

Assignment 1

Dashboard.html

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Ecommerce Website</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-sRl4kxILFvY47J16cr9ZwB07vP4J8+LH7qKQnuqkulAvNWLeN8tE5YBujZqJLB" crossorigin="anonymous">
        <link rel="stylesheet" href="dashboard.css">
</head>

<body>
    <nav class="navbar navbar-expand-lg bg-body-tertiary">
        <div class="container-fluid">
            <a class="navbar-brand" href="#" style="background-color: black; padding-left: 20px; padding-right: 20px;">Rushi's Shop</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Admin Panel</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Manage</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">About</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Contact</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

```

Assignment 1

Dashboard.html

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

```
<form class="d-flex" role="search">
    <input class="form-control me-2" type="search" placeholder="Search" aria-
label="Search" />
    <button class="btn btn-outline-success" type="submit">Search</button>
</form>
</div>
</div>
</nav>

<div class="card-container">
    <div class="card">
        
        <div class="card-body">
            <h5 class="card-title">Acer Super-Slim 144Hz Display Screen</h5>
            <p class="card-text">Some quick example text to build on the card title and make up
            the bulk of the
                card's content.</p>
            <a href="#" class="btn btn-primary">Buy Now</a>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <h5 class="card-title">Helix Ultra-Eye-Saver FHD Display Screen</h5>
            <p class="card-text">Some quick example text to build on the card title and make up
            the bulk of the
                card's content.</p>
            <a href="#" class="btn btn-primary">Buy Now</a>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <h5 class="card-title">Acer FreeSync 100Hz Super Glare Display Screen</h5>
            <p class="card-text">Some quick example text to build on the card title and make up
            the bulk of the
                card's content.</p>
            <a href="#" class="btn btn-primary">Buy Now</a>
        </div>
    </div>
</div>
```

Assignment 1

Dashboard.html

```
</div>  
</body>  
  
</html>
```

Dashboard.css

```
body {  
    background-color: #f8f9fa;  
}  
  
.navbar-brand {  
    color: white !important;  
    font-weight: bold;  
}  
  
.card {  
    border: none;  
    border-radius: 10px;  
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
    transition: transform 0.3s ease, box-shadow 0.3s ease;  
}  
  
.card:hover {  
    transform: translateY(-5px);  
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.15);  
}  
  
.card-img-top {  
    height: 200px;  
    object-fit: contain;  
    padding: 1rem;  
}  
  
.card-container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));  
    gap: 20px;  
    padding: 20px;  
    max-width: 1200px;  
    margin: 0 auto;  
}
```

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

Assignment 1

Dashboard.css

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

```
@media (max-width: 576px) {  
    .card-container {  
        grid-template-columns: 1fr;  
    }  
  
    .card-img-top {  
        height: 180px;  
    }  
}  
  
.nav-item:hover {  
    background-color: black;  
    border-radius: 5px;  
}
```

Assignment 1

Output

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

Mobile View:

Rushi's Shop



Acer Super-Slim 144Hz Display Screen

Some quick example text to build on the card title and make up the bulk of the card's content.

[Buy Now](#)



Helix Ultra-Eye-Saver FHD Display Screen

Some quick example text to build on the card title and make up the bulk of the card's content.

[Buy Now](#)



Assignment 1

Output

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

Desktop View:

Rushi's Shop

Home Admin Panel Manage Products About Contact Search Search



Acer Super-Slim 144Hz Display Screen

Some quick example text to build on the card title and make up the bulk of the card's content.

[Buy Now](#)



Helix Ultra-Eye-Saver FHD Display Screen

Some quick example text to build on the card title and make up the bulk of the card's content.

[Buy Now](#)



Acer FreeSync 100Hz Super Glare Display Screen

Some quick example text to build on the card title and make up the bulk of the card's content.

[Buy Now](#)

Assignment 2

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

index.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>User Registration</title>
    <link rel="stylesheet" href="styles.css" />
</head>

<body>
    <main class="container">
        <h1>User Registration</h1>

        <form id="regForm">
            <div class="row">
                <label for="firstName">First name</label>
                <input id="firstName" name="firstName" type="text" required />
            </div>

            <div class="row">
                <label for="lastName">Last name</label>
                <input id="lastName" name="lastName" type="text" required />
            </div>

            <div class="row">
                <label for="email">Email</label>
                <input id="email" name="email" type="email" required />
            </div>

            <div class="row">
                <label for="password">Password</label>
                <input id="password" name="password" type="password" minlength="6" required />
            </div>

            <div class="row">
                <label for="phone">Phone</label>
                <input id="phone" name="phone" type="tel" />
            </div>

            <div class="row">
                <label for="gender">Gender</label>
                <select id="gender" name="gender">
                    <option value="">-- Select --</option>
                    <option>Male</option>
                    <option>Female</option>
                    <option>Other</option>
                </select>
            </div>
        </form>
    </main>
</body>
```

Assignment 2

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

index.html

```
</select>
</div>

<div class="actions">
    <button type="submit">Register (Save locally)</button>
    <button type="button" id="viewList">View Registrations</button>
</div>

<div id="message" aria-live="polite"></div>
</form>
</main>

<script src="script.js"></script>
</body>

</html>
```

list.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Registrations List</title>
    <link rel="stylesheet" href="styles.css" />
</head>

<body>
    <main class="container">
        <h1>Registered Users</h1>

        <div class="controls">
            <button id="clear">Clear All</button>
            <button id="export">Export JSON</button>
            <a id="backLink" href="index.html">← Back to form</a>
        </div>

        <div id="listArea"></div>
    </main>

    <script>
        function renderList() {
            const area = document.getElementById('listArea');
```

Assignment 2

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

list.html

```
const raw = localStorage.getItem('registrations');
const regs = raw ? JSON.parse(raw) : [];
if (!regs.length) {
    area.innerHTML = '<p>No registrations found.</p>';
    return;
}
let html =
'<table><thead><tr><th>#</th><th>Name</th><th>Email</th><th>Phone</th><th>Gender</th><th>Time</th></tr></thead><tbody>';
regs.forEach((r, i) => {
    html += `<tr><td>${i + 1}</td><td>${escapeHtml(r.firstName)} ${escapeHtml(r.lastName)}</td><td>${escapeHtml(r.email)}</td><td>${escapeHtml(r.phone || '')}</td><td>${escapeHtml(r.gender || '')}</td><td>${escapeHtml(r.time)}</td></tr>`;
});
html += '</tbody></table>';
area.innerHTML = html;
}

function escapeHtml(s) {
    if (!s) return '';
    return String(s).replace(/&/g, '&').replace(/</g, '<').replace(/>/g, '>').replace(/"/g, '"');
}

document.getElementById('clear').addEventListener('click', () => {
    if (!confirm('Clear all registrations from localStorage?')) return;
    localStorage.removeItem('registrations');
    renderList();
});

document.getElementById('export').addEventListener('click', () => {
    const raw = localStorage.getItem('registrations') || '[]';
    const blob = new Blob([raw], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'registrations.json';
    document.body.appendChild(a);
    a.click();
    a.remove();
    URL.revokeObjectURL(url);
});
renderList();
</script>
</body>
</html>
```

Assignment 2

Output

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

User Registration

First name

Last name

Email

Password

Phone

Gender

Register (Save locally) View Registrations

Registered Users

[Clear All](#) [Export JSON](#) [← Back to form](#)

#	Name	Email	Phone	Gender	Time
1	Rushi Katte	rushi@gmail.com	99233	Male	2025-10-28T12:25:14.799Z
2	Raj Katte	rushi@gmail.com	99233	Male	2025-10-28T12:46:59.349Z

Assignment 3A

Name: Rushikesh Katte

Roll No: 23572

Sub : Web Technology

Open the git bash and check if it is installed properly or not using git version command.

```
rushi@Admin MINGW64 ~
$ git --version
git version 2.47.0.windows.2
```

If the output comes like this then your git is installed successfully.

Create Repository using git commands –

Git Set up commands –

1. The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
$ git config --global user.name "Yuvraj Pardeshi"
$ git config --global user.email yuvrajpardeshi75@gmail.com
```

```
rushi@Admin MINGW64 ~
$ git config --global user.name "Rushikesh Katte"
rushi@Admin MINGW64 ~
$ git config --global user.email rushikeshrkatte2005@gmail.com
```

Create a new project –

2. Create an folder using command `mkdir <dir name>` and to enter into this create folder use command `cd <dir name>`

```
rushi@Admin MINGW64 ~
$ mkdir samplerepo

rushi@Admin MINGW64 ~
$ cd samplerepo

rushi@Admin MINGW64 ~/samplerepo
```

Create Repository –

3. To create your current folder into repository use command git init

```
rushi@Admin MINGW64 ~/samplerepo
$ git init
Initialized empty Git repository in C:/Users/rushi/samplerepo/.git/
```

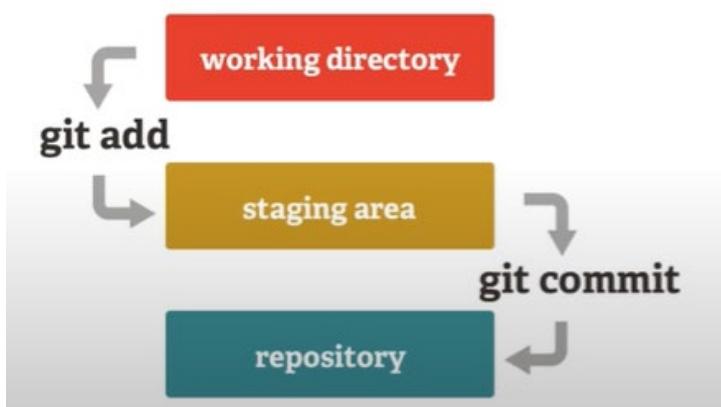
as shown in above figure your empty repo is created on the path shown in the figure.

Adding files to Repository -

4. Now here comes the concept of staging area and committing a file.

Staging area – Initially when you add files to the git folder they are not tracked by git, these files are also referred to as “untracked files”. Staging area is files that are going to be a part of next commit, which lets the git know what changes in the file are going to occur for the next commit. The command used to add the file to staging area is (git add).

Committing file - The `git commit` command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of `git commit`, The `git add` command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands “`git commit`” and “`git add`” are two of the most frequently used.



```
rushi@Admin MINGW64 ~/samplerepo (master)
$ git add index.html

rushi@Admin MINGW64 ~/samplerepo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .add.cpp.swp
```

(Added file to staging area)

```
rushi@Admin MINGW64 ~/samplerepo (master)
$ git commit -m "adding index.html"
[master (root-commit) 88bc4ea] adding index.html
 1 file changed, 8 insertions(+)
 create mode 100644 index.html
```

(Committed the file)

To check the commit status use git log command –

```
rushi@Admin MINGW64 ~/samplerepo (master)
$ git log
commit 88bc4eac5d08fb98bf51af6210c106ef4703cd03 (HEAD -> master)
Author: Rushikesh Katte <rushikeshrkatte2005@gmail.com>
Date:   Tue Oct 28 18:55:27 2025 +0530

  adding index.html
```

Pushing your files to GitHub –

5 . Login your github account and create a new repository.

The screenshot shows a GitHub user profile for 'Rushikeshhere'. The profile picture is a circular anime-style illustration of a character with blonde hair and a white coat. Below the profile picture, the user's name is listed as 'Rushikesh Katte' and their pronouns as 'he/him'. The user has 39 repositories, 7 packages, and 7 stars. A search bar at the top says 'Type / to search'. Below the search bar, there are filters for 'Type', 'Language', 'Sort', and a prominent green 'New' button, which is highlighted with a red box. Three repository cards are visible: 'complete-lms' (Private, updated 2 days ago), 'lms' (Private, updated 4 days ago), and 'FitnessTrack-master' (Public, updated 2 weeks ago). Each repository card includes a 'Star' button.

The screenshot shows the 'Create a new repository' form on GitHub. The title is 'Create a new repository'. It says 'Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#). Required fields are marked with an asterisk (*).'. There are two tabs: 'General' (selected) and 'Configuration'. In the 'General' tab, the 'Owner' is set to 'Rushikeshhere' and the 'Repository name *' is 'sample-repo', with a note that it is available. The 'Description' field contains 'sample git repo' (15 / 350 characters). In the 'Configuration' tab, the 'Choose visibility *' dropdown is set to 'Private'. At the bottom, there is a note: 'Great repository names are short and memorable. How about [probable-waddle](#)?'

After creating the repository in github, there will be the option to create your repo from scratch or if there is some local repo already created then push it. We are going to push our local repo created using git.

Pushing the local repo –

The screenshot shows the GitHub 'Create New Repository' page. At the top, there are three options: 'Set up in Desktop' (with a download icon), 'or', 'HTTPS', and 'SSH'. Below these is the URL 'git@github.com:Rushikeshishere/sample-repo.git'. A note below the URL says 'Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)'.

...or create a new repository on the command line

```
echo "# sample-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:Rushikeshishere/sample-repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:Rushikeshishere/sample-repo.git
git branch -M main
git push -u origin main
```

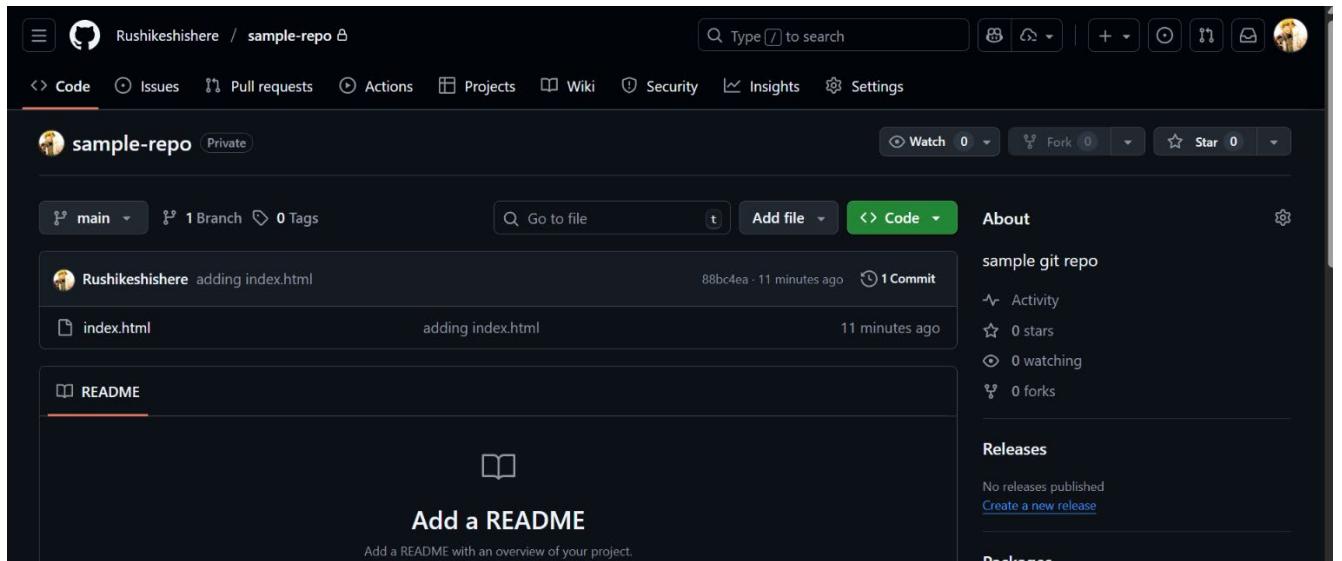
(Use the above commands to push your local repo)

```
rushi@Admin MINGW64 ~/samplerepo (main)
$ git remote remove origin

rushi@Admin MINGW64 ~/samplerepo (main)
$ git remote add origin https://github.com/Rushikeshishere/sample-repo.git

rushi@Admin MINGW64 ~/samplerepo (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 304.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Rushikeshishere/sample-repo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

rushi@Admin MINGW64 ~/samplerepo (main)
$
```



Hence we have create your repo on github and added file to the repo successfully

Assignment 3B

Name: Rushikesh Katte

Roll Number: 23572

Subject: Web Technology

1. Installing the docker

<https://docs.docker.com/desktop/install/windows-install/>

The screenshot shows the Docker Desktop setup page on the Docker documentation website. The left sidebar has a navigation menu with sections like AI and Docker Compose, PRODUCTS (Docker Desktop, Setup, Install, Mac, Mac permission requirements, Windows, Windows permission requirements, Linux, VM or VDI environments, Sign in, Allowlist, Explore Docker Desktop, Features and capabilities, Settings and maintenance). The main content area has a note about commercial use requiring a paid subscription. Below that, it says the page provides download links, system requirements, and step-by-step installation instructions for Docker Desktop on Windows. It lists three download options: 'Docker Desktop for Windows - x86_64' (button), 'Docker Desktop for Windows - x86_64 on the Microsoft Store' (button), and 'Docker Desktop for Windows - Arm (Early Access)' (button). At the bottom, there's a 'System requirements' section and a 'Table of contents' sidebar with links to System requirements, WSL: Verification and setup, Option 1: Install or update WSL via the terminal, Option 2: Install WSL via the MSI package, Install Docker Desktop on Windows, Install interactively, Install from the command line, Start Docker Desktop, and Where to go next.

Run the executable file.

2. Install WSL distribution from :

<https://learn.microsoft.com/en-us/windows/wsl/install-manual>

```
PS C:\Users\rushi> wsl --install
Downloading: Ubuntu
Installing: Ubuntu
Distribution successfully installed. It can be launched via 'wsl.exe -d Ubuntu'
Launching Ubuntu...
Provisioning the new WSL instance Ubuntu
This might take a while...
Create a default Unix user account: rushi
New password:
Retype new password:
passwd: password updated successfully
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

rushi@Admin:/mnt/c/Users/rushi$ |
```

3. Restart the system.

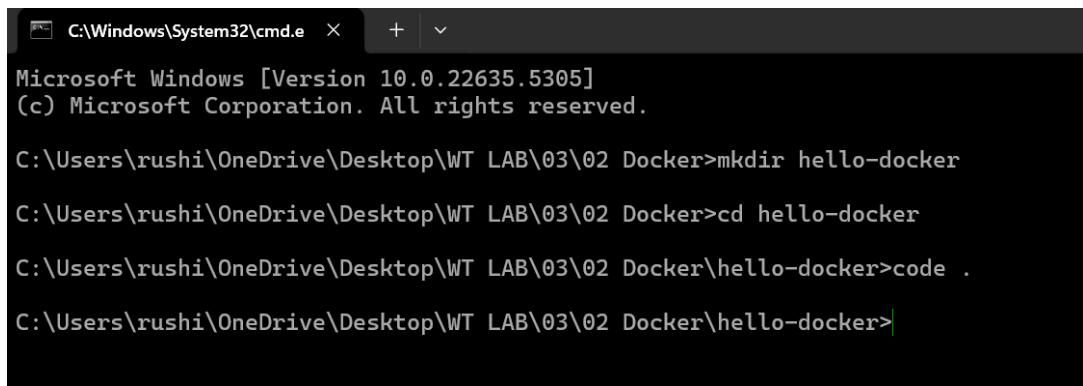
4. On command prompt run the command docker version. It should look like :

```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32>docker version
Client:
  Version:          28.5.1
  API version:     1.51
  Go version:      go1.24.8
  Git commit:      e180ab8
  Built:           Wed Oct  8 12:19:16 2025
  OS/Arch:         windows/amd64
  Context:         desktop-linux

Server: Docker Desktop 4.49.0 (208700)
  Engine:
    Version:          28.5.1
    API version:     1.51 (minimum version 1.24)
    Go version:      go1.24.8
    Git commit:      f8215cc
    Built:           Wed Oct  8 12:17:24 2025
    OS/Arch:         linux/amd64
    Experimental:   false
  containerd:
    Version:          1.7.27
    GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
  runc:
    Version:          1.2.5
    GitCommit:        v1.2.5-0-g59923ef
  docker-init:
    Version:          0.19.0
    GitCommit:        de40ad0

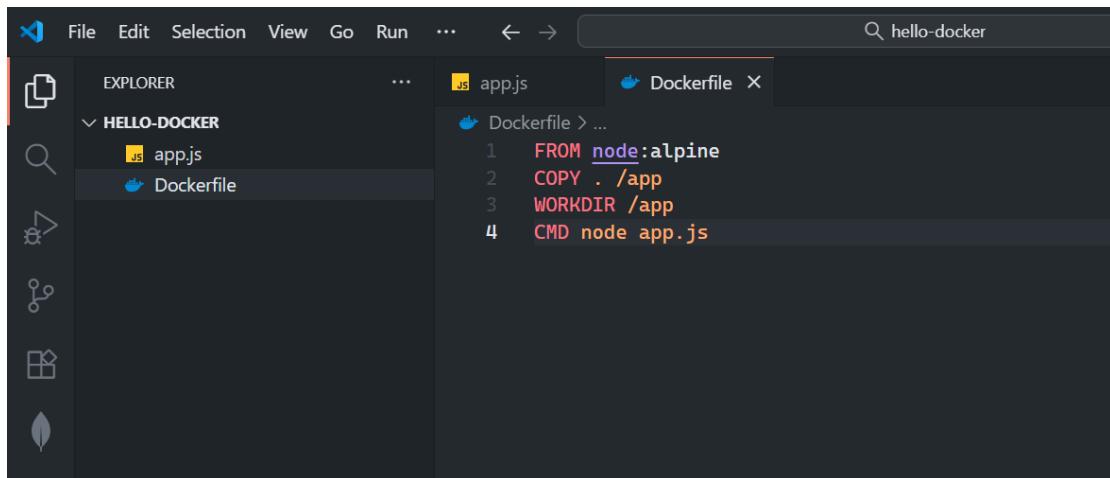
C:\Windows\System32>
```

5. Run the following commands :



```
C:\Windows\System32\cmd.e × + ▾  
Microsoft Windows [Version 10.0.22635.5305]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker>mkdir hello-docker  
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker>cd hello-docker  
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker\hello-docker>code .  
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker\hello-docker>
```

6. The Visual Studio Code will open. Inside VS code open folder docker-hello and create file app.js and Dockerfile:



7. Inside app.js file type the code:

```
console.log("Hello docker");
```

8. Inside Dockerfile type the code :

```
FROM node:alpine
```

```
COPY . /app
```

```
WORKDIR /app
```

```
CMD node app.js
```

9. In the command prompt type the command :

docker build -t hello-docker .

```
Microsoft Windows [Version 10.0.22635.5305]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker\hello-docker>docker build -t hello-docker .
[+] Building 46.6s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 97B
=> [internal] load metadata for docker.io/library/node:alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 162B
=> [1/3] FROM docker.io/library/node:alpine@sha256:bce79c648e05d584ad9ae2b45ed663ae6f07ebfa9e5fe6f5b7f0165aca0
=> => resolve docker.io/library/node:alpine@sha256:bce79c648e05d584ad9ae2b45ed663ae6f07ebfa9e5fe6f5b7f0165aca0
=> => sha256:d9f0123823b504c41373c028b427fcf56cf338c454b4f291afc415de5e070464 443B / 443B 0.0s
=> => sha256:4350616372df2de9f6b15ad5b0bfc771662215b9192defaeb7d94a96978bb0ec 1.26MB / 1.26MB 0.0s
=> => sha256:6d0bdc936f33e7a2e7faae4e44192a62c33e1c9f1fe253e3f046afb31737eeb5 54.99MB / 54.99MB 37.1s
=> => sha256:2d35ebdb57d9971fea0cac1582aa78935adf8058b2cc32db163c98822e5dfa1b 3.80MB / 3.80MB 0.0s
=> => extracting sha256:2d35ebdb57d9971fea0cac1582aa78935adf8058b2cc32db163c98822e5dfa1b 0.7s
=> => extracting sha256:6d0bdc936f33e7a2e7faae4e44192a62c33e1c9f1fe253e3f046afb31737eeb5 1.26MB / 1.26MB 0.0s
=> => extracting sha256:4350616372df2de9f6b15ad5b0bfc771662215b9192defaeb7d94a96978bb0ec 0.0s
=> => extracting sha256:d9f0123823b504c41373c028b427fcf56cf338c454b4f4291afc415de5e070464 0.0s
=> [2/3] COPY . /app
=> [3/3] WORKDIR /app
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:b1d40101763fe38cc935cf6e91e414c67c59c0398bb26eal52cc43fe957d0e15
=> => exporting config sha256:a2435c2e6ed7428e29a42dbbc4234f63338870f636f3a9fde0e2f83146868dea
=> => exporting attestation manifest sha256:a7887ec84e67f656af30545853bd555cfe61204aedf19105598aae8588906805
=> => exporting manifest list sha256:c078fc60516878292fcc0f31bdhfae95881fa8cb81d9b01fa5515c23101e82a7 0.0s
```

10. Then type the command :

docker image ls

11. You should get the output image :

```
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\02 Docker\hello-docker>docker image ls
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-docker    latest   c078fc605168   8 seconds ago  242MB
```

Assignment 3C

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

Firstly, lets install Node.js from <https://nodejs.org/>

The screenshot shows the official Node.js website at nodejs.org/en/. The page features a dark green hexagonal background pattern. At the top, there's a navigation bar with links for Classroom, Roadmaps, Practice Projects, UI Design, icons, Flowbite, Learn, About, Download, Blog, Docs, Contribute, Certification, a search bar with placeholder 'Start typing...', and keyboard shortcut 'Ctrl + K'. Below the navigation is a large heading 'Run JavaScript Everywhere'. A brief description follows: 'Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.' Two buttons are present: 'Get Node.js®' and 'Get security support for EOL Node.js versions'. On the right side, there's a code editor window titled 'Create an HTTP Server' showing a simple Node.js script to run a server on port 3000 and respond with 'Hello World!'. Other tabs in the code editor include 'Write Tests', 'Read and Hash a File', and 'Streams Pipeline'.

Follow the steps to complete the download.

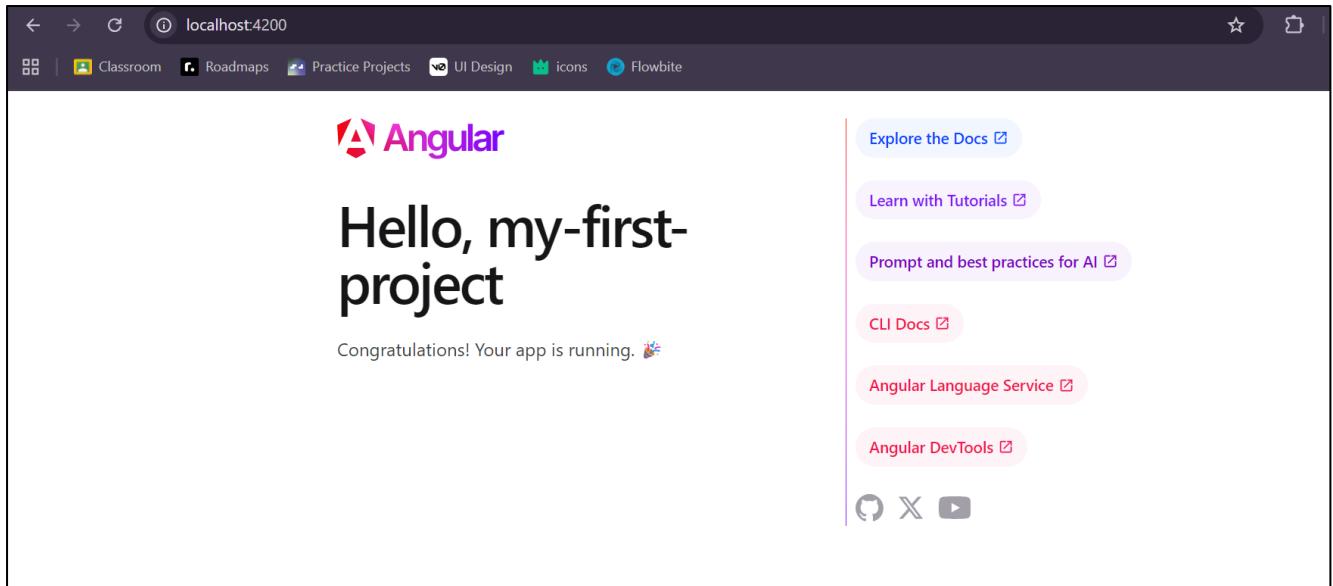
1. Create Angular App

```
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\03 Angular\my-first-project>ng new user-auth-app
✓ Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ]
  ]
✓ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
✓ Do you want to create a 'zoneless' application without zone.js? Yes
✓ Which AI tools do you want to configure with Angular best practices? https://angular.dev/ai/develop-with-ai
None
CREATE user-auth-app/angular.json (2337 bytes)
CREATE user-auth-app/package.json (1110 bytes)
CREATE user-auth-app/README.md (1533 bytes)
CREATE user-auth-app/tsconfig.json (1026 bytes)
CREATE user-auth-app/.editorconfig (331 bytes)
CREATE user-auth-app/.gitignore (647 bytes)
CREATE user-auth-app/tsconfig.app.json (444 bytes)
CREATE user-auth-app/tsconfig.spec.json (422 bytes)
CREATE user-auth-app/.vscode/extensions.json (134 bytes)
CREATE user-auth-app/.vscode/launch.json (490 bytes)
CREATE user-auth-app/.vscode/tasks.json (980 bytes)
CREATE user-auth-app/src/main.ts (228 bytes)
CREATE user-auth-app/src/index.html (310 bytes)
CREATE user-auth-app/src/styles.css (81 bytes)
CREATE user-auth-app/src/app/app.spec.ts (812 bytes)
CREATE user-auth-app/src/app/app.ts (307 bytes)
CREATE user-auth-app/src/app/app.css (0 bytes)
CREATE user-auth-app/src/app/app.html (20464 bytes)
CREATE user-auth-app/src/app/app.config.ts (395 bytes)
CREATE user-auth-app/src/app/app.routes.ts (80 bytes)
CREATE user-auth-app/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
  Successfully initialized git.
```

```
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\03 Angular\my-first-project>cd user-auth-app
C:\Users\rushi\OneDrive\Desktop\WT LAB\03\03 Angular\my-first-project\user-auth-app>ng serve -o
Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

  No
Global setting: enabled
Local setting: disabled
Effective status: disabled
Initial chunk files | Names           | Raw size
main.js              | main            | 48.48 kB
styles.css           | styles          | 95 bytes
                           | Initial total | 48.57 kB

Application bundle generation complete. [2.308 seconds] - 2025-10-28T16:38:18.195Z
Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
  → Local: http://localhost:4200/
  → press h + enter to show help
```



Angular User Authentication Application

Project Overview

This is a comprehensive **User Authentication Application** built with **Angular 20** featuring modern Angular capabilities including standalone components, signals, and control flow syntax. The application provides complete user authentication functionality with registration, login, and profile management.

Project Architecture

Technology Stack

- **Frontend Framework**: Angular 20.3.0
- **Language**: TypeScript 5.9.2
- **Styling**: CSS3 with modern features
- **State Management**: Angular Signals
- **Data Persistence**: Browser localStorage
- **Build Tool**: Angular CLI 20.3.7

Project Structure

```

```
user-auth-app/
├── src/
│ └── app/
│ ├── app.ts # Main application component
│ ├── app.html # Application template
│ ├── app.css # Global styles
│ ├── app.config.ts # Application configuration
│ ├── app.routes.ts # Routing configuration
│ └── models/
│ └── user.model.ts # User interface definition
│ └── services/
│ └── auth.ts # Authentication service
└── login/
 ├── login/
 │ ├── login.ts # Login component
 │ ├── login.html # Login template
 │ └── login.css # Login styles
 ├── register/
 │ ├── register.ts # Registration component
 │ ├── register.html # Registration template
 │ └── register.css # Registration styles
 └── profile/
 ├── profile/
 │ ├── profile.ts # Profile component
 │ ├── profile.html # Profile template
 │ └── profile.css # Profile styles
 └── main.ts # Application bootstrap
 └── index.html # Main HTML file
 └── styles.css # Global styles
 └── package.json # Dependencies and scripts
 └── angular.json # Angular configuration
 └── tsconfig.json # TypeScript configuration
 └── tsconfig.app.json # TypeScript app configuration
```

```

📁 Complete Source Code

1. Package Configuration

```

##### `package.json` 
```json
{
 "name": "user-auth-app",
 "version": "0.0.0",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build",
 "watch": "ng build --watch --configuration development"
 },
 "private": true,
 "dependencies": {
 "@angular/common": "^20.3.0",
 "@angular/compiler": "^20.3.0",
 "@angular/core": "^20.3.0",
 "@angular/forms": "^20.3.0",
 "@angular/platform-browser": "^20.3.0",
 "@angular/router": "^20.3.0",
 "rxjs": "~7.8.0",
 }
}
```

```

```
"tslib": "^2.3.0"
},
"devDependencies": {
  "@angular/build": "^20.3.7",
  "@angular/cli": "^20.3.7",
  "@angular/compiler-cli": "^20.3.0",
  "typescript": "~5.9.2"
}
}
```
`angular.json`
```json
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "user-auth-app": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular/build:application",
          "options": {
            "browser": "src/main.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [
              {
                "glob": "**/*",
                "input": "public"
              }
            ],
            "styles": [
              "src/styles.css"
            ]
          },
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "500kB",
                  "maximumError": "1MB"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "4kB",
                  "maximumError": "8kB"
                }
              ],
              "outputHashing": "all"
            },
            "development": {
              "optimization": false,
              "vendorChunk": true,
              "sourceMap": true
            }
          }
        }
      }
    }
  }
}
```

```

    "extractLicenses": false,
    "sourceMap": true
  },
},
"defaultConfiguration": "production"
},
"serve": {
  "builder": "@angular/build:dev-server",
  "configurations": {
    "production": {
      "buildTarget": "user-auth-app:build:production"
    },
    "development": {
      "buildTarget": "user-auth-app:build:development"
    }
  },
  "defaultConfiguration": "development"
},
"extract-i18n": {
  "builder": "@angular/build:extract-i18n"
}
}
},
"cli": {
  "analytics": false
}
}
```

```

#### #### 2. Application Bootstrap

```

`src/main.ts`
```typescript
import { bootstrapApplication } from '@angular/platform-browser';
import { appConfig } from './app/app.config';
import { App } from './app/app';

bootstrapApplication(App, appConfig)
  .catch((err) => console.error(err));
```

`src/index.html`
```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>UserAuthApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

```

#### #### 3. Application Configuration

```
`src/app/app.config.ts`
```typescript  
import { AppConfig, provideBrowserGlobalErrorListeners, provideZonelessChangeDetection } from  
'@angular/core';  
import { provideRouter } from '@angular/router';  
  
import { routes } from './app.routes';  
  
export const appConfig: AppConfig = {  
  providers: [  
    provideBrowserGlobalErrorListeners(),  
    provideZonelessChangeDetection(),  
    provideRouter(routes)  
  ]  
};  
```
```

```
`src/app/app.routes.ts`
```typescript  
import { Routes } from '@angular/router';  
import { Register } from './register/register';  
import { Login } from './login/login';  
import { Profile } from './profile/profile';  
  
export const routes: Routes = [  
  { path: '', redirectTo: '/login', pathMatch: 'full' },  
  { path: 'register', component: Register },  
  { path: 'login', component: Login },  
  { path: 'profile', component: Profile },  
  { path: '**', redirectTo: '/login' }  
];  
```
```

#### #### 4. Main Application Component

```
`src/app/app.ts`
```typescript  
import { Component, signal } from '@angular/core';  
import { RouterOutlet } from '@angular/router';  
  
@Component({  
  selector: 'app-root',  
  imports: [RouterOutlet],  
  templateUrl: './app.html',  
  styleUrls: ['./app.css'  
})  
export class App {  
  protected readonly title = signal('user-auth-app');  
}  
```
```

```
`src/app/app.html`
```html  
<router-outlet></router-outlet>  
```
```

```
`src/app/app.css`
```css  
/* Empty - styles handled by individual components */  
```
```

#### ### 5. Data Models

```
`src/app/models/user.model.ts`
```typescript  
export interface User {  
    id?: number;  
    username: string;  
    email: string;  
    password?: string;  
    firstName?: string;  
    lastName?: string;  
    phone?: string;  
}  
```
```

#### ### 6. Authentication Service

```
`src/app/services/auth.ts`
```typescript  
import { Injectable, signal } from '@angular/core';  
import { User } from './models/user.model';  
  
@Injectable({  
    providedIn: 'root'  
})  
export class Auth {  
    private users: User[] = [];  
    private currentUser = signal<User | null>(null);  
    private nextId = 1;  
  
    constructor() {  
        // Load users from localStorage  
        const storedUsers = localStorage.getItem('users');  
        if (storedUsers) {  
            this.users = JSON.parse(storedUsers);  
            this.nextId = Math.max(...this.users.map(u => u.id || 0), 0) + 1;  
        }  
  
        // Load current user from localStorage  
        const storedCurrentUser = localStorage.getItem('currentUser');  
        if (storedCurrentUser) {  
            this.currentUser.set(JSON.parse(storedCurrentUser));  
        }  
    }  
  
    // Register a new user  
    register(user: User): { success: boolean; message: string } {  
        // Check if user already exists  
        const existingUser = this.users.find(  
            u => u.username === user.username || u.email === user.email  
        );
```

```
if (existingUser) {
  return {
    success: false,
    message: 'Username or email already exists!'
  };
}

// Add new user
const newUser: User = {
  ...user,
  id: this.nextId++
};

this.users.push(newUser);
this.saveUsers();

return {
  success: true,
  message: 'Registration successful! Please login.'
};
}

// Login user
login(username: string, password: string): { success: boolean; message: string } {
  const user = this.users.find(
    u => (u.username === username || u.email === username) && u.password === password
  );

  if (user) {
    // Don't store password in current user
    const { password: _, ...userWithoutPassword } = user;
    this.currentUser.set(userWithoutPassword);
    localStorage.setItem('currentUser', JSON.stringify(userWithoutPassword));

    return {
      success: true,
      message: 'Login successful!'
    };
  }
}

return {
  success: false,
  message: 'Invalid username or password!'
};

// Logout user
logout(): void {
  this.currentUser.set(null);
  localStorage.removeItem('currentUser');
}

// Get current user
getCurrentUser() {
  return this.currentUser;
}

// Check if user is logged in
```

```

isloggedIn(): boolean {
  return this.currentUser() !== null;
}

// Save users to localStorage
private saveUsers(): void {
  localStorage.setItem('users', JSON.stringify(this.users));
}
}
```

```

#### ### 7. Login Component

```

`src/app/login/login.ts`
```typescript
import { Component, signal } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { Router, RouterLink } from '@angular/router';
import { Auth } from '../services/auth';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-login',
  imports: [FormsModule, CommonModule, RouterLink],
  templateUrl: './login.html',
  styleUrls: ['./login.css'],
})
export class Login {
  username = '';
  password = '';
  message = signal('');
  messageType = signal<'success' | 'error'>('success');

  constructor(private authService: Auth, private router: Router) {
    // Redirect if already logged in
    if (this.authService.isLoggedIn()) {
      this.router.navigate(['/profile']);
    }
  }

  onSubmit(): void {
    if (!this.username || !this.password) {
      this.showMessage('Please enter username and password!', 'error');
      return;
    }

    const result = this.authService.login(this.username, this.password);

    if (result.success) {
      this.showMessage(result.message, 'success');
      setTimeout(() => {
        this.router.navigate(['/profile']);
      }, 1000);
    } else {
      this.showMessage(result.message, 'error');
    }
  }
}

```



```
.login-card {  
  background: white;  
  border-radius: 10px;  
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);  
  padding: 40px;  
  width: 100%;  
  max-width: 400px;  
}  
}
```

```
.login-card h2 {  
  text-align: center;  
  color: #333;  
  margin-bottom: 30px;  
  font-size: 28px;  
}  
}
```

```
.alert {  
  padding: 12px;  
  border-radius: 5px;  
  margin-bottom: 20px;  
  text-align: center;  
  font-weight: 500;  
}  
}
```

```
.alert-success {  
  background-color: #d4edda;  
  color: #155724;  
  border: 1px solid #c3e6cb;  
}  
}
```

```
.alert-error {  
  background-color: #f8d7da;  
  color: #721c24;  
  border: 1px solid #f5c6cb;  
}  
}
```

```
.form-group {  
  margin-bottom: 20px;  
}  
}
```

```
.form-group label {  
  display: block;  
  margin-bottom: 5px;  
  color: #555;  
  font-weight: 600;  
}  
}
```

```
.form-control {  
  width: 100%;  
  padding: 12px;  
  border: 1px solid #ddd;  
  border-radius: 5px;  
  font-size: 14px;  
  transition: border-color 0.3s;  
  box-sizing: border-box;  
}  
}
```

```
.form-control:focus {
```

```

outline: none;
border-color: #667eea;
}

.btn {
  width: 100%;
  padding: 12px;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  font-weight: 600;
  cursor: pointer;
  transition: background-color 0.3s;
}

.btn-primary {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
}

.btn-primary:hover:not(:disabled) {
  opacity: 0.9;
}

.btn-primary:disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

.register-link {
  text-align: center;
  margin-top: 20px;
  color: #666;
}

.register-link a {
  color: #667eea;
  text-decoration: none;
  font-weight: 600;
}

.register-link a:hover {
  text-decoration: underline;
}

```

```

### ### 8. Register Component

```

`src/app/register/register.ts`
```typescript
import { Component, signal } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { Router, RouterLink } from '@angular/router';
import { Auth } from '../services/auth';
import { User } from '../models/user.model';
import { CommonModule } from '@angular/common';

@Component({

```

```
selector: 'app-register',
imports: [FormsModule, CommonModule, RouterLink],
templateUrl: './register.html',
styleUrl: './register.css',
})
export class Register {
user: User = {
  username: '',
  email: '',
  password: '',
  firstName: '',
  lastName: '',
  phone: ''
};

confirmPassword = '';
message = signal('');
messageType = signal<'success' | 'error'>('success');

constructor(private authService: Auth, private router: Router) { }

onSubmit(): void {
// Validation
if (!this.user.username || !this.user.email || !this.user.password) {
  this.showMessage('Please fill in all required fields!', 'error');
  return;
}

if (this.user.password !== this.confirmPassword) {
  this.showMessage('Passwords do not match!', 'error');
  return;
}

if (this.user.password.length < 6) {
  this.showMessage('Password must be at least 6 characters long!', 'error');
  return;
}

// Register user
const result = this.authService.register(this.user);

if (result.success) {
  this.showMessage(result.message, 'success');
  setTimeout(() => {
    this.router.navigate(['/login']);
  }, 1500);
} else {
  this.showMessage(result.message, 'error');
}
}

private showMessage(msg: string, type: 'success' | 'error'): void {
  this.message.set(msg);
  this.messageType.set(type);
  setTimeout(() => this.message.set(""), 5000);
}
```

```

```

`src/app/register/register.html`
```html
<div class="register-container">
  <div class="register-card">
    <h2>Register</h2>

    @if (message()) {
      <div class="alert" [class.alert-success]="messageType() === 'success'">
        [class.alert-error]="messageType() === 'error'">
        {{ message() }}
      </div>
    }

    <form (ngSubmit)="onSubmit()" #registerForm="ngForm">
      <div class="form-group">
        <label for="username">Username *</label>
        <input type="text" id="username" name="username" [(ngModel)]="user.username" required
          class="form-control" placeholder="Enter username">
      </div>

      <div class="form-group">
        <label for="email">Email *</label>
        <input type="email" id="email" name="email" [(ngModel)]="user.email" required class="form-
control"
          placeholder="Enter email">
      </div>

      <div class="form-row">
        <div class="form-group">
          <label for="firstName">First Name</label>
          <input type="text" id="firstName" name="firstName" [(ngModel)]="user.firstName" class="form-
control"
            placeholder="First name">
        </div>

        <div class="form-group">
          <label for="lastName">Last Name</label>
          <input type="text" id="lastName" name="lastName" [(ngModel)]="user.lastName" class="form-
control"
            placeholder="Last name">
        </div>
      </div>

      <div class="form-group">
        <label for="phone">Phone</label>
        <input type="tel" id="phone" name="phone" [(ngModel)]="user.phone" class="form-control"
          placeholder="Enter phone number">
      </div>

      <div class="form-group">
        <label for="password">Password *</label>
        <input type="password" id="password" name="password" [(ngModel)]="user.password" required
          minlength="6"
          class="form-control" placeholder="Enter password">
      </div>

      <div class="form-group">

```

```
<label for="confirmPassword">Confirm Password *</label>
<input type="password" id="confirmPassword" name="confirmPassword"
[(ngModel)]="confirmPassword"
    required class="form-control" placeholder="Confirm password">
</div>

<button type="submit" class="btn btn-primary" [disabled]="!registerForm.form.valid">
    Register
</button>
</form>

<p class="login-link">
    Already have an account? <a routerLink="/login">Login here</a>
</p>
</div>
</div>
```

`src/app/register/register.css`

```css
.register-container {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    padding: 20px;
}

.register-card {
    background: white;
    border-radius: 10px;
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
    padding: 40px;
    width: 100%;
    max-width: 500px;
}

.register-card h2 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
    font-size: 28px;
}

.alert {
    padding: 12px;
    border-radius: 5px;
    margin-bottom: 20px;
    text-align: center;
    font-weight: 500;
}

.alert-success {
    background-color: #d4edda;
    color: #155724;
    border: 1px solid #c3e6cb;
}
```

```

```
.alert-error {
 background-color: #f8d7da;
 color: #721c24;
 border: 1px solid #f5c6cb;
}

.form-group {
 margin-bottom: 20px;
}

.form-row {
 display: grid;
 grid-template-columns: 1fr 1fr;
 gap: 15px;
}

.form-group label {
 display: block;
 margin-bottom: 5px;
 color: #555;
 font-weight: 600;
}

.form-control {
 width: 100%;
 padding: 12px;
 border: 1px solid #ddd;
 border-radius: 5px;
 font-size: 14px;
 transition: border-color 0.3s;
 box-sizing: border-box;
}

.form-control:focus {
 outline: none;
 border-color: #667eea;
}

.btn {
 width: 100%;
 padding: 12px;
 border: none;
 border-radius: 5px;
 font-size: 16px;
 font-weight: 600;
 cursor: pointer;
 transition: background-color 0.3s;
}

.btn-primary {
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
 color: white;
}

.btn-primary:hover:not(:disabled) {
 opacity: 0.9;
}
```

```

.btn-primary:disabled {
 opacity: 0.6;
 cursor: not-allowed;
}

.login-link {
 text-align: center;
 margin-top: 20px;
 color: #666;
}

.login-link a {
 color: #667eea;
 text-decoration: none;
 font-weight: 600;
}

.login-link a:hover {
 text-decoration: underline;
}

@media (max-width: 600px) {
 .form-row {
 grid-template-columns: 1fr;
 }

 .register-card {
 padding: 30px 20px;
 }
}
```
```

```

#### ### 9. Profile Component

```

`src/app/profile/profile.ts`
```typescript
import { Component, OnInit, computed } from '@angular/core';
import { Router } from '@angular/router';
import { Auth } from '../services/auth';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-profile',
  imports: [CommonModule],
  templateUrl: './profile.html',
  styleUrls: ['./profile.css'],
})
export class Profile implements OnInit {
  currentUser = computed(() => this.authService.getCurrentUser());
}

constructor(private authService: Auth, private router: Router) { }

ngOnInit(): void {
  // Redirect to login if not authenticated
  if (!this.authService.isLoggedIn()) {
    this.router.navigate(['/login']);
  }
}
```

```

```

}

onLogout(): void {
 this.authService.logout();
 this.router.navigate(['/login']);
}
}

```
##### `src/app/profile/profile.html`  

```html


<div class="profile-card">
 <div class="profile-header">
 <div class="profile-avatar">
 {{ currentUser()?.username && currentUser()!.username.charAt(0).toUpperCase() || 'U' }}
 </div>
 <h2>Welcome, {{ currentUser()?.username }}!</h2>
 </div>

 <div class="profile-info">
 <h3>User Information</h3>

 <div class="info-group">
 <label>Username:</label>
 {{ currentUser()?.username }}
 </div>

 <div class="info-group">
 <label>Email:</label>
 {{ currentUser()?.email }}
 </div>

 @if (currentUser()?.firstName) {
 <div class="info-group">
 <label>First Name:</label>
 {{ currentUser()?.firstName }}
 </div>
 }

 @if (currentUser()?.lastName) {
 <div class="info-group">
 <label>Last Name:</label>
 {{ currentUser()?.lastName }}
 </div>
 }

 @if (currentUser()?.phone) {
 <div class="info-group">
 <label>Phone:</label>
 {{ currentUser()?.phone }}
 </div>
 }

 @if (currentUser()?.id) {
 <div class="info-group">
 <label>User ID:</label>
 {{ currentUser()?.id }}
 </div>
 }
 </div>
 </div>


```

```
</div>
}
</div>

<button class="btn btn-logout" (click)="onLogout()">
 Logout
</button>
</div>
</div>
```

##### `src/app/profile/profile.css`  
``css
.profile-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  padding: 20px;
}

.profile-card {
  background: white;
  border-radius: 10px;
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
  padding: 40px;
  width: 100%;
  max-width: 600px;
}

.profile-header {
  text-align: center;
  margin-bottom: 30px;
  padding-bottom: 20px;
  border-bottom: 2px solid #f0f0f0;
}

.profile-avatar {
  width: 100px;
  height: 100px;
  border-radius: 50%;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 48px;
  font-weight: bold;
  margin: 0 auto 20px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

.profile-header h2 {
  color: #333;
  margin: 0;
  font-size: 28px;
}
```

```
.profile-info {  
    margin-bottom: 30px;  
}  
  
.profile-info h3 {  
    color: #555;  
    margin-bottom: 20px;  
    font-size: 20px;  
    border-bottom: 2px solid #667eea;  
    padding-bottom: 10px;  
}  
  
.info-group {  
    display: flex;  
    padding: 15px;  
    margin-bottom: 10px;  
    background-color: #f8f9fa;  
    border-radius: 5px;  
    align-items: center;  
}  
  
.info-group label {  
    font-weight: 600;  
    color: #555;  
    min-width: 120px;  
    margin-right: 15px;  
}  
  
.info-group span {  
    color: #333;  
    flex: 1;  
    word-break: break-word;  
}  
  
.btn {  
    width: 100%;  
    padding: 12px;  
    border: none;  
    border-radius: 5px;  
    font-size: 16px;  
    font-weight: 600;  
    cursor: pointer;  
    transition: all 0.3s;  
}  
  
.btn-logout {  
    background-color: #dc3545;  
    color: white;  
}  
  
.btn-logout:hover {  
    background-color: #c82333;  
    transform: translateY(-2px);  
    box-shadow: 0 4px 10px rgba(220, 53, 69, 0.3);  
}  
  
@media (max-width: 600px) {
```

```
.profile-card {  
  padding: 30px 20px;  
}  
  
.info-group {  
  flex-direction: column;  
  align-items: flex-start;  
}  
  
.info-group label {  
  min-width: auto;  
  margin-bottom: 5px;  
}  
}  
...  
...
```

10. Global Styles

```
##### `src/styles.css`  
```css  
/* You can add global styles to this file, and also import other style files */
```

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

body {
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
 background-color: #f5f5f5;
}
...
```

### ## 🚀 Getting Started

```
Prerequisites
- Node.js (version 18 or higher)
- npm (Node Package Manager)
```

#### ### Installation Steps

##### 1. \*\*Clone or download the project\*\*

```
```bash  
# If using git  
git clone <repository-url>  
cd user-auth-app  
...  
...
```

2. **Install dependencies**

```
```bash  
npm install
...
...
```

##### 3. \*\*Start the development server\*\*

```
```bash  
npm start  
...  
...
```

4. **Open your browser**
Navigate to `http://localhost:4200`

Build for Production

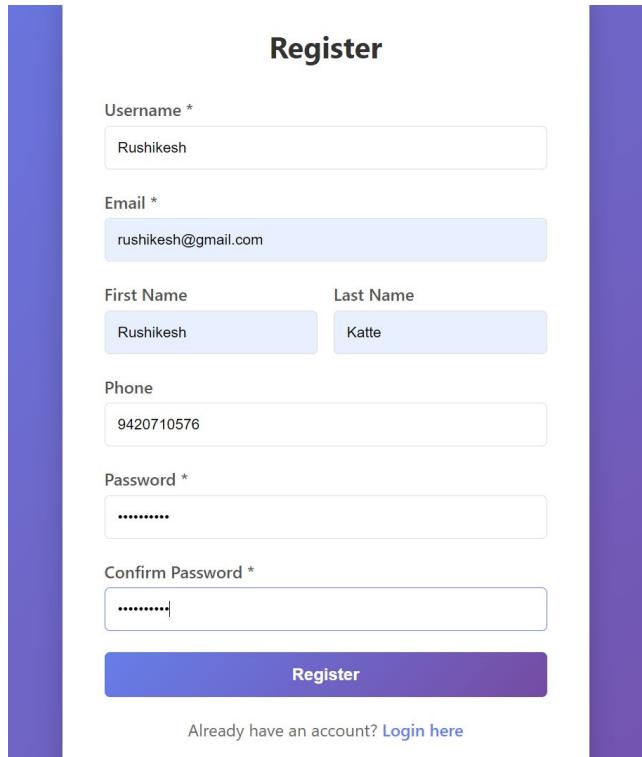
```bash

npm run build

```

Output

Register



The screenshot shows a registration form titled "Register". It includes fields for Username, Email, First Name, Last Name, Phone, Password, and Confirm Password. A "Register" button is at the bottom, and a link to "Login here" is at the bottom right.

Register

Username *

Email *

First Name

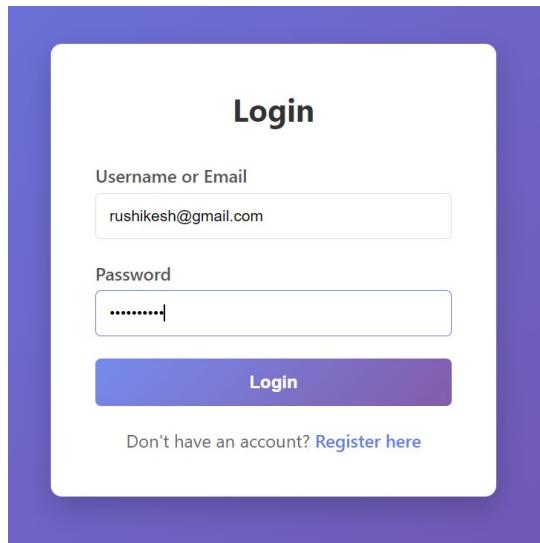
Last Name

Phone

Password *

Confirm Password *

Login



The screenshot shows a login form titled "Login". It includes fields for Username or Email and Password. A "Login" button is at the bottom, and a link to "Register here" is at the bottom right.

Login

Username or Email

Password

User Profile



Welcome, Rushikesh!

User Information

Username: Rushikesh

Email: rushikesh@gmail.com

First Name: Rushikesh

Last Name: Katte

Phone: 9420710576

User ID: 1

[Logout](#)

Assignment 4A

Name: Rushikesh Katte

Roll Number: 23572

Subject: Web Technology

1 Create a folder

```
mkdir static-website
```

```
cd static-website
```

2 Initialize Node.js project

```
npm init -y
```

This will create a package.json file automatically.

3 Install Express

```
npm install express
```

4 Create folder for static files

```
mkdir public
```

Inside public folder, create an index.html file:

 **public/index.html**

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>My Static Website</title>
</head>
<body>
    <h1>Hello World! This is a static website served by Node.js</h1>
</body>
</html>
```

5 Create main server file

Create server.js in the root directory:



```
const express = require('express');
const app = express();
const PORT = 3000;

// Serve static files from "public" folder
app.use(express.static('public'));

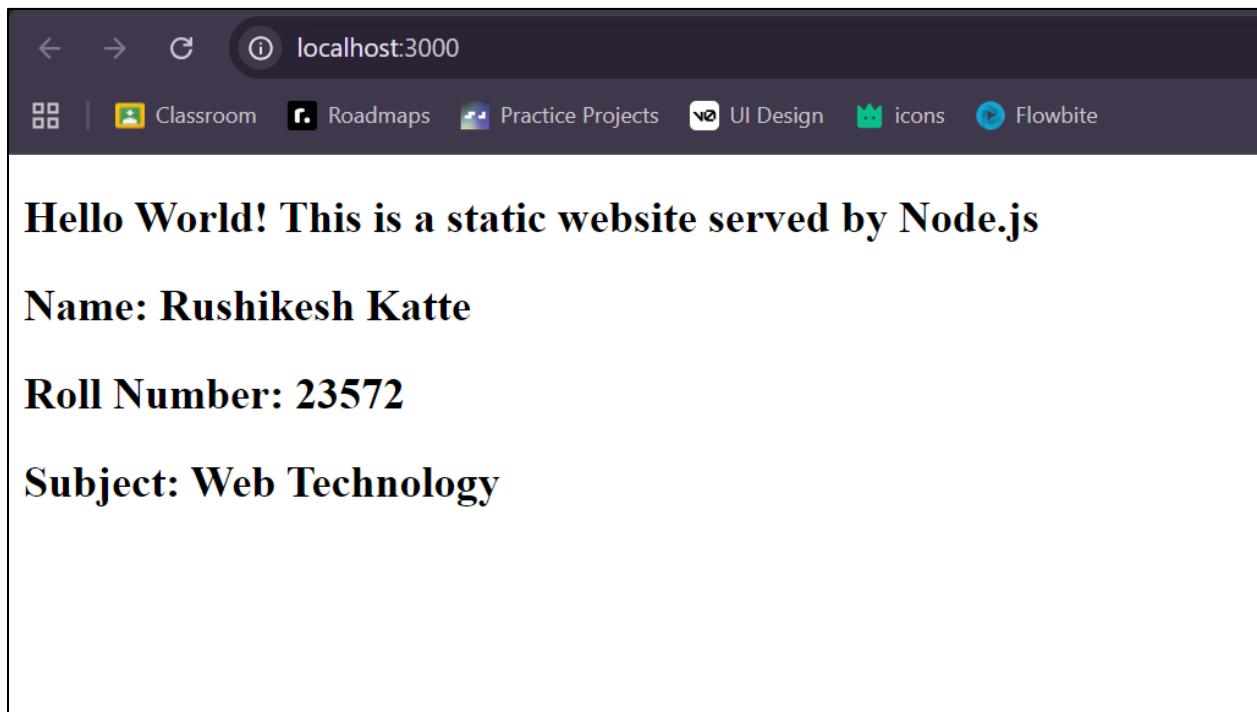
app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);
});
```

6 Start the server

```
node server.js
```

Now open your browser → **http://localhost:3000**

You will see your static website served by Node.js



Assignment 4B

Name: Rushikesh Katte

Roll Number: 23572

Subject: Web Technology

BACKEND:

1 Create project folder

```
mkdir crud-api  
cd crud-api
```

2 Initialize Node project

```
npm init -y
```

3 Install dependencies

```
npm install express mongoose cors
```

Folder Structure

```
crud-api  
|   +-- server.js  
|   +-- models  
|       |   +-- itemModel.js
```

Create MongoDB Model

📌 **models/itemModel.js**

```
const mongoose = require('mongoose');

const ItemSchema = new mongoose.Schema({
  name: String,
  description: String,
  price: Number
});

module.exports = mongoose.model('Item', ItemSchema);
```

Main Server Code (CRUD APIs)

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const Item = require('./models/itemModel');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(express.json());

mongoose.connect('mongodb://localhost:27017/assignment3C')
.then(() => console.log("MongoDB Connected"))
.catch(error => console.log(error));

app.post('/items', async (req, res) => {

  try {
    const item = await Item.create(req.body);
    res.status(201).json(item);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

app.get('/items', async (req, res) => {
  const items = await Item.find();
  res.json(items);
});

app.put('/items/:id', async (req, res) => {
  try {
    const item = await Item.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(item);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
```

```

app.delete('/items/:id', async (req, res) => {
  try {
    await Item.findByIdAndDelete(req.params.id);
    res.json({ message: "Item deleted successfully" });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));

```

Run Server

node server.js

```
C:\Users\rushi\OneDrive\Desktop\WT LAB\04\02. CRUD\crud-api>node server.js
Server running on http://localhost:3000
MongoDB Connected
```

Test API using Postman

Operation	Method	Endpoint	Request Body Example
Create	POST	/items	{ "name": "Pen", "description": "Blue Ink", "price": 10 }
Read	GET	/items	Not required
Update	PUT	/items/:id	{ "price": 15 }
Delete	DELETE	/items/:id	Not required

Create:

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar displays collections like 'Assignment 4B' and environments. The main area shows a 'New Request' dialog for a 'POST' method to 'http://localhost:3000/items'. The 'Body' tab is selected, showing a raw JSON payload:

```
{ "name": "Pen", "description": "Blue Ink", "price": 10 }
```

The response section shows a successful '201 Created' status with a response body identical to the request payload.

Read:

The screenshot shows the Postman interface with a dark theme. The sidebar and request setup are similar to the previous 'Create' screenshot. The 'GET' method is selected for the request to 'http://localhost:3000/items'. A 'Query Params' table is present with a single entry 'Key'. The response section shows a successful '200 OK' status with a JSON array containing one item, which is identical to the one created in the 'Create' request.

Key	Value	Description
Key	Value	Description

Update:

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar displays a collection named "crud_wt" containing several requests, with one labeled "Assignment 4B / New Request" highlighted. The main workspace shows an "Assignment 4B / New Request" screen. The method is set to "PUT" and the URL is "http://localhost:3000/items/6901819db76a279a94f190c7". The "Body" tab is selected, showing a JSON payload: { "price": 15 }. Below the request, the response status is "200 OK" with a response time of 27 ms and a body size of 358 B. The response body is displayed as a JSON object with fields: _id, name, description, price, and __v.

Delete:

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar displays a collection named "crud_wt" containing several requests, with one labeled "Assignment 4B / New Request" highlighted. The main workspace shows an "Assignment 4B / New Request" screen. The method is set to "DELETE" and the URL is "http://localhost:3000/items/6901819db76a279a94f190c7". The "Body" tab is selected, showing an empty JSON object. Below the request, the response status is "200 OK" with a response time of 18 ms and a body size of 306 B. The response body is displayed as a JSON object with a single field: "message" with the value "Item deleted successfully".

Frontend:

Folder Structure

Add a folder in the same project:

```
crud-api
├── server.js
└── models/
└── public/
    ├── index.html
    └── script.js
```

Also update your server to serve static files:

In **server.js**, add:

```
app.use(express.static('public'));
```

Frontend Code

1 index.html

public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRUD Frontend</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="p-4">
  <div class="container">
    <h2 class="mb-4 text-center">CRUD Application</h2>

    <!-- Add / Update Form -->
    <div class="card p-3">
      <input type="hidden" id="itemId">
      <div class="mb-2">
        <label>Name</label>
        <input type="text" id="name" class="form-control">
      </div>
      <div class="mb-2">
        <label>Description</label>
        <input type="text" id="description" class="form-control">
      </div>
    </div>
  </div>
</body>
</html>
```

```

</div>
<div class="mb-2">
  <label>Price</label>
  <input type="number" id="price" class="form-control">
</div>
<button class="btn btn-success" onclick="addItem()">Add Item</button>
<button class="btn btn-warning d-none" id="updateBtn" onclick="updateItem()">Update Item</button>
</div>

<hr>

<h4>Items</h4>
<table class="table table-bordered">
  <thead>
    <tr>
      <th>Name</th>
      <th>Description</th>
      <th>Price</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody id="itemTable"></tbody>
</table>
</div>

<script src="script.js"></script>
</body>
</html>

```

2 script.js

 public/script.js

```

const API = "http://localhost:3000/items";

async function fetchItems() {
  const res = await fetch(API);
  const items = await res.json();

  const table = document.getElementById("itemTable");
  table.innerHTML = "";

  items.forEach(item => {
    table.innerHTML += `
      <tr>
        <td>${item.name}</td>
        <td>${item.description}</td>
        <td>${item.price}</td>
        <td>
          <button class="btn btn-primary btn-sm" onclick="editItem('${item._id}')">Edit</button>

```

```

        <button class="btn btn-danger btn-sm" onclick="deleteItem('${item._id}')">Delete</button>
      </td>
    </tr>
  );
};

}

async function addItem() {
  const name = document.getElementById("name").value;
  const description = document.getElementById("description").value;
  const price = document.getElementById("price").value;

  await fetch(API, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name, description, price })
  });

  clearForm();
  fetchItems();
}

async function editItem(id) {
  const res = await fetch(`${API}/${id}`);
  const item = await res.json();

  document.getElementById("itemId").value = item._id;
  document.getElementById("name").value = item.name;
  document.getElementById("description").value = item.description;
  document.getElementById("price").value = item.price;

  document.querySelector("button.btn-success").classList.add("d-none");
  document.getElementById("updateBtn").classList.remove("d-none");
}

async function updateItem() {
  const id = document.getElementById("itemId").value;
  const name = document.getElementById("name").value;
  const description = document.getElementById("description").value;
  const price = document.getElementById("price").value;

  await fetch(`${API}/${id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name, description, price })
  });

  clearForm();
  fetchItems();
}

async function deleteItem(id) {
  await fetch(`${API}/${id}`, { method: "DELETE" });
}

```

```

    fetchItems();
}

function clearForm() {
    document.getElementById("itemId").value = "";
    document.getElementById("name").value = "";
    document.getElementById("description").value = "";
    document.getElementById("price").value = "";

    document.querySelector("button.btn-success").classList.remove("d-none");
    document.getElementById("updateBtn").classList.add("d-none");
}

// Initial load
fetchItems();

```

Run Full Application

1 Start MongoDB

mongod

2 Run backend

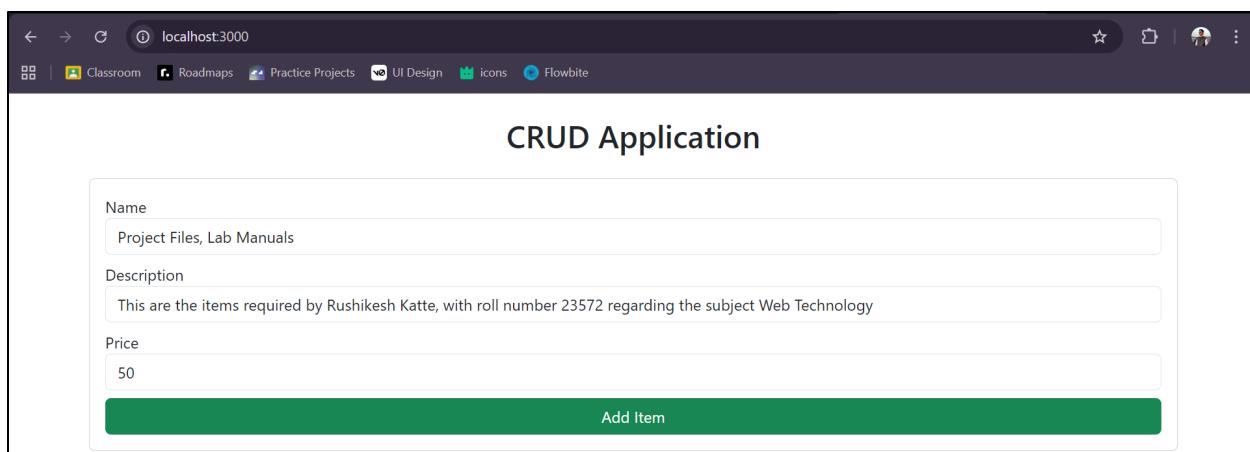
node server.js

3 Open browser

<http://localhost:3000>

You can now **Add / View / Edit / Delete** items from UI
 Full CRUD system working!

Add Item:



Items			
Name	Description	Price	Actions
Project Files, Lab Manuals	This are the items required by Rushikesh Katte, with roll number 23572 regarding the subject Web Technology	50	Edit Delete

Update Item:

CRUD Application

Name	Project Files, Lab Manuals, Pen, Pencil
Description	This are the items required by Rushikesh Katte, with roll number 23572 regarding the subject Web Technology.
Price	50
Update Item	

Items			
Name	Description	Price	Actions
Project Files, Lab Manuals, Pen, Pencil	This are the items required by Rushikesh Katte, with roll number 23572 regarding the subject Web Technology.	50	Edit Delete

Delete Item:

Items			
Name	Description	Price	Actions

Assignment 5A

Name: Rushikesh Katte

Roll Number: 23572

Subject: Web Technology

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>jQuery Mobile Demo</title>

    <link rel="stylesheet" href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
    <script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>
    <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>

<style>
    .hero {
        text-align: center;
        padding: 24px 12px;
    }

    .hero h2 {
```

```
margin: 0 0 8px;  
}  
  
.footer-note {  
font-size: 12px;  
color: #666;  
text-align: center;  
padding: 8px 0;  
}  
  
.responsive-img {  
max-width: 100%;  
height: auto;  
border-radius: 8px;  
}  
  
</style>  
</head>  
  
<body>  
  
<!-- HOME PAGE -->  
<div data-role="page" id="home" data-theme="a">  
  <div data-role="header" data-position="fixed">  
    <h1>My Mobile Site</h1>  
    <div data-role="navbar">  
      <ul>  
        <li><a href="#home" class="ui-btn-active" data-icon="home">Home</a></li>
```

```
<li><a href="#list" data-icon="bullets">List</a></li>
<li><a href="#contact" data-icon="mail">Contact</a></li>
</ul>
</div>
</div>

<div role="main" class="ui-content">
  <div class="hero">
    <h2>Welcome</h2>
    <p>A simple, responsive jQuery Mobile starter.</p>
  </div>

  <!-- jQuery Mobile responsive grid -->
  <div class="ui-grid-a ui-responsive">
    <div class="ui-block-a">
      <div class="ui-body ui-body-a">
        <h3>Fast Setup</h3>
        <p>Built with CDN links—no local setup required.</p>
      </div>
    </div>
    <div class="ui-block-b">
      <div class="ui-body ui-body-a">
        <h3>Responsive</h3>
        <p>Adapts to phones, tablets, and desktops.</p>
      </div>
    </div>
  </div>
</div>
```

```
<div style="margin-top:16px; text-align:center;">
    
</div>

<a href="#list" class="ui-btn ui-btn-b ui-corner-all ui-shadow" data-icon="arrow-r"
    data-iconpos="right">Explore the List</a>
</div>

<div data-role="footer" data-position="fixed">
    <h4>Home</h4>
</div>
</div>

<!-- LIST PAGE -->
<div data-role="page" id="list" data-theme="a">
    <div data-role="header" data-position="fixed">
        <h1>Items</h1>
        <div data-role="navbar">
            <ul>
                <li><a href="#home" data-icon="home">Home</a></li>
                <li><a href="#list" class="ui-btn-active" data-icon="bullets">List</a></li>
                <li><a href="#contact" data-icon="mail">Contact</a></li>
            </ul>
        </div>
    </div>
</div>
```

```
<div role="main" class="ui-content">
  <form class="ui-filterable">
    <input id="filter-input" data-type="search" placeholder="Search items...">
  </form>

  <ul data-role="listview" data-inset="true" data-filter="true" data-input="#filter-input">
    <li data-role="list-divider">Featured</li>
    <li><a href="#">Alpha</a></li>
    <li><a href="#">Bravo</a></li>
    <li><a href="#">Charlie</a></li>
    <li><a href="#">Delta</a></li>
    <li data-role="list-divider">More</li>
    <li><a href="#">Echo</a></li>
    <li><a href="#">Foxtrot</a></li>
    <li><a href="#">Golf</a></li>
    <li><a href="#">Hotel</a></li>
  </ul>

  <a href="#home" class="ui-btn ui-corner-all" data-icon="arrow-l">Back to Home</a>
</div>

<div data-role="footer" data-position="fixed">
  <h4>List</h4>
</div>
</div>

<!-- CONTACT PAGE -->
```

```
<div data-role="page" id="contact" data-theme="a">
  <div data-role="header" data-position="fixed">
    <h1>Contact</h1>
    <div data-role="navbar">
      <ul>
        <li><a href="#home" data-icon="home">Home</a></li>
        <li><a href="#list" data-icon="bullets">List</a></li>
        <li><a href="#contact" class="ui-btn-active" data-icon="mail">Contact</a></li>
      </ul>
    </div>
  </div>

  <div role="main" class="ui-content">
    <form id="contact-form">
      <div class="ui-field-contain">
        <label for="name">Name</label>
        <input type="text" name="name" id="name" placeholder="Jane Doe" required>
      </div>
      <div class="ui-field-contain">
        <label for="email">Email</label>
        <input type="email" name="email" id="email" placeholder="jane@example.com" required>
      </div>
      <div class="ui-field-contain">
        <label for="message">Message</label>
        <textarea name="message" id="message" placeholder="Write your message..." required></textarea>
      </div>
    </form>
  </div>
```

```
<button type="submit" class="ui-btn ui-btn-b ui-corner-all ui-shadow" data-icon="check">Send</button>

</form>

</div>

<div data-role="footer" data-position="fixed">
    <h4>Contact</h4>
</div>
</div>

<script>
    // Prevent actual navigation for '#' links in list for demo clarity
    $(document).on('click', 'a[href="#"]', function (e) {
        e.preventDefault();
    });

    // Demo submit handler
    $(document).on('submit', '#contact-form', function (e) {
        e.preventDefault();

        var name = $('#name').val().trim();
        var email = $('#email').val().trim();
        var message = $('#message').val().trim();

        if (!name || !email || !message) {
            alert('Please fill out all fields.');
            return;
        }

        alert('Thanks, ' + name + '! Your message was captured locally.');
    });

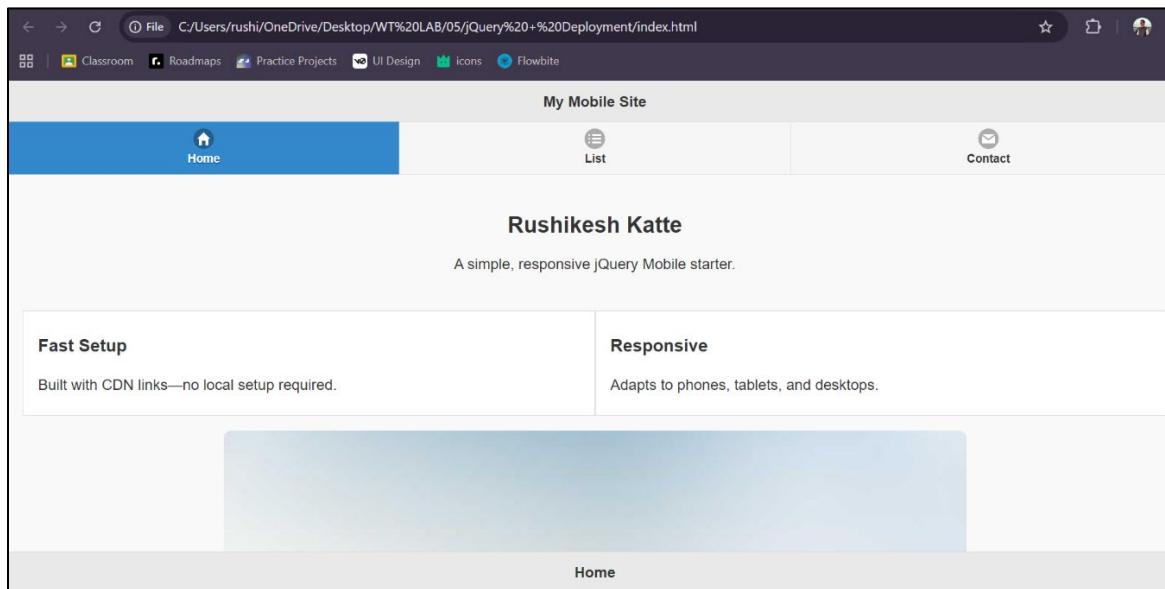
```

```

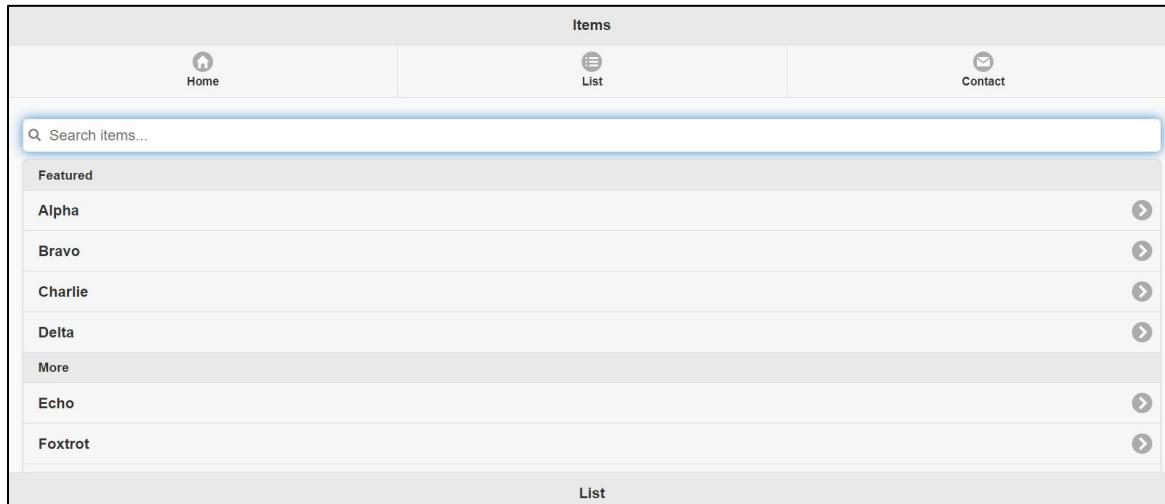
        this.reset();
    });
</script>
</body>
</html>

```

Home:



List:



Contact:

Contact

 Home	 List	 Contact
Name	Jane Doe	
Email	jane@example.com	
Message	Write your message...	
Send		

Contact

Assignment 5B

Name: Rushikesh Katte
Roll Number: 23572
Subject: Web Technology

First Create a Bucket with selections as follows:

The screenshot shows the 'Create bucket' configuration page in the AWS S3 console. The 'Bucket name' field contains 'myawsbucket'. The 'AWS Region' dropdown is set to 'US West (Oregon) us-west-2'. A note about copying settings from an existing bucket is present, with a 'Choose bucket' button.

The screenshot shows the 'Default encryption' configuration page. It indicates that server-side encryption is automatically applied to new objects stored in the bucket. Under 'Encryption key type', 'Amazon S3-managed keys (SSE-S3)' is selected. Under 'Bucket Key', 'Enable' is selected. A note at the bottom states: 'After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.' A 'Create bucket' button is visible at the bottom right.

Assignment 5B

Upload Project Files:

The screenshot shows the AWS S3 console interface. In the top navigation bar, 'Services' is selected. The path 'Amazon S3 > Buckets > aws-made-easy-website' is visible. A modal window titled 'Upload one file to this site?' is open, containing the message 'This will upload all files from "images". Only do this if you trust the site.' with 'Upload' and 'Cancel' buttons. Below the modal, the main page shows the 'Upload' section with instructions to 'Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more'. A dashed box highlights a drag-and-drop area with the placeholder 'Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.'. Below this, a table lists 'Files and folders (1 Total, 264.0 B)'. The table includes columns for Name, Folder, Type, and Size. One entry is shown: 'index.html' (text/html, 264.0 B). Buttons for 'Remove', 'Add files', and 'Add folder' are available above the table. A 'Destination' section is at the bottom.

Enable Static Website Hosting:

The screenshot shows the 'Edit static website hosting' configuration page for the 'aws-made-easy-website' bucket. The top navigation bar shows the path 'Amazon S3 > Buckets > aws-made-easy-website > Edit static website hosting'. The main section is titled 'Static website hosting' with the sub-instruction 'Use this bucket to host a website or redirect requests. Learn more'. It contains two main configuration sections: 'Static website hosting' (with 'Enable' selected) and 'Hosting type' (with 'Host a static website' selected). A note at the bottom states: 'For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access'.

Assignment 5B

Uncheck Block public access:

The screenshot shows the 'Block public access (bucket settings)' section in the AWS Management Console. It includes a descriptive text about blocking public access through various methods like ACLs and policies, and a list of checkboxes for different access types. One checkbox, 'Block all public access', is checked.

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Output at the domain as follows:

<http://my-static-site.s3-website.us-west-2.amazonaws.com/index.html>

The screenshot shows a browser displaying a mobile-optimized website titled 'My Mobile Site'. The header includes navigation links for 'Home', 'List', and 'Contact'. The main content area features a heading 'Rushikesh Katte' and a subtext 'A simple, responsive jQuery Mobile starter.' Below this are two columns: 'Fast Setup' (with the note 'Built with CDN links—no local setup required.') and 'Responsive' (with the note 'Adapts to phones, tablets, and desktops.'). A large, scenic image of a mountain river valley serves as the background for the content area.