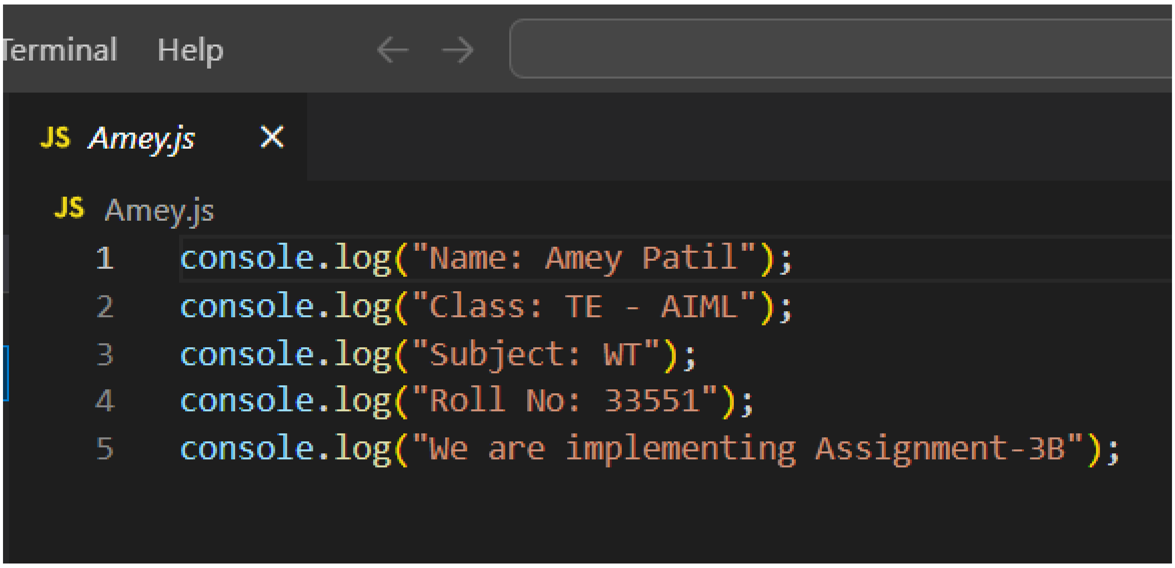
Assignment – 3 (B)

Step 1 : Installing docker, nodejs and wsl distribution

Step 2 : Restart the your system.

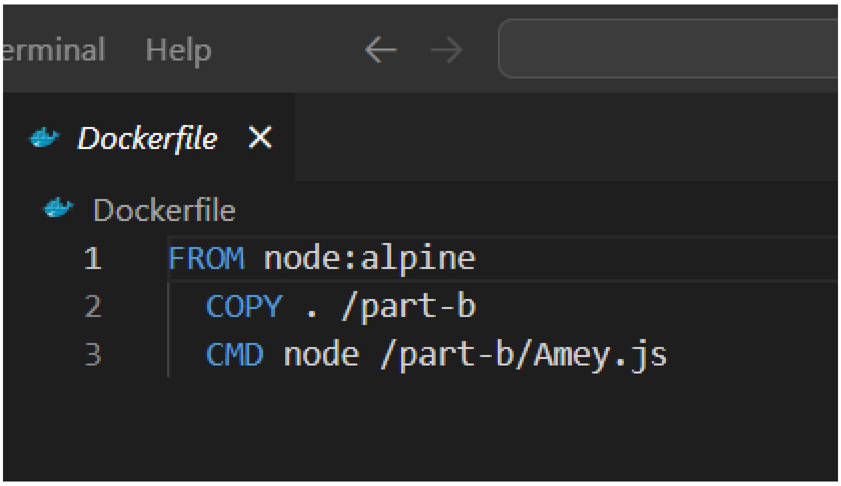
Step 3: Create a folder. Let the folder name be: part-b

Step-4: Create any file with any extension(such as .js, .txt, .py, etc....) in the same above folder(Here, part-b) and write some content in the file. Let the file name be:prasanna.js The file contains the following code:



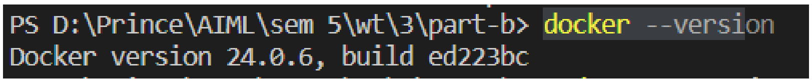
```
console.log("Name: Amey Patil");
console.log("Class: TE - AIML");
console.log("Subject: WT");
console.log("Roll No: 33551");
console.log("We are implementing Assignment-3B");
```

Step-5: Create file in the same folder with file name as- Dockerfile and press enter key. File: Dockerfile



```
FROM node:alpine
  COPY . /part-b
  CMD node /part-b/Amey.js
```

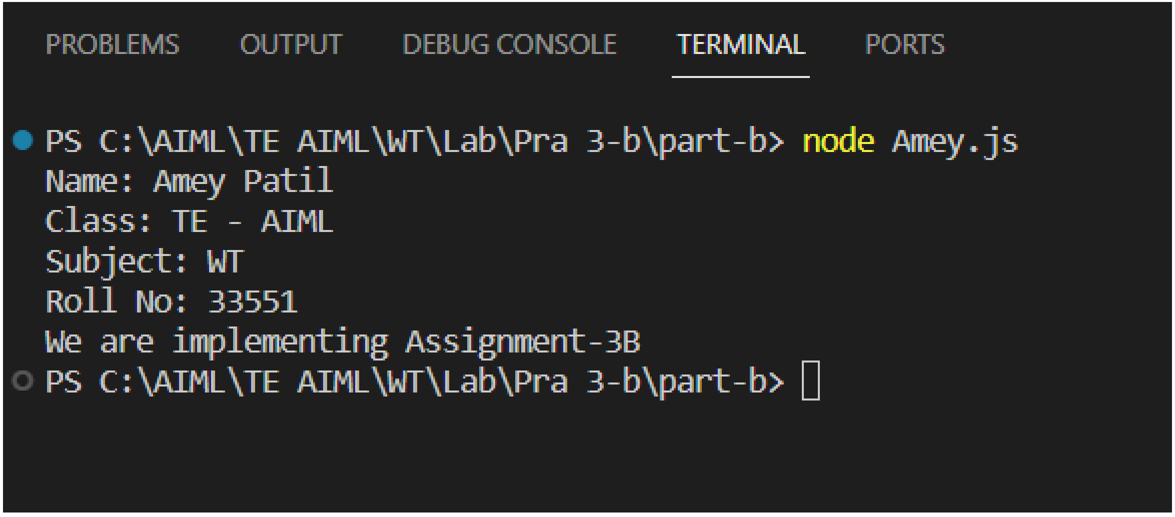
Step 6:Open terminal in VS Code and type the following command: docker --version



```
PS D:\Prince\AIML\sem 5\wt\3\part-b> docker --version
Docker version 24.0.6, build ed223bc
```

We will see the version of your docker if it is installed successfully.

Step-7: In the same terminal in VS Code, type the following command: node prasanna.js .

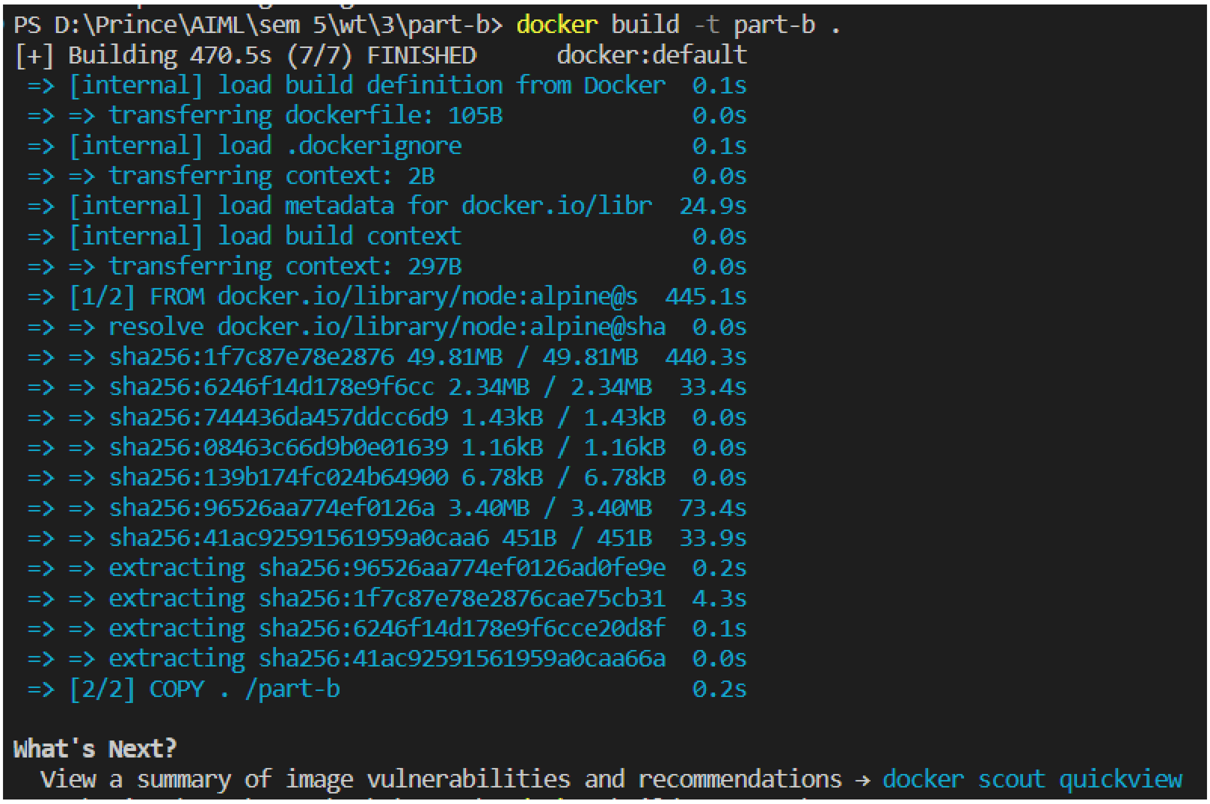


```
PS C:\AIML\TE AIML\WT\Lab\Pra 3-b\part-b> node Amey.js
Name: Amey Patil
Class: TE - AIML
Subject: WT
Roll No: 33551
We are implementing Assignment-3B
PS C:\AIML\TE AIML\WT\Lab\Pra 3-b\part-b>
```

It will display the contents in your file which is created in Step-4.

Step 8: In the same terminal in VS Code, type the following command: docker build -t part-b .

`docker build -t part-b .` command creates a Docker image tagged as "part-b" from the current directory's Dockerfile.



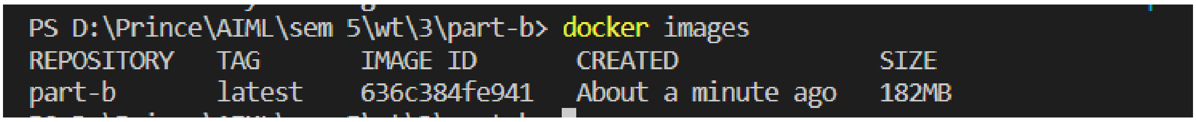
```
PS D:\Prince\AIML\sem 5\wt\3\part-b> docker build -t part-b .
[+] Building 470.5s (7/7) FINISHED      docker:default
 => [internal] load build definition from Docker  0.1s
 => => transferring dockerfile: 105B              0.0s
 => [internal] load .dockerignore                 0.1s
 => => transferring context: 2B                   0.0s
 => [internal] load metadata for docker.io/libr  24.9s
 => [internal] load build context                 0.0s
 => => transferring context: 297B                 0.0s
 => [1/2] FROM docker.io/library/node:alpine@s  445.1s
 => => resolve docker.io/library/node:alpine@sha  0.0s
 => => sha256:1f7c87e78e2876 49.81MB / 49.81MB  440.3s
 => => sha256:6246f14d178e9f6cc 2.34MB / 2.34MB  33.4s
 => => sha256:744436da457ddcc6d9 1.43kB / 1.43kB  0.0s
 => => sha256:08463c66d9b0e01639 1.16kB / 1.16kB  0.0s
 => => sha256:139b174fc024b64900 6.78kB / 6.78kB  0.0s
 => => sha256:96526aa774ef0126a 3.40MB / 3.40MB  73.4s
 => => sha256:41ac92591561959a0caa6 451B / 451B  33.9s
 => => extracting sha256:96526aa774ef0126ad0fe9e  0.2s
 => => extracting sha256:1f7c87e78e2876cae75cb31  4.3s
 => => extracting sha256:6246f14d178e9f6cce20d8f  0.1s
 => => extracting sha256:41ac92591561959a0caa66a  0.0s
 => [2/2] COPY . /part-b                          0.2s

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

We will see that it is successfully built.

Step 9 : In the same terminal in VS Code, type the following command: docker images

`docker images` lists available Docker images on your system, displaying their repository, tag, ID, and size information.



```
PS D:\Prince\AIML\sem 5\wt\3\part-b> docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
part-b       latest    636c384fe941   About a minute ago   182MB
```

We will see the docker images with REPOSITORY, TAG, IMAGE ID, CREATED (Time), SIZE.

How does docker work?

Docker works by providing a standard way to package and run applications. Docker containers are isolated from each other and from the underlying host operating system, which allows them to be easily moved from one environment to another.

Docker containers are built on top of images, which are lightweight, read-only file systems that contain everything needed to run an application. Images are created by packaging together the application code, all of its dependencies, and a base operating system. Once an image has been created, it can be used to create containers. Containers are lightweight and can be started and stopped quickly. They also share the kernel with the host operating system, which makes them more efficient than virtual machines.

Docker containers can be used to run a variety of applications, including web applications, databases, and batch jobs. They can also be used to create development environments and to deploy applications to production. Docker is a powerful tool that can help you to improve the efficiency and scalability of your application development and deployment process.