CAPSTONE PROJECT -05

Objective:
The objective of this practical is to demonstrate the process of containerizing a web application using Docker.

Tools Used:

1. Docker: Containerization platform for building, sharing, and running containerized applications.
2. Text Editor (VS Code): For writing Dockerfile and editing website files.

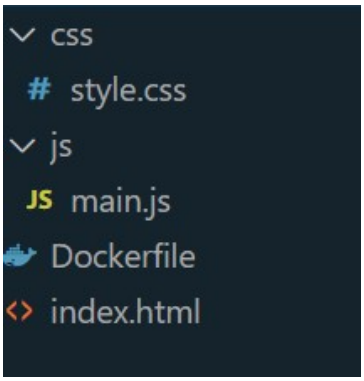
Modules:

1. Introduction to Docker:
    - Overview of containerization and its benefits.
    - Introduction to Docker and its role in modern software development.
    - Understanding Docker images, containers, and Dockerfile.
2. Setting Up Development Environment:
    - Installation of Docker Desktop or Docker Engine depending on the operating system.
    - Basic configuration and verification of Docker installation.
3. Creating a Dockerfile:
    - Understanding the structure and syntax of Dockerfile.
    - Defining the base image and required dependencies.
    - Configuring environment variables, working directory, and exposed ports.
    - Copying application files into the container.
4. Building Docker Images:
    - Building Docker images using the docker build command.
    - Monitoring the build process and understanding build output.
    - Tagging and naming Docker images for better organization.
5. Running Docker Containers:
    - Running Docker containers from built images using the docker run command.
    - Mapping container ports to host ports for accessing the application.

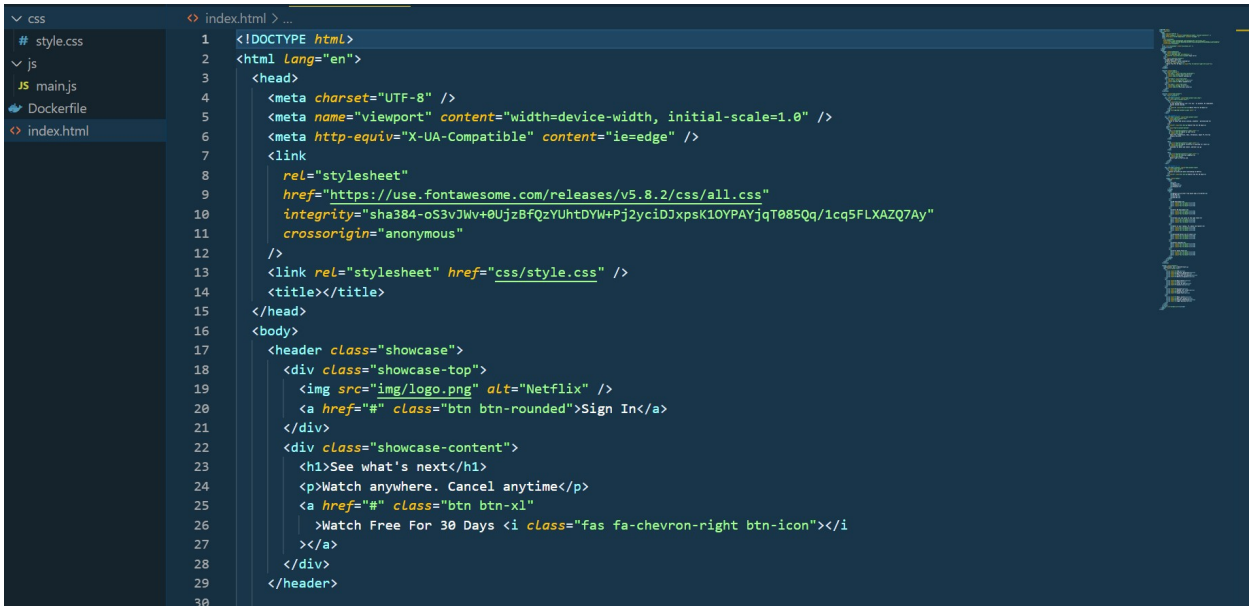
- Managing container lifecycle, including starting, stopping, and removingcontainers.

Result:

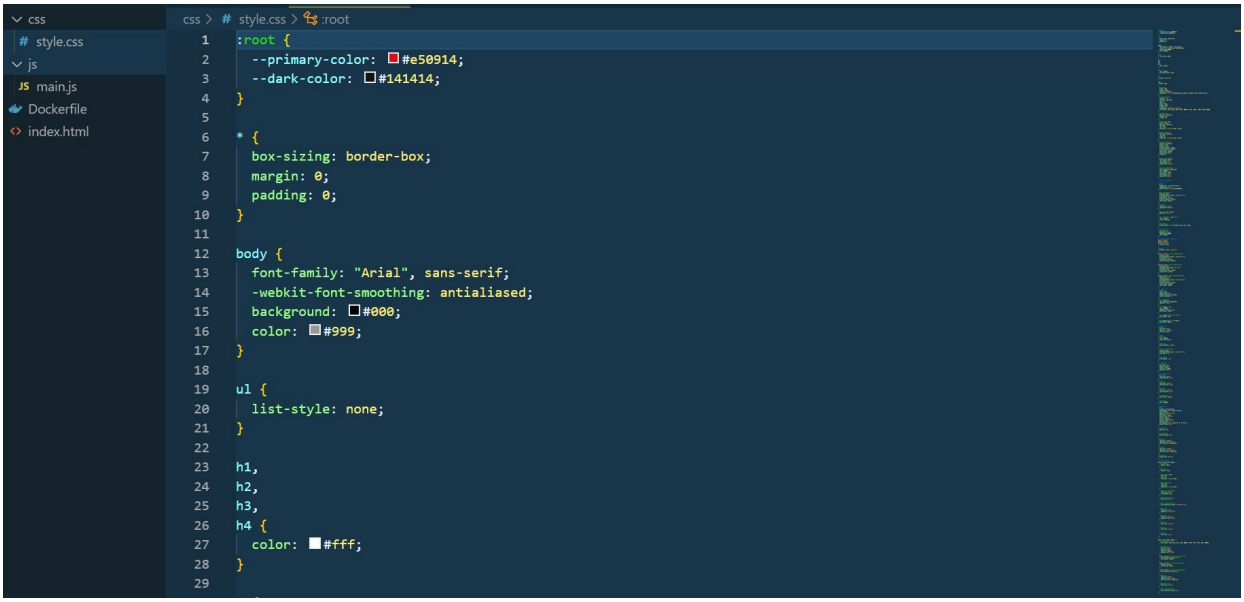
Netflix-master folder structure:



Index.html file-



Style.css file-



```
:root {
  --primary-color: #e50914;
  --dark-color: #141414;
}

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: "Arial", sans-serif;
  -webkit-font-smoothing: antialiased;
  background: #000;
  color: #999;
}

ul {
  list-style: none;
}

h1,
h2,
h3,
h4 {
  color: #fff;
}
```

main.js file-

```
const tabItems = document.querySelectorAll(".tab-item");
const tabContentItems = document.querySelectorAll(".tab-content-item");

// Select tab content
function selectItem(e) {
  removeBorder();
  removeShow();
  // Add border to current tab
  this.classList.add("tab-border");
  // Grab content item from DOM
  const tabContentItem = document.querySelector(`#${this.id}-content`);
  // Add show class
  tabContentItem.classList.add("show");
}

function removeBorder() {
  tabItems.forEach(item => item.classList.remove("tab-border"));
}

function removeShow() {
  tabContentItems.forEach(item => item.classList.remove("show"));
}

// Listen for tab click
tabItems.forEach(item => item.addEventListener("click", selectItem));
```

We will define the Dockerfile to build the Docker image:

Dockerfile

```
# Use the official Nginx image as the base image
FROM nginx:latest

# Copy HTML, CSS, and JavaScript files to NGINX default HTML directory
COPY index.html /usr/share/nginx/html/
COPY css/style.css /usr/share/nginx/html/css/
COPY js/main.js /usr/share/nginx/html/js/
```

Building the Docker Image:

- Open a terminal and navigate to the "netflix-master" directory.

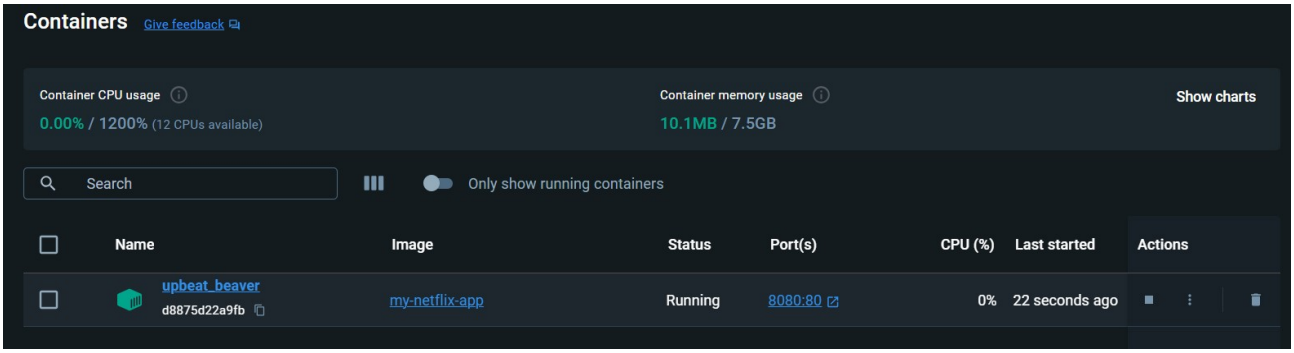
```
PS C:\Users\sushant kaddu\Desktop\netflix-master> docker build -t my-netflix-app .
>>
[+] Building 0.0s (0/0)  docker:default
[+] Building 1.5s (9/9) FINISHED                                                docker:default
 => [internal] load build definition from Dockerfile                                      0.0s
 => => transferring dockerfile: 312B                                                      0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                           1.1s
 => [internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                           0.0s
 => [1/4] FROM docker.io/library/nginx:latest@sha256:ed6d2c43c8fbcd3eaa44c9dab6d94cb346234476230dc1681227aa72d07181ee  0.0s
 => [internal] load build context                                                         0.0s
 => => transferring context: 140B                                                         0.0s
 => CACHED [2/4] COPY index.html /usr/share/nginx/html/                                   0.0s
 => CACHED [3/4] COPY css/style.css /usr/share/nginx/html/css/                            0.0s
 => CACHED [4/4] COPY js/main.js /usr/share/nginx/html/js/                                0.0s
 => exporting to image                                                                    0.0s
 => => exporting layers                                                                   0.0s
 => => writing image sha256:4770bb1f8904407b3841643e76a8ba47fdc9bbd91e0ccd0a637f62b7cc4b5e73  0.0s
 => => naming to docker.io/library/my-netflix-app                                         0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Deploying the Container:

- Once the image is built successfully, deploy a container

```
What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\sushant kaddu\Desktop\netflix-master> docker run -d -p 8080:80 my-netflix-app
>>
d8875d22a9fb0be3c76ef8a4d81d3222cc40cc5f705ecb73d2026c210594016a
PS C:\Users\sushant kaddu\Desktop\netflix-master>
```

Container inside the Docker application:



On localhost:8080