">

his will load the manifest file and identify our site as an application to the browser.

Verify that everything worked by opening Chrome DevTools (⌘-⌥-I or Ctrl-Shift-I) and selecting the Application tab. Under the application tab, select the Manifest section and you should see the information we just entered in our manifest file.



Conclusion:- manifest is the json file tells the browser about your Progressive Web App and how it should behave when installed on the user's desktop or mobile device. It is recommended to add more pixels sizes images.

Roll NO:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGYACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA lab

COURSE CODE	ITL604
EXPERIMENT NO.	09
EXPERIMENT TITLE	To code and register a service worker, and complete the install and activation process for a new service worker for the E- commerce PWA.
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	29/03/2022
SUBMISSION DATE	05/04/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION		PRESENTATION		UNDERSTANDING		TOTAL MARKS	
	4		4		7		15
NAME& SIGN. OF FACULTY		Prof Dr. S. M. Patil					

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E- commerce PWA.

Theory:

The Web App Manifest identifies our website as an app to the browser. The ServiceWorker is the last piece of the puzzle, the piece that will take our plain, old, website and give it superpowers like loading offline.

The ServiceWorker is a JavaScript worker that sits between your application and the network. With it (and some supporting APIs like Cache API) we're able to have full control over how our application behaves in any network situation. But before we get ahead of ourselves, let's create an empty ServiceWorker and register it.

In the root of your application, create a file sw.js. You can leave it empty for now.

Next, we need to register our ServiceWorker so that our browser can install it. Go into app.js, and add the following line in the load event listener.

App.js

```
window.addEventListener('load', e => {
  new PWAConfApp();
  registerSW();
});
```

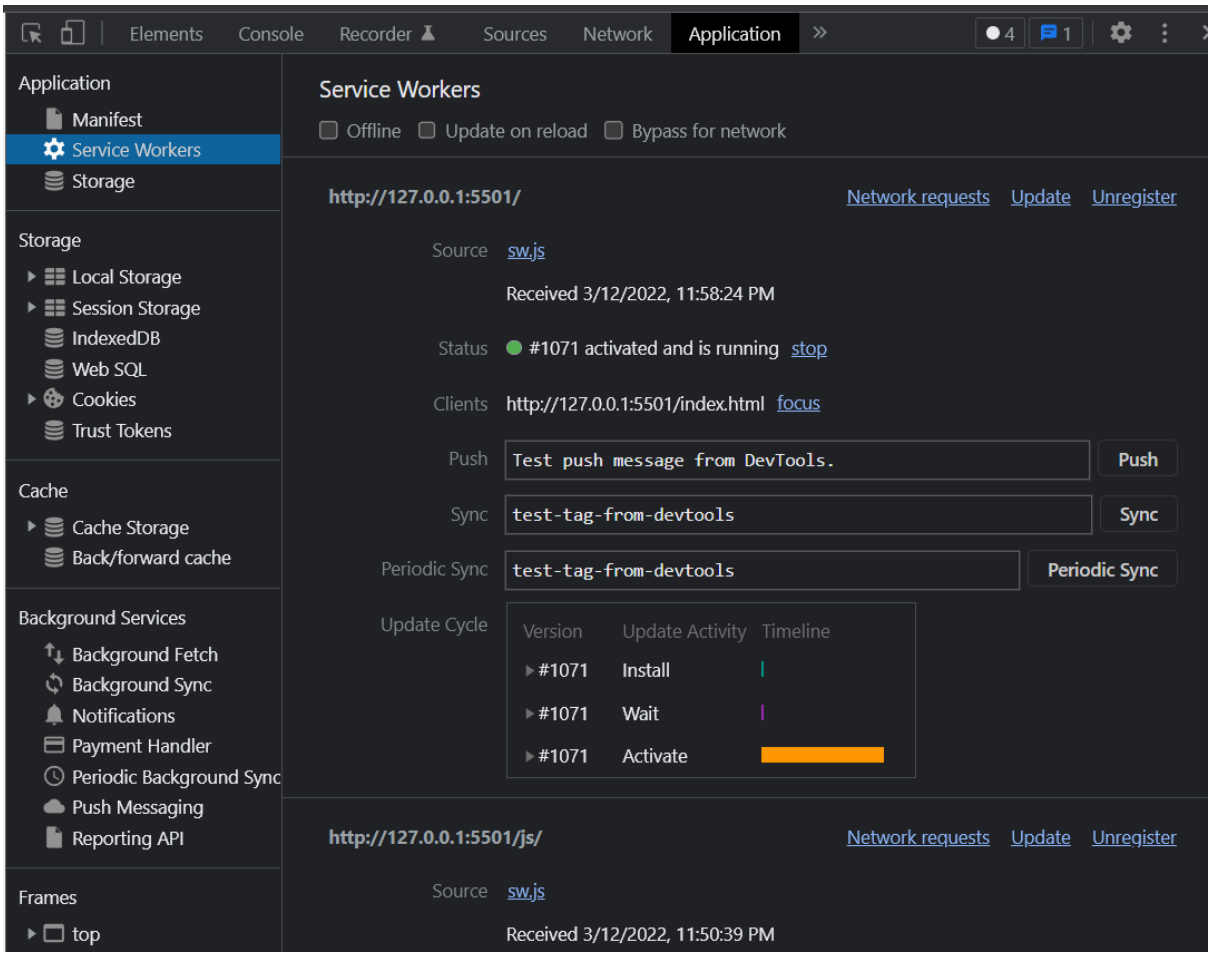
Before we register the ServiceWorker, we want to ensure that the browser supports it. Add the following method after the load even listener:

App.js

```
async function registerSW() {
  if ('serviceWorker' in navigator) {
    try {
      await navigator.serviceWorker.register('./sw.js');
    } catch (e) {
      alert('ServiceWorker registration failed. Sorry about
that.');
    }
```

1. We're using an async function for registering the ServiceWorker for easier to read code.
2. Ensure that the browser supports ServiceWorker before trying to register one.
3. Register the ServiceWorker with navigator.serviceWorker.register.
4. Let the user know if things failed.
5. Show an unintrusive notice that offline is not supported if the browser doesn't support ServiceWorker.

Reload your browser and switch to the *ServiceWorker* section of the *Application* tab in DevTools. You should see your service worker listed. To make our lives easier while developing, check "Update on reload"**.** Normally, a new ServiceWorker is only loaded once all tabs using it have been closed, so we would not get our new version visible by just refreshing the page while developing otherwise.



Conclusion: Service workers are event driven. Both the installation and activation processes trigger corresponding install and activate events to which the service workers can respond.

Roll No:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGYACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA lab

COURSE CODE	ITL604
EXPERIMENT NO.	10
EXPERIMENT TITLE	To implement Service worker events like fetch, sync and push for E-commerce PWA.
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	05/04/2022
SUBMISSION DATE	12/04/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION	PRESENTATION	UNDERSTANDING	TOTAL MARKS
4	4	7	15
NAME& SIGN. OF FACULTY	Prof Dr. S. M. Patil		

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA

Theory:

What is fetch?

The Fetch API is a simple interface for fetching resources. Fetch makes it easier to make web requests and handle responses than with the older XMLHttpRequest, which often requires additional logic (for example, for handling redirects).

Fetch is supported across all modern browsers, but there is a polyfill if you need to support older browsers.

The fetch() method takes the path to a resource as input. The method returns a promise that resolves to the Response of that request.

Making a request

Let's look at a simple example of fetching a JSON file:

We pass the path for the resource we want to retrieve as a parameter to fetch. In this example, this is **examples/example.json**. The fetch call returns a promise that resolves to a response object.

When the promise resolves, the response is passed to .then. This is where the response could be used. If the request does not complete, .catch takes over and is passed the corresponding error.

Note, the previous example uses promise chaining, however async/await can simplify your code. The remaining examples use async/await or top-level await.

Here is the same example as before, but converted to use top-level await.

Caching and serving static assets

The ServiceWorker file is entirely event-driven. This means that it won't run any code unless it's responding to an event. The main events we are interested in are install and fetch. We can listen to them by adding the following lines to our sw.js file. In a ServiceWorker context, self refers to the ServiceWorker itself.

sw.js

```
self.addEventListener("install", async (event) => {
  console.log("install event");
});

self.addEventListener("fetch", async (event) => {
  console.log("fetch event");
});
```

The install event is called whenever a new ServiceWorker file is detected.

sw.js

```
const cacheName = "pwa-conf-v1";
const staticAssets = [
"./",
"./index.html",
 "./js/app.js",
 "./css/styles.css"
];
```

Notice the use of relative paths for the static assets. This is needed especially if you are on a host like GitHub Pages, where your app isn't deployed to the root of the domain. Also note that we've added paths for both ./ and ./index.html. The Cache API is very literal, we need to explicitly cache both even though they result in the same file on the server.

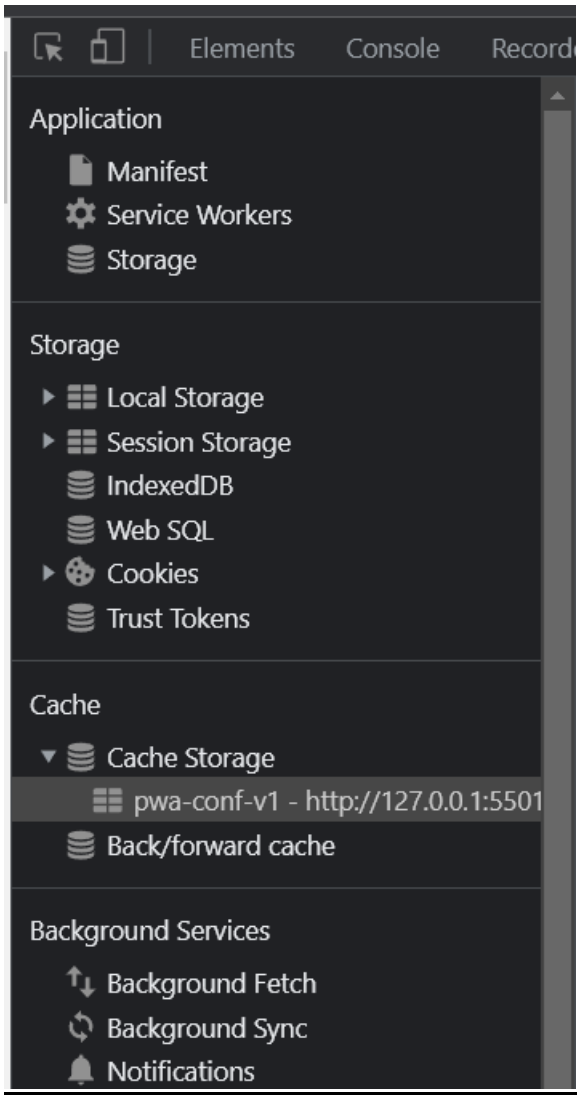
With our assets sorted out, we can now setup our cache:

sw.js

```
self.addEventListener("install", async (event) => {
  const cache = await caches.open(cacheName);
  await cache.addAll(staticAssets);
});
```

1. Open or create a new cache with the name we specified.
2. Tell the cache to fetch and add all the static asse

Reload the browser and open the Cache Storage section of the Application tab in DevTools to verify that everything worked as expected.



Conclusion: fetch, sync and push events are very important for making PWA offline by using cache storage.

Roll No:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGYACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA

COURSE CODE	ITL604
EXPERIMENT NO.	11
EXPERIMENT TITLE	To study and implement deployment of Ecommerce PWA to GitHub Pages.
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	12/04/2022
SUBMISSION DATE	19/04/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION		PRESENTATION		UNDERSTANDING		TOTAL MARKS	
	4		4		7		15
NAME& SIGN. OF FACULTY		Prof Dr. S. M. Patil					

Aim: To study and implement deployment of Ecommerce PWA to GitHub Pages

Theory:

Step1: create a new local repository in github(https://github.com/chandu-jadhav) with valid name

Step2: then go to your static website

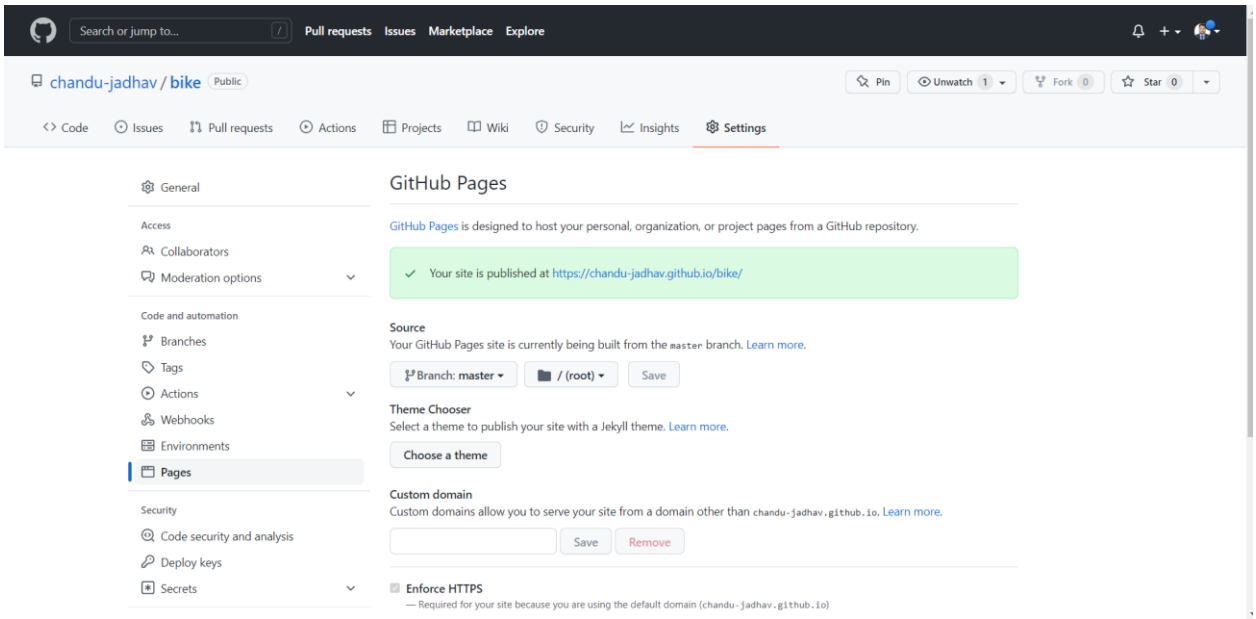
Step3: then in visual studio code inside terminal do following

1) git init
2) git add .
3) git commit -m "completed"
4) git remote add origin https://github.com/chandu-jadhav/bike.git
5) git push -u origin master

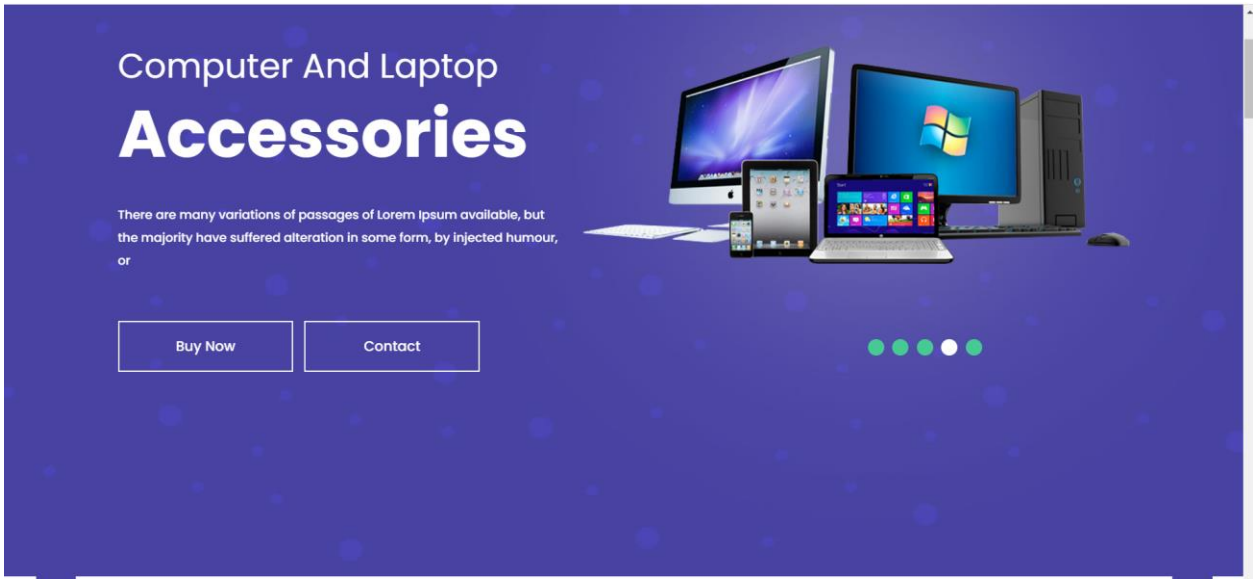
step4: once all git oprations are done then move back to your repository in github go into setting then goto pages section on left side

step5: select there master on place of none and save it

now you will see following:



Once you click on that link you will see following output:



Conclusion: Thus, we have successfully and implemented deployment of Ecommerce PWA to GitHub Pages.

Roll No:32

BHARATI VIDYAPEETH
COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGYACADEMIC YEAR: 2021-2022

COURSE NAME: MAD & PWA lab

COURSE CODE	ITL604
EXPERIMENT NO.	12
EXPERIMENT TITLE	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
NAME OF STUDENT	Chandrakant Anil Jadhav
ROLL NO.	32
CLASS	TE-IT
SEMESTER	VI
GIVEN DATE	19/04/2022
SUBMISSION DATE	26/04/2022
CORRECTION DATE	
REMARK	

TIMELY SUBMISSION	PRESENTATION	UNDERSTANDING	TOTAL MARKS
4	4	7	15
NAME& SIGN. OF FACULTY	Prof Dr. S. M. Patil		

Aim: To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

Theory:

How do I tell if all of my Progressive Web App (PWA) features are in order? Lighthouse is an open-source tool from Google that audits a web app for PWA features. It provides a set of metrics to help guide you in building a PWA with a full application-like experience for your users.

Lighthouse tests if your app:

- Can load in offline or flaky network conditions
- Is relatively fast
- Is served from a secure origin
- Uses certain accessibility best practices

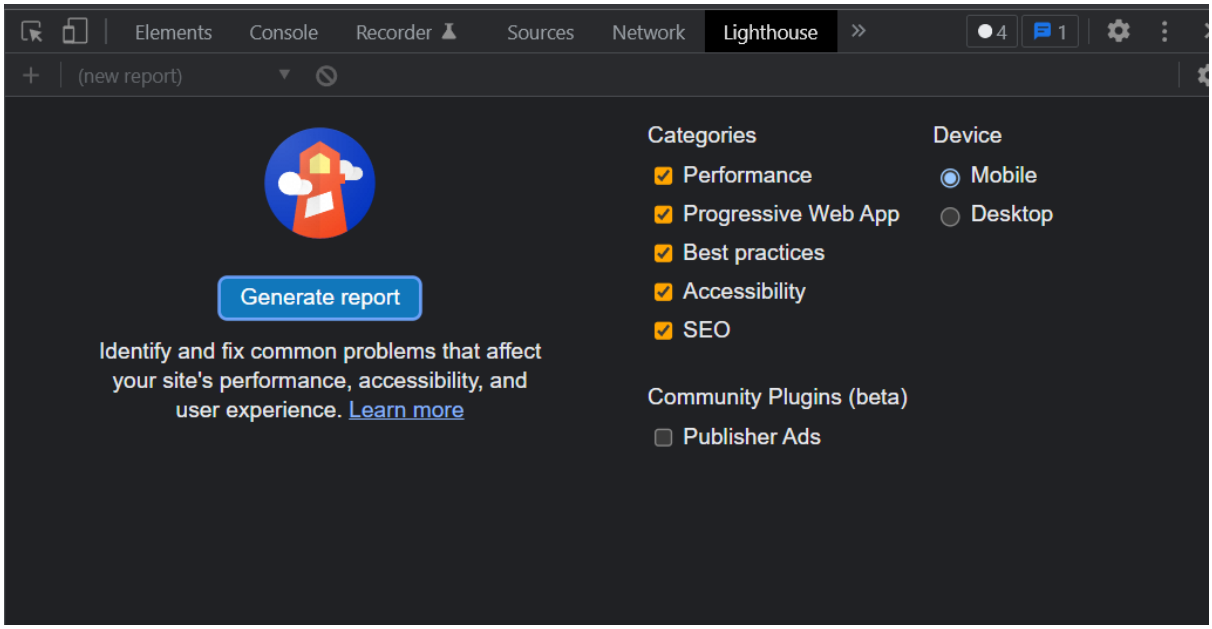
Lighthouse is available as a Chrome extension for Chrome 52 (and later) and a command line tool.

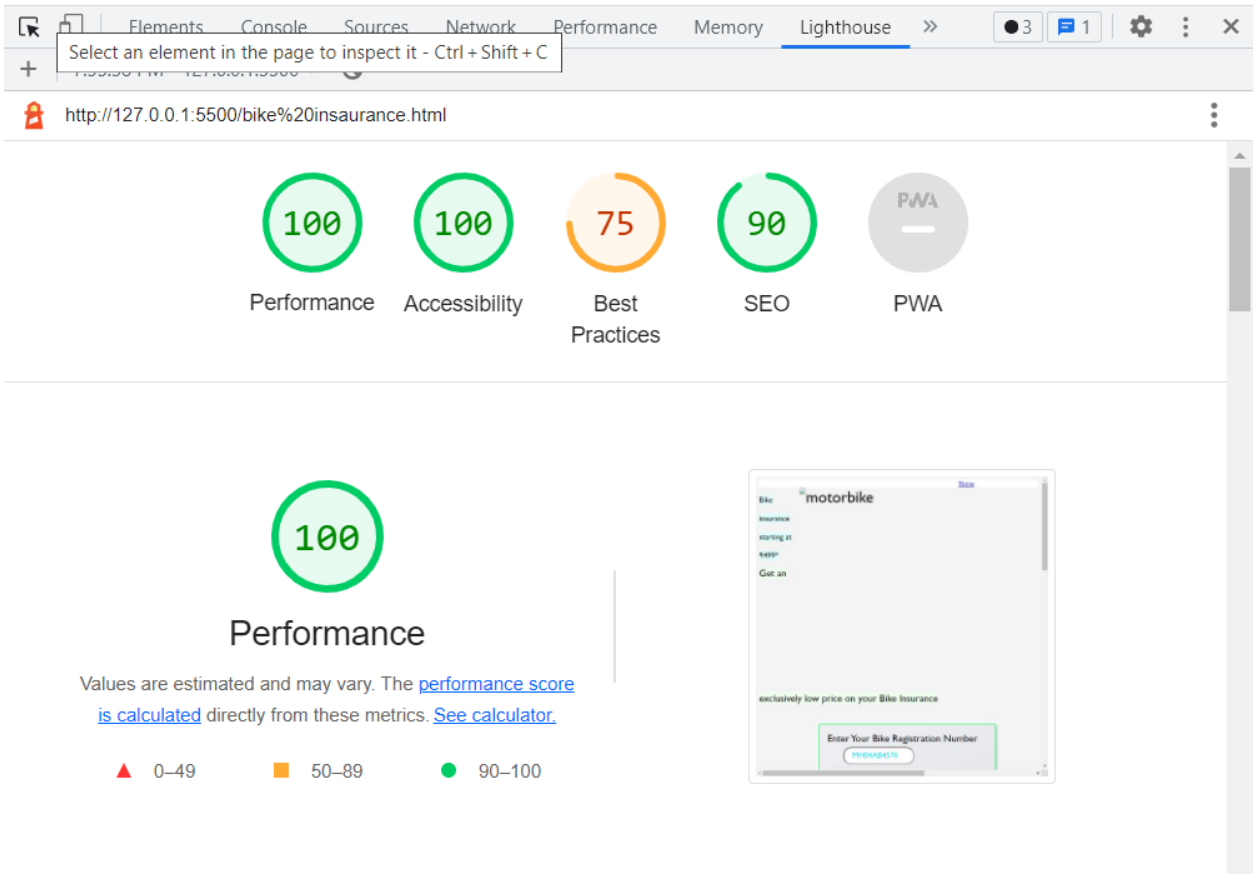
Running Lighthouse as a Chrome extension

Download the Lighthouse Chrome extension from the Chrome Web Store.

When installed it places an icon in your taskbar.

Run Lighthouse on your application by selecting the icon and choosing **Generate report** (with your app open in the browser page).





Conclusion: Thus, we have successfully tested this website using google Lighthouse PWA Analysis Tool and verified that this is PWA and working as expected.