

Project Assignment 1: Getting Started with Docker Report

1. Get started guide:

a. Important link

- i. **Link to view the docker image:** <https://hub.docker.com/repository/docker/biraaj/getting-started>

b. Part 1: Overview of the get started guide

- i. **Container:** sandboxed process running on a host machine that is isolated from all other processes running on that host machine.
- ii. **Image:** An image is a read-only template with instructions for creating a Docker container

c. Part 2: how to build and run an image as a container

i. Clone the getting-started-app repository

1. Command: `git clone https://github.com/docker/getting-started-app.git`

2. **Git clone:** The git clone command is used to create a copy of a specific repository or branch within a repository.

The screenshot shows the Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with a dark theme, showing a folder named 'FINAL_EXP' containing 'getting-started-app' with files like 'spec', 'src', 'package.json', 'README.md', and 'yarn.lock'. The main area is the 'Welcome' view, which includes sections for 'Start' (with options like 'New File...', 'Open File...', 'Open Folder...', 'Clone Git Repository...', and 'Connect to...') and 'Walkthroughs' (with links to 'Get Started with VS Code' and 'Getting Started with Docker'). Below the Welcome view is a terminal window with the following PowerShell output:

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> echo %date% %time%; whoami;
%date%
%time%
wrathgladiator\biraa
PS D:\UTA\spring 24\courses\ds\project1\final_exp> git clone https://github.com/docker/getting-started-app.git
Cloning into 'getting-started-app'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 68 (delta 12), reused 10 (delta 10), pack-reused 33Receiving objects: 92% (63/68)
Receiving objects: 100% (68/68), 1.75 MiB | 4.86 MiB/s, done.
Resolving deltas: 100% (12/12), done.
PS D:\UTA\spring 24\courses\ds\project1\final_exp>
```

ii. Creating Docker File

1. **Dockerfile:** A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a project structure: FINAL_EXP > getting-started-app > Dockerfile. The Dockerfile is selected and highlighted in blue. The main area is the Editor, which displays the following Dockerfile content:

```
1 # syntax=docker/dockerfile:1
2
3 FROM node:18-alpine
4 WORKDIR /app
5 COPY .
6 RUN yarn install --production
7 CMD ["node", "src/index.js"]
8 EXPOSE 3000
```

Below the Editor is a Terminal window showing the output of a command:

```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

D:\UTA\spring 24\courses\ds\project1\final_exp>echo %date% %time%; %USERNAME%
Sun 02/11/2024 16:23:33.45; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp>
```

iii. Build the image

1. Command: **docker build -t getting-started**.
 - a. The docker build command builds Docker images from a Dockerfile.
 - b. -t option here is to Name the docker build

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure with a folder named "FINAL_EXP" containing a "getting-started-app" folder which includes "spec", "src", and "Dockerfile". Other files like "package.json", "README.md", and "yarn.lock" are also listed.
- Terminal:** The terminal tab is active, displaying the command "docker build -t getting-started ." and its execution log. The log shows the build process from loading the Dockerfile to transferring context, resolving images, and finally writing the image to the registry.

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 16:29:34.02; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker build -t getting-started .
[+] Building 16.1s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 188B
=> resolve image config for docker.io/docker/dockerfile:1
=> [auth] docker/dockerfile:pull token for registry-1.docker.io
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab2021
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437
=> [internal] load build context
=> => transferring context: 6.50MB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:02b89ae2a96e59517dee29e3411d72696785fa239dacd0f484710076b6eab8f6
=> => naming to docker.io/library/getting-started
```

iv. Start an app container

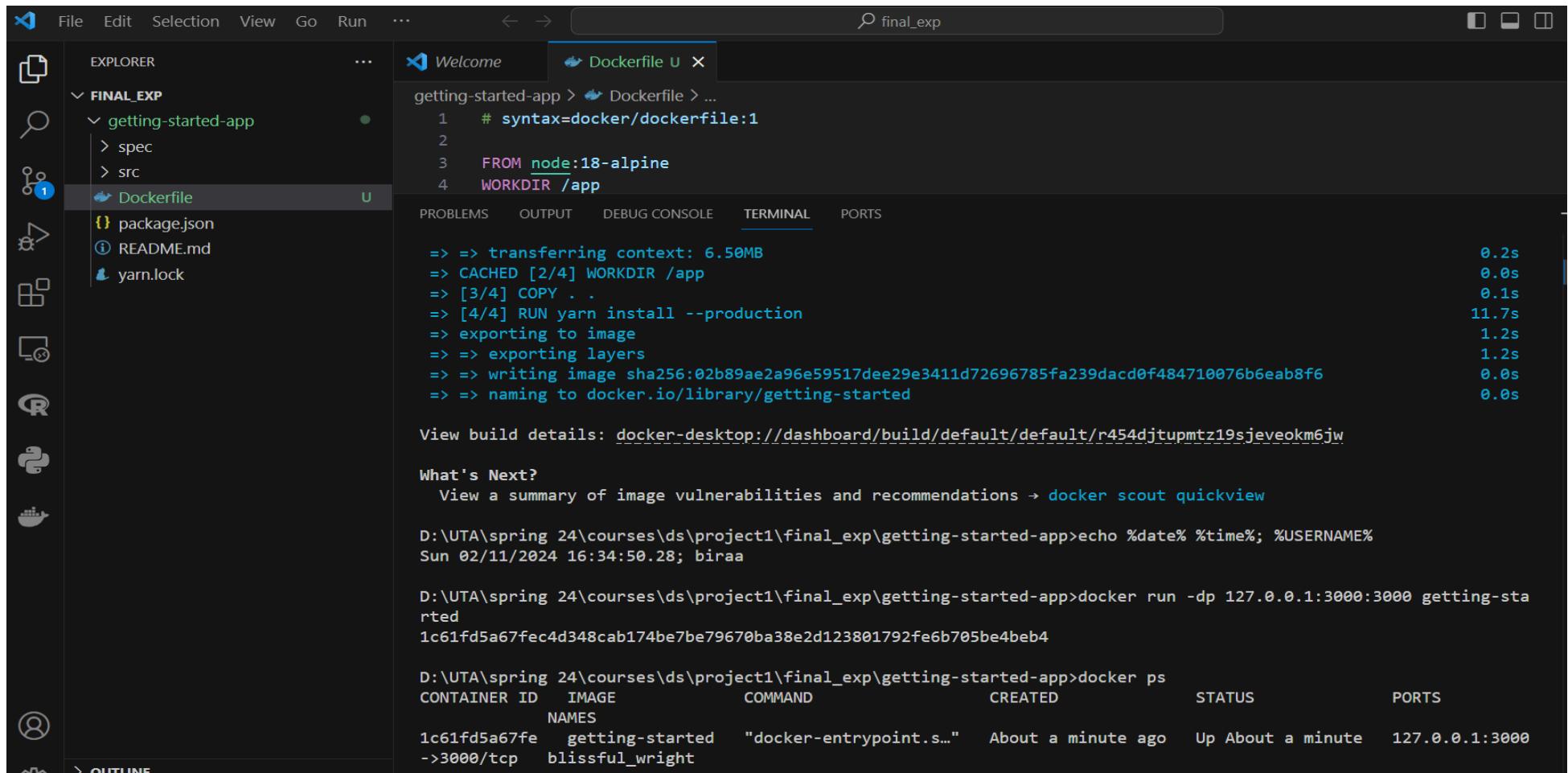
1. Command: **docker run -dp 127.0.0.1:3000:3000 getting-started**

- a. Docker run command is used to run a docker container.

- b. Here -d runs the container in background
- c. -p stands for –publish and creates a port mapping between host and a container. I.e 127.0.0.1:3000:3000 and here the container is **getting-started**

2. Command: **docker ps**

- a. This command lists all the running docker processes or containers.



The screenshot shows the Visual Studio Code interface with a dark theme. On the left is the Explorer sidebar showing a project structure under 'FINAL_EXP' with files like 'getting-started-app', 'Dockerfile', 'package.json', 'README.md', and 'yarn.lock'. The 'Dockerfile' tab is active in the main editor area. The terminal tab at the bottom shows the build logs for a Dockerfile:

```

getting-started-app > Dockerfile > ...
1 # syntax=docker/dockerfile:1
2
3 FROM node:18-alpine
4 WORKDIR /app

=> => transferring context: 6.50MB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:02b89ae2a96e59517dee29e3411d72696785fa239dacd0f484710076b6eab8f6
=> => naming to docker.io/library/getting-started

View build details: docker-desktop://dashboard/build/default/default/r454djtu...jw

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

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time% %USERNAME%
Sun 02/11/2024 16:34:50.28; biraa

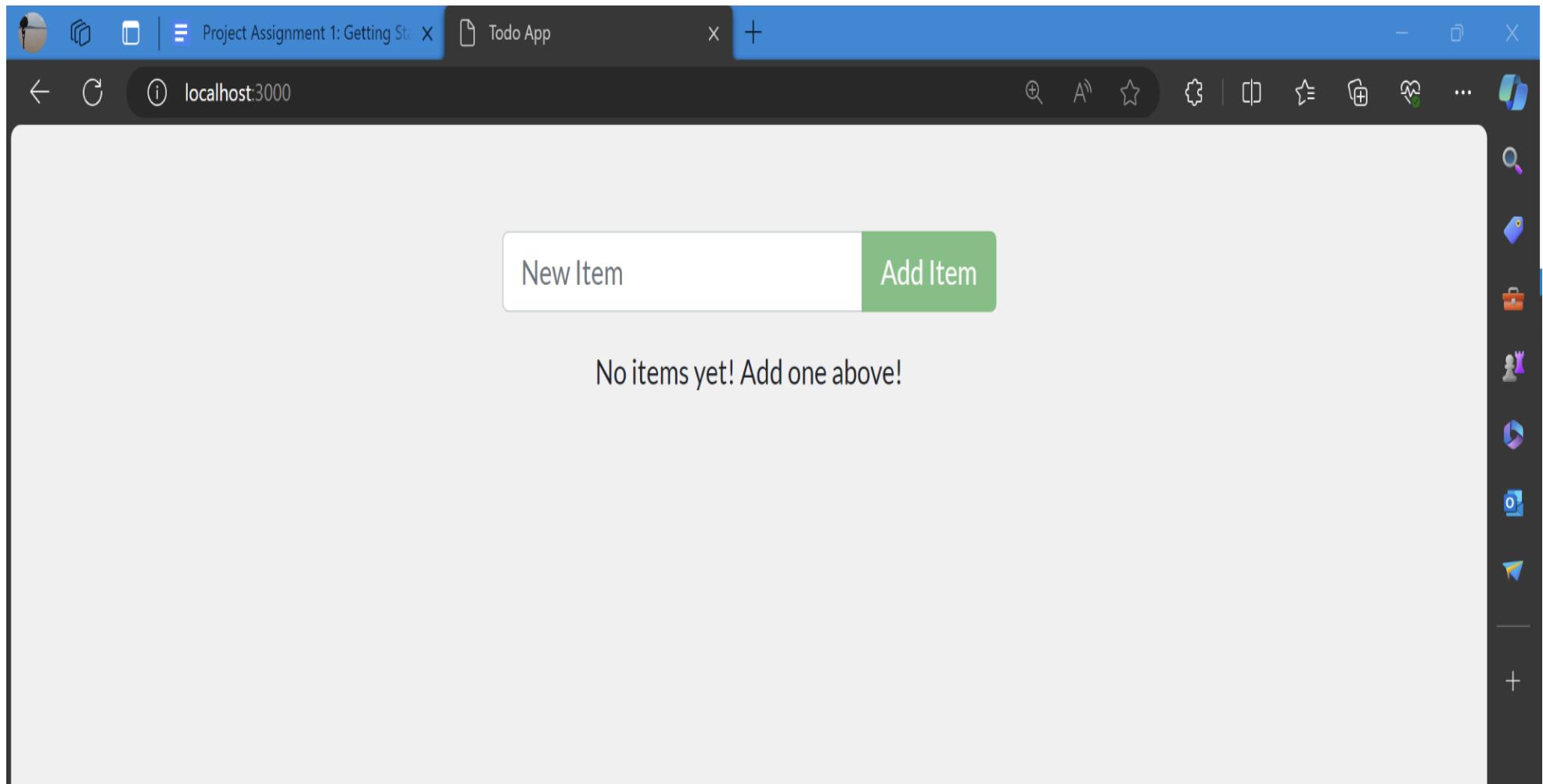
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -dp 127.0.0.1:3000:3000 getting-started
1c61fd5a67fec4d348cab174be7be79670ba38e2d123801792fe6b705be4beb4

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
1c61fd5a67fe getting-started "docker-entrypoint.s..." About a minute ago Up About a minute 127.0.0.1:3000->3000/tcp blissful_wright

```

v. Viewing the app after the container runs

1. Url: <http://localhost:3000>



d. Part 3: Update the application

i. Update the source code

1. Here we change some lines in app.js to see the updates being reflected after the build.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "FINAL_EXP". The "src" folder contains "static" which has "js" and "css" subfolders. "js" contains "app.js", "babel.min.js", "react-bootstrap.js", "react-dom.production.min.js", and "react.production.min.js". "css" contains "index.html", "index.js", and "Dockerfile".
- Editor:** The active file is "app.js". The code is as follows:

```
51
52     return (
53         <React.Fragment>
54             <AddItemForm onNewItem={onNewItem} />
55             {items.length === 0 && (
56                 <p className="text-center">You have no todo items yet! Add one above!</p>
57             )}
58             {items.length === 0 && (
59                 <p className="text-center">No items yet! Add one above!</p>
60             )}
61             {items.map(item => (
62                 <ItemDisplay
63                     item={item}
64                     key={item.id}
65                     onItemUpdate={onItemUpdate}
66             )}
67         )
68     )
69 }
```

- Terminal:** The terminal shows the following Docker command output:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
1c61fd5a67fe      getting-started    "docker-entrypoint.s..."   About a minute ago   Up About a minute   127.0.0.1:3000
->3000/tcp          blissful_wright
```

- Bottom Terminal:** The bottom terminal shows the command "echo %date% %time% %USERNAME%" and its output "Sun 02/11/2024 16:46:16.85; biraa".

ii. Remove the old container using CLI:

1. Commands:

a. **docker ps** (explained in part 2)

b. **docker stop 1c61fd5a67fe**

i. This command stops the run container. Here the container ID is **1c61fd5a67fe**

c. **docker rm 1c61fd5a67fe**

i. This command removes the container from the process list. Here we remove container **1c61fd5a67fe**

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows a project structure under "FINAL_EXP" named "getting-started-app". It includes files like Dockerfile, app.js, babel.min.js, react-bootstrap.js, react-dom.production.min.js, react.production.min.js, index.html, index.js, package.json, README.md, and yarn.lock.
- Code Editor:** The "app.js" file is open, showing React code. A specific line of code is highlighted:

```
51
52     return (
53         <React.Fragment>
54             <AddItemForm onNewItem={onNewItem} />
55             {items.length === 0 && (
56                 <p className="text-center">You have no todo items yet! Add one above!</p>
```
- Terminal View:** The terminal tab is active, showing the following Docker commands:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker stop
"docker stop" requires at least 1 argument.
See 'docker stop --help'.
```

```
Usage: docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

```
Stop one or more running containers
```

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker stop 1c61fd5a67fe
1c61fd5a67fe
```

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker rm 1c61fd5a67fe
1c61fd5a67fe
```

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1c61fd5a67fe	getting-started	"docker-entrypoint.s..."	16 minutes ago	Up 16 minutes	127.0.0.1:3000->3000/t	blissful_wright

iii. Build and Run the updated container

1. **docker build -t getting-started .**(Explained in part2)
2. **docker run -dp 127.0.0.1:3000:3000 getting-started**(Explained in part2)

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal output displays the command `docker build -t getting-started .` being run in the directory `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app`. The logs show the build process, including loading the Dockerfile, transferring context files, and finally creating the Docker image `sha256:38f104161a55a50183ae020e19340197c521fa620411c05ee1e6b36a6d2b4753`.

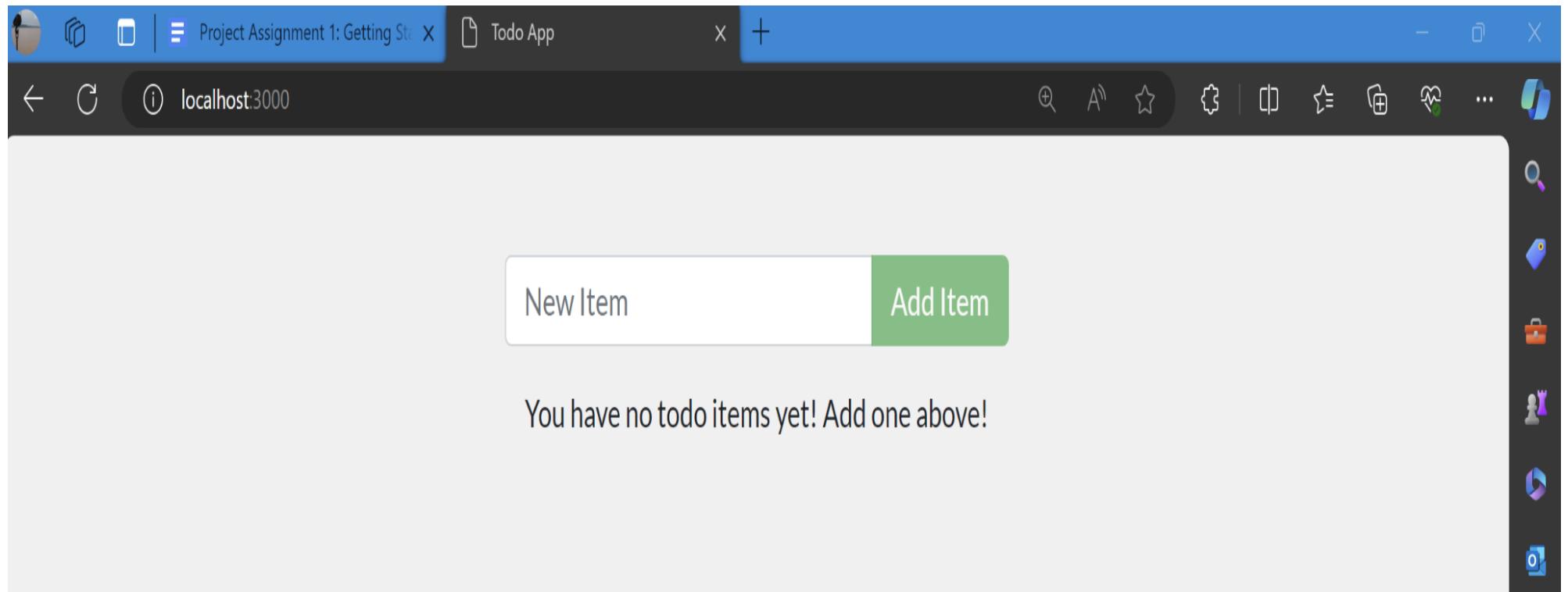
```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time% %USERNAME%
Sun 02/11/2024 16:54:30.25; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker build -t getting-started .
[+] Building 14.3s (13/13) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 188B
  => resolve image config for docker.io/docker/dockerfile:1
  => [auth] docker/dockerfile:pull token for registry-1.docker.io
  => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab2021
  => [internal] load metadata for docker.io/library/node:18-alpine
  => [auth] library/node:pull token for registry-1.docker.io
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/4] FROM docker.io/library/node:18-alpine@sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437
  => [internal] load build context
  => => transferring context: 10.05kB
  => CACHED [2/4] WORKDIR /app
  => [3/4] COPY .
  => [4/4] RUN yarn install --production
  => exporting to image
  => => exporting layers
  => => writing image sha256:38f104161a55a50183ae020e19340197c521fa620411c05ee1e6b36a6d2b4753
  => => naming to docker.io/library/getting-started

View build details: docker-desktop://dashboard/build/default/default/nz00vi7siqr5tnxmxgm6vyymtf

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

The terminal also shows the command `docker run -dp 127.0.0.1:3000:3000 getting-started` being run, which results in the output `733651c0c7e9296db41a936307f967f29ed3cb77207eb4ded5d20059286434db`.

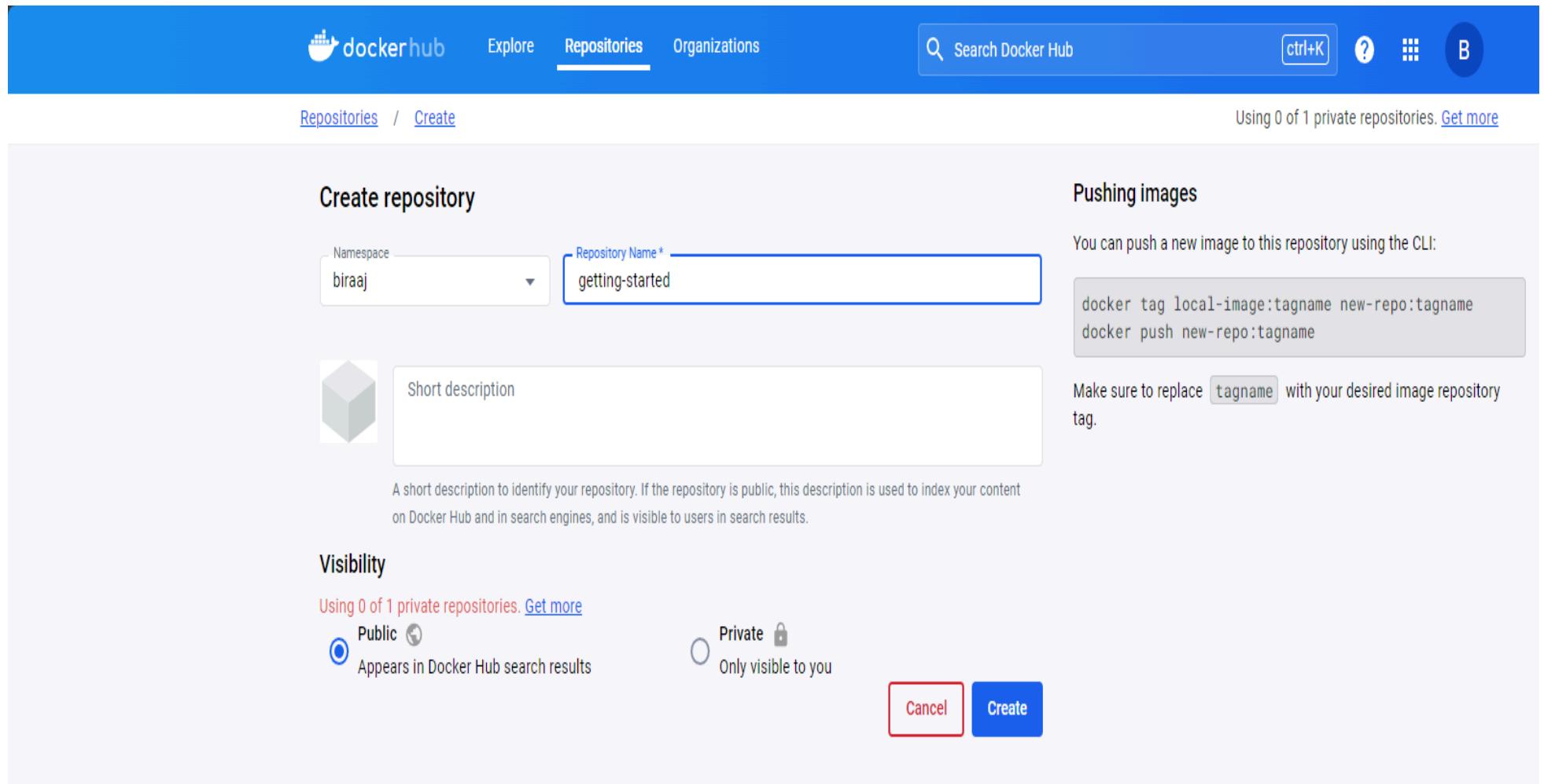


e. Part 4: Share the application

i. Link to view the docker image:

<https://hub.docker.com/repository/docker/biraaj/getting-started>

- ii. Signed in to Docker hub
- iii. Created a repository with a name “getting-started”
- iv. Pushed the image
 - 1. Signed in to the docker hub in cli.



The screenshot shows the Docker Hub interface for creating a new repository. At the top, there's a blue header bar with the Docker Hub logo, navigation links for 'Explore', 'Repositories' (which is underlined), 'Organizations', a search bar ('Search Docker Hub'), and various user icons. Below the header, the URL 'Repositories / Create' is visible, along with a message 'Using 0 of 1 private repositories. [Get more](#)'. The main form is titled 'Create repository'. It includes fields for 'Namespace' (set to 'biraaj') and 'Repository Name*' (set to 'getting-started'). There's also a 'Short description' field containing a placeholder icon of a grey cube. A note below it explains that this description is used for indexing and search results. The 'Visibility' section shows that 0 of 1 private repositories are being used. It offers two options: 'Public' (selected, indicated by a blue radio button) and 'Private' (indicated by a grey radio button). The 'Public' option includes the note 'Appears in Docker Hub search results'. On the right side, there's a 'Pushing images' section with instructions for using the CLI: 'docker tag local-image:tagname new-repo:tagname' and 'docker push new-repo:tagname'. It also advises replacing 'tagname' with the desired image repository tag. At the bottom right of the form are 'Cancel' and 'Create' buttons.

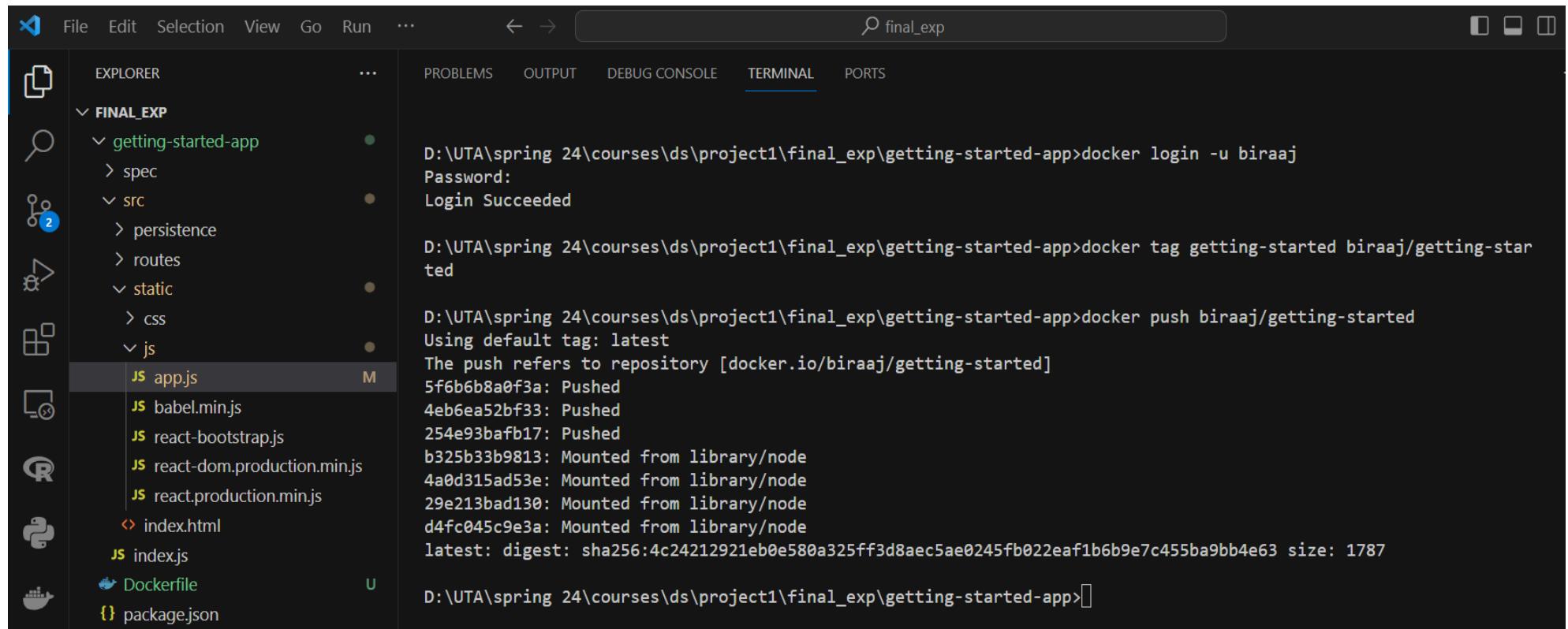
2. Use the docker tag and Docker push to push the image into the newly created repository
 - a. **docker login -u biraaj;**
 - i. This command signs in to docker hub in cli. Here the username is biraaj and it will ask password after executing this command.

b. docker tag getting-started biraaj/getting-started:

- i. This command gives a new name to the image.

c. docker push biraaj/getting-started

- i. This command pushes the newly created image to the repository mentioned at the end here.



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** final_exp
- Terminal Tab:** TERMINAL (selected)
- Output in Terminal:**

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker login -u biraaj
Password:
Login Succeeded

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker tag getting-started biraaj/getting-started
Using default tag: latest
The push refers to repository [docker.io/biraaj/getting-started]
5f6b6b8a0f3a: Pushed
4eb6ea52bf33: Pushed
254e93bafb17: Pushed
b325b33b9813: Mounted from library/node
4a0d315ad53e: Mounted from library/node
29e213bad130: Mounted from library/node
d4fc045c9e3a: Mounted from library/node
latest: digest: sha256:4c24212921eb0e580a325ff3d8aec5ae0245fb022eaf1b6b9e7c455ba9bb4e63 size: 1787

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>
```
- Explorer View:** Shows a project structure under 'FINAL_EXP' with 'getting-started-app' expanded, showing 'spec', 'src' (with 'persistence', 'routes', 'static' subfolders), 'css', 'js' (with 'app.js' selected), 'babel.min.js', 'react-bootstrap.js', 'react-dom.production.min.js', 'react.production.min.js', 'index.html', 'index.js', 'Dockerfile', and 'package.json'.

The screenshot shows the Docker Hub interface for a repository named 'biraaj/getting-started'. The top navigation bar includes links for 'Explore', 'Repositories' (which is the active tab), and 'Organizations'. A search bar at the top right contains the placeholder 'Search Docker Hub' with keyboard shortcut 'ctrl+K'. Below the search bar, it says 'Using 0 of 1 private repositories. [Get more](#)'. The main content area shows the repository details under the 'General' tab. It includes a note to 'Add a short description for this repository' and a 'Docker commands' section with the command 'docker push biraaj/getting-started:tagname'. The repository name 'biraaj / getting-started' is displayed, along with a 'Description' section stating 'This repository does not have a description' and a 'Last pushed' timestamp of '2 minutes ago'. The 'Tags' section shows one tag named 'latest' which is an 'Image'. The 'Automated Builds' section provides information about connecting GitHub or Bitbucket for automatic builds.

biraaj / getting-started

Description
This repository does not have a description

Last pushed: 2 minutes ago

Docker commands
To push a new tag to this repository:
`docker push biraaj/getting-started:tagname`

Tags
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	---	2 minutes ago

[See all](#)

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

[Upgrade](#)

3. We can see the latest tag as image is pushed to docker hub. As we didn't add a tag name it mentioned it as latest.
- v. Ran the Image on a new instance to verify that it runs on different machine.
1. opened Play with Docker and logged into the users(biraaj) docker account.
 2. Selected the ADD NEW INSTANCE
 3. In terminal ran the freshly pushed docker app

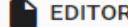
03:57:47

CLOSE SESSION

Instances  

+ ADD NEW INSTANCE

192.168.0.18 node1

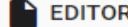
 

cn4lckc7_cn4lcm47vdo000eq4ci0

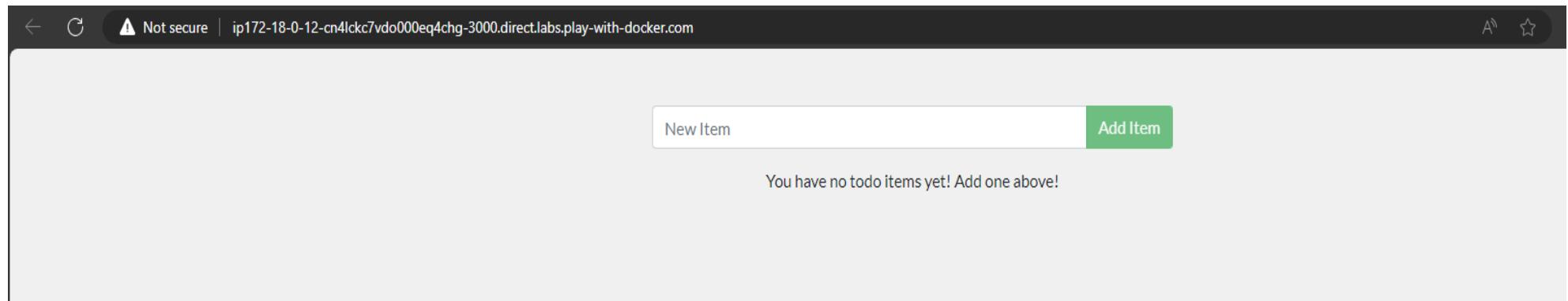
IP 192.168.0.18 OPEN PORT 3000

Memory 8.81% (352.4MiB / 3.906GiB) CPU 0.78%

SSH ssh ip172-18-0-12-cn4lckc7vdo000eq4chg@direct.labs.play 

```
#####
# WARNING!!!!
# This is a sandbox environment. Using personal credentials
# is HIGHLY! discouraged. Any consequences of doing so are
# completely the user's responsibilites.
#
# The PWD team.
#####
[node1] (local) root@192.168.0.18 ~
$ docker run -dp 0.0.0.0:3000 biraaj/getting-started
Unable to find image 'biraaj/getting-started:latest' locally
latest: Pulling from biraaj/getting-started
4abcf2066143: Pull complete
eb6c7c29ba4d: Pull complete
3d4a65156edf: Pull complete
5bdb6c27eb32: Pull complete
fc11686fa126: Pull complete
73ad7bc80341: Pull complete
48cb3e6e7f06: Pull complete
Digest: sha256:4c24212921eb0e580a325ff3d8aec5ae0245fb022eaf1b6b9e7c455ba9bb4e63
Status: Downloaded newer image for biraaj/getting-started:latest
5b478bd313cf038e11016d7ced1ab36e9811ab195da580b864fda85d8dfd08b7
```



f. Part 5: persist the DB

- i. Created a volume using docker volume command
 1. Command: **docker volume create todo-db**
 - a. This command is used to create volumes which is todo-db here.
 - b. Volumes provide the ability to connect specific filesystem paths of the container back to the host machine
- ii. Stopped and remove the todo app container using **docker rm -f <id>**
 1. As we know that docker rm is used to remove containers as explained in part the -f options forcefully stops the container and removes it.
- iii. Started the todo app container using “**docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started**”(explained in part2), but added the --mount option to specify a volume mount. Gave the volume a name, and mounted it to **/etc/todos** in the container.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** final_exp
- Terminal Tab:** TERMINAL (highlighted)
- Terminal Content:**

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 17:33:36.86; bira

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker volume create todo-db
todo-db

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
733651c0c7e9        getting-started   "docker-entrypoint.s..."   39 minutes ago   Up 39 minutes   127.0.0.1:3000->3000/t
cp                  beautiful_feynman

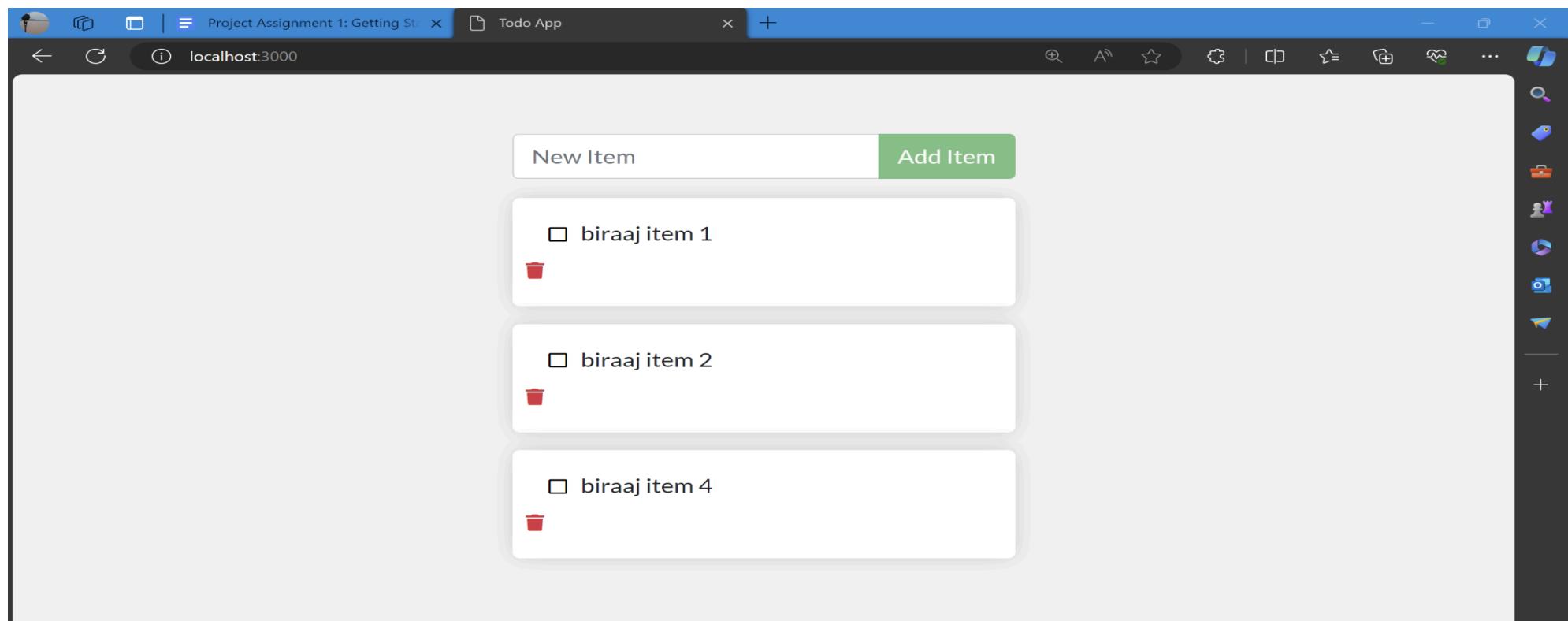
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker rm -f 733651c0c7e9
733651c0c7e9

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -dp 127.0.0.1:3000:3000 --mount typ
e=volume,src=todo-db,target=/etc/todos getting-started
1077f6ad92ca85666bdc037a09640b5942daf6624f737344531bef5bec339edf

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>
```
- Explorer View:** Shows a file tree for the 'FINAL_EXP' project, including 'getting-started-app' with subfolders 'spec', 'src' (containing 'persistence', 'routes', 'static' with 'css' and 'js' subfolders), and files like 'app.js', 'babel.min.js', 'react-bootstrap.js', etc.

iv. Verify that the data persists.

1. Once the container started we added some items to the application.



2. Stopped the container using `docker rm -f <id>`(explained in part2) and started a new container.

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 17:42:02.65; biraaj

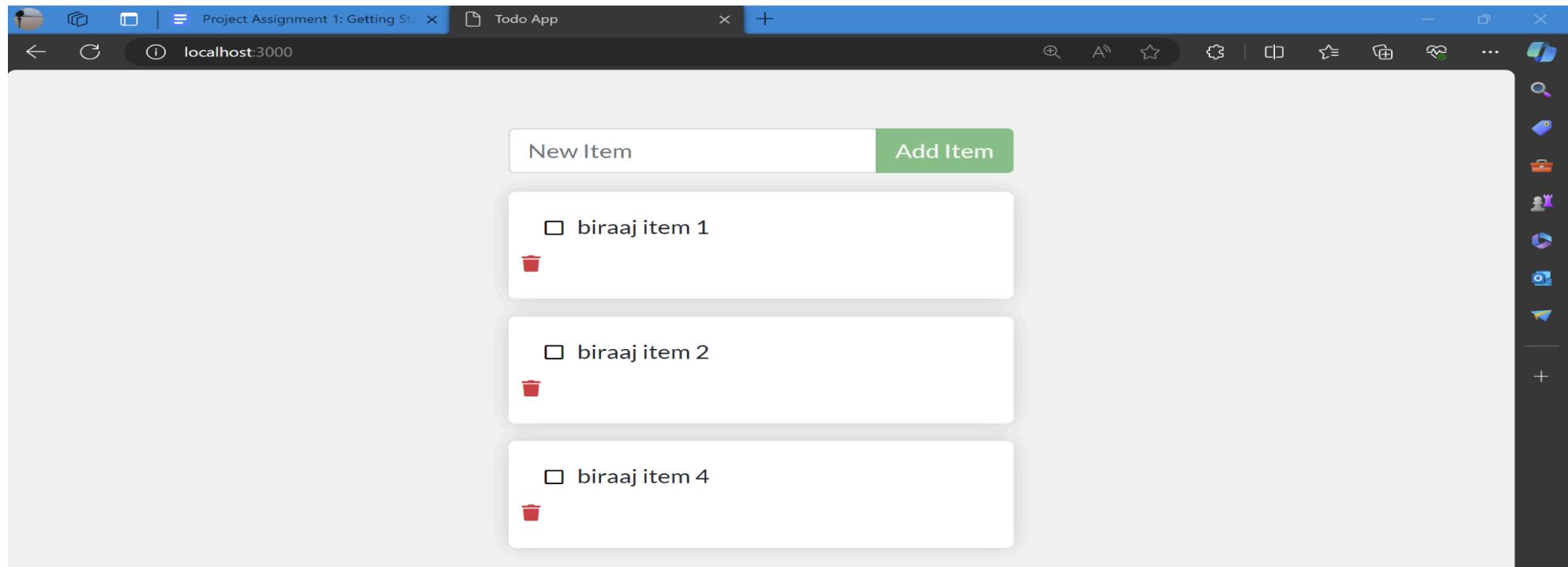
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
1077f6ad92ca	getting-started	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	127.0.0.1:3000->3000/tcp
		gallant_napier			

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker rm -f 1077f6ad92ca
1077f6ad92ca

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -dp 127.0.0.1:3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
d5826b5abb1b93735935b9d80834430de7686ac8d1bb98a2692f323c741060e4

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>[]

3. Checking if the data persists by refreshing the app. We see that same data is visible again after starting the container from scratch again.



v. Checked the volume info

1. The command **docker volume inspect** is used to see where the docker is storing data in host filesystem.

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for file operations, search, and other tools. The Explorer pane on the left displays a project structure under 'FINAL_EXP'. The 'getting-started-app' folder contains 'spec', 'src', 'persistence', 'routes', and 'static' subfolders. Inside 'static' are 'css' and 'js' folders, with 'app.js' being the selected file. Other files visible include 'babel.min.js', 'react-bootstrap.js', 'react-dom.production.min.js', 'react.production.min.js', 'index.html', 'index.js', 'Dockerfile', 'package.json', 'README.md', and 'yarn.lock'. The top navigation bar includes File, Edit, Selection, View, Go, Run, and a three-dot menu. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The status bar at the bottom shows the path 'D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app' and the file name 'final_exp'. The main area is the Terminal pane, which contains the following text:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 17:45:55.59; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker volume inspect
"docker volume inspect" requires at least 1 argument.
See 'docker volume inspect --help'.

Usage: docker volume inspect [OPTIONS] VOLUME [VOLUME...]

Display detailed information on one or more volumes

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker volume inspect todo-db
[
  {
    "CreatedAt": "2024-02-11T23:33:50Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
    "Name": "todo-db",
    "Options": null,
    "Scope": "local"
  }
]
```

g. Part 6: Using bind mounts

i. Trying the bind mounts

1. Start bash in an Ubuntu container

- a. Command: **docker run -it --mount "type=bind,src=%cd%,target=/src" ubuntu bash**

- i. This run command with mount is explained in part 5 and here we added extra parameters src ,target ubuntu and bash were src is current working directory, target is where the directory should appear inside the container and ubuntu bash starts the bash in terminal.
2. Docker starts an interactive bash session in the root directory of the container.
 3. Created a file called **myfile.txt** inside source which is now observed in the getting-started-app directory

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure under 'FINAL_EXP'. The 'getting-started-app' folder contains 'spec', 'src', 'static', 'css', 'js', 'index.html', 'index.js', 'Dockerfile', and 'myfile.txt'. The 'src' folder is expanded, showing 'app.js' selected. The Terminal tab at the top is active, showing the output of a Docker run command. The terminal session shows:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 18:03:58.59; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -it --mount "type=bind,src=%cd%,tar
get=/src" ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
57c139bbda7e: Pull complete
Digest: sha256:e9569c25505f33ff72e88b2990887c9dcf230f23259da296eb814fc2b41af999
Status: Downloaded newer image for ubuntu:latest
root@107a10b35a9f:/# pwd
/
root@107a10b35a9f:/# ls
bin dev home lib32 libx32 mnt proc run src sys usr
boot etc lib lib64 media opt root sbin srv tmp var
root@107a10b35a9f:/# cd src
root@107a10b35a9f:/src# ls
Dockerfile README.md package.json spec src yarn.lock
root@107a10b35a9f:/src# touch myfile.txt
root@107a10b35a9f:/src# ls
Dockerfile README.md myfile.txt package.json spec src yarn.lock
root@107a10b35a9f:/src# 
```

A red box highlights the 'myfile.txt' file in the Explorer and the 'myfile.txt' entry in the terminal's ls output.

4. Deleting the file from the host we saw that the file was gone from the container too and we stopped all interactive sessions using **ctrl+D**

The screenshot shows a Microsoft Visual Studio Code interface. On the left is the Explorer sidebar, which displays a project structure under 'FINAL_EXP'. The 'getting-started-app' folder contains files like 'app.js', 'babel.min.js', 'react-bootstrap.js', etc., and a 'Dockerfile'. The 'TERMINAL' tab is selected at the top, showing a command-line session. The terminal output is as follows:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 18:03:58.59; biraaj

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -it --mount "type=bind,src=%cd%,tar
get=/src" ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
57c139bbda7e: Pull complete
Digest: sha256:e9569c25505f33ff72e88b2990887c9dcf230f23259da296eb814fc2b41af999
Status: Downloaded newer image for ubuntu:latest
root@107a10b35a9f:/# pwd
/
root@107a10b35a9f:/# ls
bin dev home lib32 libx32 mnt proc run  src sys usr
boot etc lib lib64 media opt root sbin srv tmp var
root@107a10b35a9f:/# cd src
root@107a10b35a9f:/src# ls
Dockerfile README.md package.json spec  src  yarn.lock
root@107a10b35a9f:/src# touch myfile.txt
root@107a10b35a9f:/src# ls
Dockerfile README.md myfile.txt package.json spec  src  yarn.lock
root@107a10b35a9f:/src# ls
Dockerfile README.md package.json spec  src  yarn.lock
root@107a10b35a9f:/src#
exit

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>
```

ii. Development containers

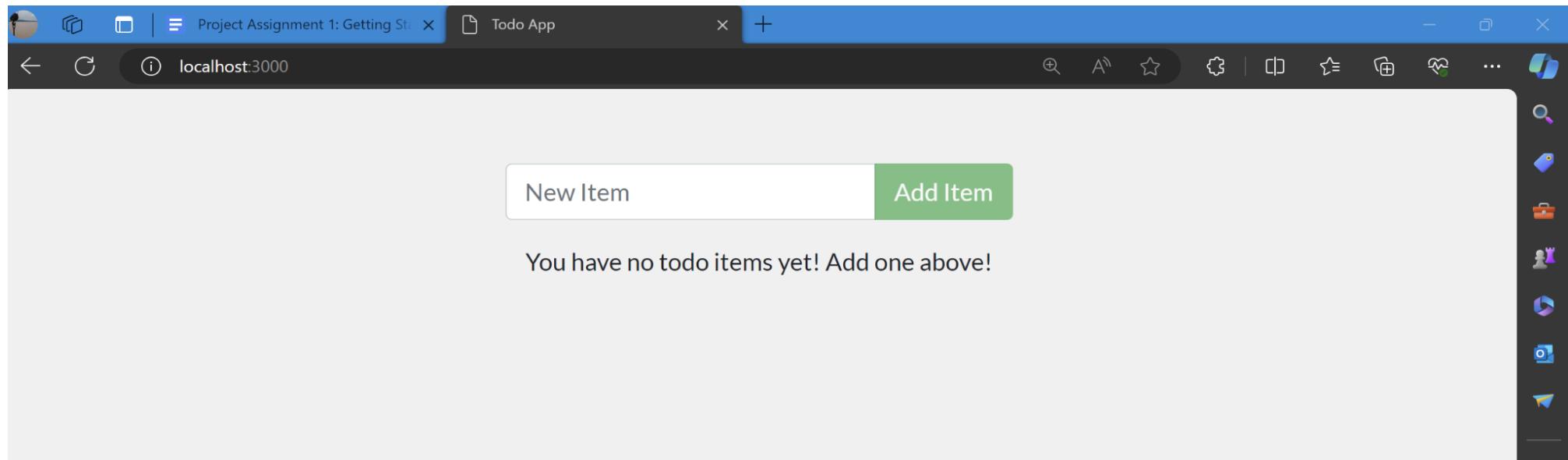
1. Removed all the running containers of getting-started app using the command `docker rm -f <container-id>`
2. Then Ran the command below to initiate a development container with **nodemon**
 - a. Command: `docker run -dp 127.0.0.1:3000:3000 -w /app --mount "type=bind,src=$pwd,target=/app" node:18-alpine sh -c "yarn install && yarn run dev"`

- i. The command is well explained in part2 and 3 . so here **-w /app** - sets the "working directory" or the current directory that the command will run from. **--mount type=bind,src=\$pwd,target=/app**, bind mount the current directory from the host into the /app directory in the container. **node:18-alpine** - the image to use. Note that this is the base image for app from the Dockerfile. **sh -c "yarn install && yarn run dev"** - the command. It is starting a shell using sh (alpine doesn't have bash) and running yarn install to install packages and then running yarn run dev to start the development server. If we look in the package.json, we will see that the dev script starts nodemon.
- b. Watch logs using the command **docker logs <container-id>**
 - i. This command is used to see logs for docker run as we see here the starting of nodemon and sqlite db.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Displays the project structure for "FINAL_EXP". The "js" folder is currently selected.
- Terminal (Bottom):** Shows a command-line session with the following logs:

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> cd ..\getting-started-app\  
PS D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app> whoami;  
wrathgladiator\biraa  
PS D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app> docker run -dp 127.0.0.1:3000 `  
>> -w /app --mount "type=bind,src=$pwd,target=/app"  
>> node:18-alpine `  
>> sh -c "yarn install && yarn run dev"  
0e260feaebe9a78e0533d83b05937b9b08993a5886bf0156621fc3faab2d8b96  
PS D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app> docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS  
NAMES  
0e260feaebe9 node:18-alpine "docker-entrypoint.s..." 24 seconds ago Up 23 seconds 127.0.0.1:3000->3000/tcp  
p charming_mclaren  
PS D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app> docker logs -f 0e260feaebe9  
yarn install v1.22.19  
[1/4] Resolving packages...  
[2/4] Fetching packages...  
[3/4] Linking dependencies...  
[4/4] Building fresh packages...  
Done in 37.48s.  
yarn run v1.22.19  
$ nodemon -L src/index.js  
[nodemon] 2.0.20  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node src/index.js`  
Using sqlite database at /etc/todos/todo.db  
Listening on port 3000  
PS D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>
```

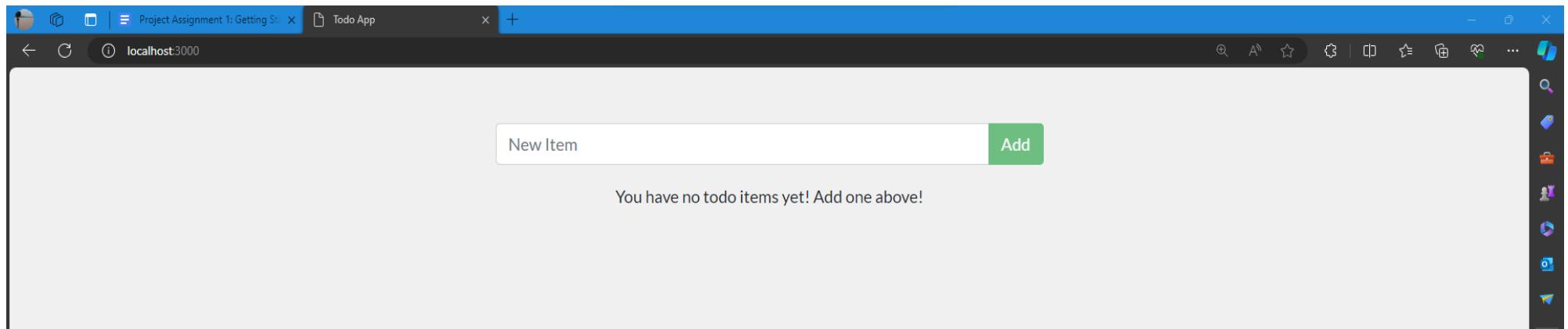


- c. After updating the **app.js** the change is reflected without stopping the container or building a new image.

The screenshot shows a dark-themed interface of the Visual Studio Code editor. On the left is the Explorer sidebar with project files like `getting-started-app`, `src`, and `static`. The main area displays the `app.js` file under the `src` folder. A search bar at the top right contains the query `final_exp`. The code editor shows several lines of JSX code, with line 109 highlighted in red and line 110 highlighted in green. The status bar at the bottom indicates there are 2 of 2 changes in the file.

```
102           <InputGroup.Append>
103             <Button
104               type="submit"
105               variant="success"
106               disabled={!newItem.length}
107               className={submitting ? 'disabled' : ''}
108             >
109               {submitting ? 'Adding...' : 'Add'}
110             <Button>
111               className={submitting ? 'disabled' : ''}
112             >
113             <Form>
114           );
115         }
116       
```

```
117     function ItemDisplay({ item, onItemUpdate, onItemRemoval }) {
```



h. Part7: Multi container apps

- i. Created a container network using command "**docker network create todo-app**":
 1. Here a network called todo-app is created using the above command.
- ii. Ran MySQL container and attached it to the created network
 1. **docker run -d --network todo-app --network-alias mysql -v todo-mysql-data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=secret -e MYSQL_DATABASE=todos mysql:8.0**
 - a. This Docker command runs a MySQL container in detached mode (-d) with the specified configurations:
 - b. It connects the container to a Docker network named "todo-app" with the alias "mysql".
 - c. It mounts a Docker volume named "todo-mysql-data" to persist MySQL data.
 - d. Sets the MySQL root password to "secret" and creates a database named "todos".
 - e. It uses the MySQL 8.0 image to create the container.
 2. **docker exec -it <mysql-container-id> mysql -u root -p**
 - a. This Docker command allows interactive access (-it) to a running MySQL container (<mysql-container-id>) as the root user (-u root) with a prompt for the password (-p).

The screenshot shows the Visual Studio Code interface. The terminal tab is active, displaying the following command-line session:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 20:42:00.85; bira
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker network create todo-app
f972d8d713431721bcd23aa995011e256231aea254b14916fee059739166ca28
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker run -d ^
--network todo-app --network-alias mysql ^
-v todo-mysql-data:/var/lib/mysql ^
-e MYSQL_ROOT_PASSWORD=secret ^
-e MYSQL_DATABASE=todos ^
mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
b8307a22608d: Pull complete
6e74d3ab3202: Pull complete
98cb945026c4: Pull complete
b94f6baa2a82: Pull complete
e6ccb91f8f50: Pull complete
21cd46a2493b: Pull complete
5d1fdb378f69: Pull complete
b33a073ddd89: Pull complete
0cafccc0d406: Pull complete
5a686d059649: Pull complete
7d292e5aace3: Pull complete
Digest: sha256:d848240fd25e2bc1c4f1f3a1f0a0f32582871feb0373dfb8203a52f390120e6f
Status: Downloaded newer image for mysql:8.0
aa4584fa2cc656562f839eec6aa4adb668b64467d058f92e1479da4845ed3b5
```

The file explorer sidebar shows a project structure under 'FINAL_EXP':

- getting-started-app
 - node_modules
 - spec
 - src
 - persistence
 - routes
 - static
 - css
 - js
 - app.js
 - babel.min.js
 - react-bootstrap.js
 - react-dom.production.min.js
 - react.production.min.js
 - index.html
 - index.js- .dockerignore
- Dockerfile
- package.json
- README.md
- yarn.lock
- node_modules
- yarn.lock

The 'app.js' file is selected in the file list.

iii. Checked if the database was running using the commands below.

1. **mysql> SHOW DATABASES;**

- a. This shows all the available databases.

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal output is as follows:

```
Status: Downloaded newer image for mysql:8.0  
aa4584fa2cc656562f839eec6aa4adb668b64467d058f92e1479da4845ed3b5  
  
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
aa4584fa2cc6 mysql:8.0 "docker-entrypoint.s..." 4 minutes ago Up 4 minutes 3306/tcp, 33060/tcp wonderfu_l_pasteur  
  
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker exec -it aa4584fa2cc6 mysql -u root -p  
  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 8  
Server version: 8.0.36 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| todos |  
+-----+  
5 rows in set (0.01 sec)
```

iv. Used **nicolaka/netshoot** container to troubleshoot networking issues.

1. Command1: **docker run -it --network todo-app nicolaka/netshoot**

- a. This Docker command runs the "netshoot" container interactively (-it) within the "todo-app" Docker network, providing a toolbox of networking utilities for troubleshooting and diagnostics.
2. Command2: **dig mysql**
 - a. This command uses dig to perform a DNS lookup for the hostname "mysql".
3. In the "ANSWER SECTION", we see an A record for mysql that resolves to 172.18.0.2 . While mysql isn't normally a valid hostname, Docker was able to resolve it to the IP address of the container that had that network alias as we used the **--network-alias** earlier.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The terminal tab is active, displaying the output of a Docker container named 'final_exp'. The terminal output includes:

```
Digest: sha256:a7c92e1a2fb9287576a16e107166fee7f9925e15d2c1a683dbb1f4370ba9bfe8
Status: Downloaded newer image for nicolaka/netshoot:latest
dP          dP          dP
88          88          88
88d888b. .d8888b. d8888P .d8888b. 88d888b. .d8888b. .d8888b. d8888P
88' `88 88oooood8 88 Y8ooooo. 88' `88 88' `88 88' `88 88
88 88 88. ... 88     88 88 88 88. .88 88. .88 88
dP dP `88888P' dP `88888P' dP dP `88888P' `88888P' dP

Welcome to Netshoot! (github.com/nicolaka/netshoot)
Version: 0.11

afb8fc335ae7 ~ dig mysql
; <>> DiG 9.18.13 <>> mysql
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15705
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;;
;; QUESTION SECTION:
;mysql.                      IN      A
;;
;; ANSWER SECTION:
mysql.                       600    IN      A      172.18.0.2
;;
;; Query time: 20 msec
;; SERVER: 127.0.0.11#53(127.0.0.11) (UDP)
;; WHEN: Mon Feb 12 02:52:57 UTC 2024
;; MSG SIZE rcvd: 44
```

The file explorer on the left shows a project structure for 'FINAL_EXP' containing 'getting-started-app' with 'node_modules', 'spec', 'src' (containing 'persistence', 'routes', 'static' with 'css' and 'js'), and files like 'app.js', 'babel.min.js', etc. It also shows '.dockerignore', 'Dockerfile', 'package.json', 'README.md', 'yarn.lock', and environment variables 'MYSQL_HOST=mysql', 'MYSQL_USER=root', 'MYSQL_PASSWORD=secret', and 'MYSQL_DB=todos'.

4. Started the todo app with MySQL environment variables using the commands below:
 - a. `docker run -dp 127.0.0.1:3000:3000 -w /app -v "%cd%:/app" --network todo-app -e MYSQL_HOST=mysql -e MYSQL_USER=root -e MYSQL_PASSWORD=secret -e MYSQL_DB=todos node:18-alpine sh -c "yarn install && yarn run dev"`

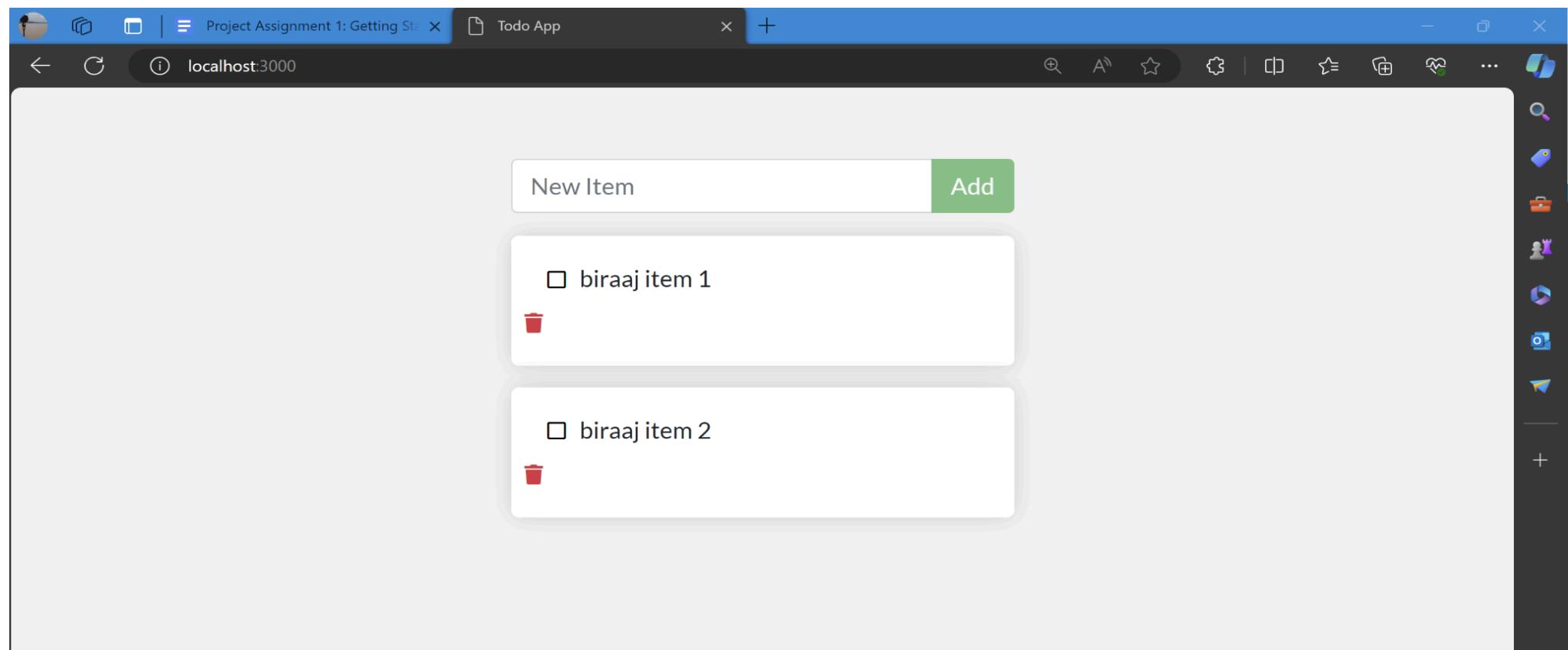
- i. This Docker command runs a Node.js application in detached mode (-d), mapping port 3000 on localhost (127.0.0.1:3000:3000), using the "todo-app" network, and setting environment variables for MySQL connection. It mounts the current directory (%cd%) to /app in the container, sets the working directory to /app, and executes the commands to install dependencies and runs the development server.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** The project structure is displayed under the folder `FINAL_EXP`. It includes a `getting-started-app` folder containing `node_modules`, `spec`, and `src` subfolders. The `src` folder contains `persistence`, `routes`, and `static` subfolders, with `css` and `js` further nested. Specific files shown include `app.js`, `babel.min.js`, `react-bootstrap.js`, `react-dom.production.min.js`, `react.production.min.js`, `index.html`, and `index.js`. There are also `.dockerignore` and `Dockerfile` files.
- TERMINAL:** The terminal tab is active, showing command-line output from the project directory.
 - Output of `echo %date% %time%;`: Sun 02/11/2024 21:05:11.95; biraa
 - Output of `docker run -dp 127.0.0.1:3000 ^` followed by several environment variable assignments (e.g., `-w /app -v "%cd%:/app"`, `-e MYSQL_HOST=mysql`, etc.).
 - Output of `yarn install && yarn run dev`:
6e63dfe09d81eae4900ce6b8a054f564140b910e030406344f0ca756f7c07683
 - Output of `docker ps`:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6e63dfe09d81	node:18-alpine	"docker-entrypoint.s..."	8 seconds ago	Up 7 seconds	127.0.0.1:3000->3000/tcp	quizzical_curiie
aa4584fa2cc6	mysql:8.0	"docker-entrypoint.s..."	22 minutes ago	Up 22 minutes	3306/tcp, 33060/tcp	wonderful_pasteur
 - Output of `docker logs -f 6e63dfe09d81`:
yarn install v1.22.19
[1/4] Resolving packages...
success Already up-to-date.
Done in 0.34s.
yarn run v1.22.19
\$ nodemon -L src/index.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Waiting for mysql:3306.
Connected!
Connected to mysql db at host mysql
Listening on port 3000

5. Adding items to the todo list and checking them in the MySQL database.



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a project structure named "FINAL_EXP" containing "getting-started-app", "node_modules", "spec", "src" (with "persistence", "routes", "static" subfolders), "css", "js" (with "app.js", "babel.min.js", "react-bootstrap.js", "react-dom.production.min.js", "react.production.min.js", "index.html", "index.js", ".dockerignore", "Dockerfile", "package.json", "README.md", "yarn.lock"), and "node_modules" and "yarn.lock" files.
- Terminal:** The active tab shows the following output:

```
yarn install v1.22.19
[1/4] Resolving packages...
success Already up-to-date.
Done in 0.34s.
yarn run v1.22.19
$ nodemon -L src/index.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Waiting for mysql:3306.
Connected!
Connected to mysql db at host mysql
Listening on port 3000
^C
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker exec -it aa4584fa2cc6 mysql -p todos
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from todo_items;
```
- MySQL Data:** The terminal shows the results of the SQL query:

id	name	completed
f8922bfd-e233-44c8-9a96-71fa4179d54f	biraaj item 1	0
66019ed7-002d-4262-8960-6d9ea513b002	biraaj item 2	0

2 rows in set (0.00 sec)

i. Part 8: using docker-compose

- Created a docker-compose file with the following content

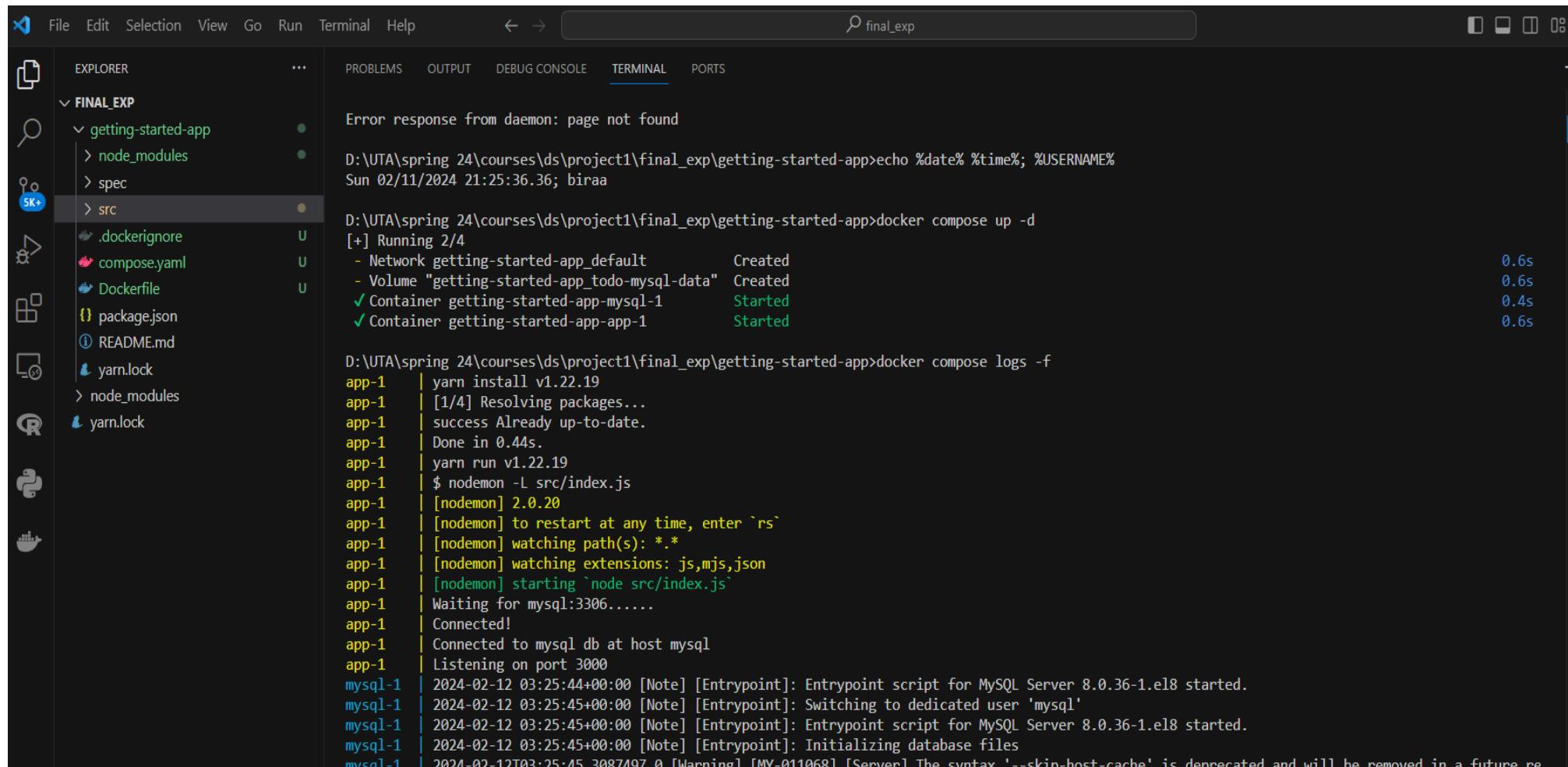
```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos

  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

volumes:
  todo-mysql-data:
```

- ii. Deleted all running containers using “**docker rm -f \$(docker ps -aq)**” to have uninterrupted run. Here **docker ps -aq** lists all the docker container ids that are running.
- iii. Run docker-compose as below and verify it in the logs
 1. Command: **docker compose up -d**
 - a. Starts up the application stack as mentioned in docker compose file.
 - b. Command: **docker compose logs -f**

- i. This shows logs from each of the services interleaved into a single stream. The -f flag follows the log, so will give the live output as it's generated.
- ii. **docker compose logs -f app** helps in viewing the logs for specific service as required.

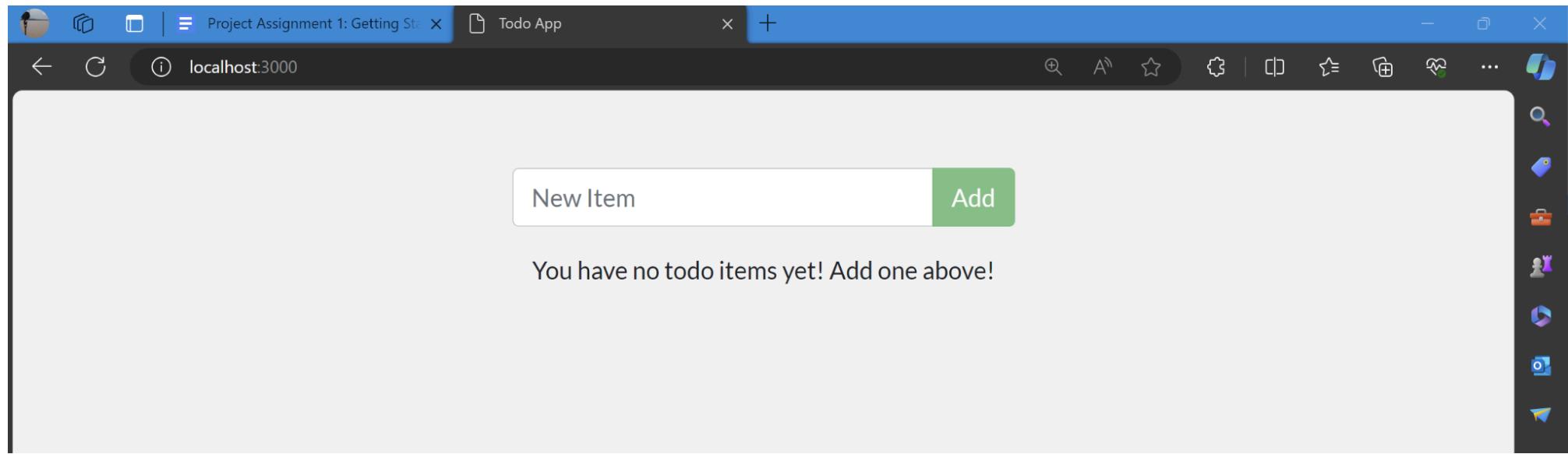


The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the output of a Docker command. The logs show the creation of a network and volumes, and the starting of two containers: 'getting-started-app-mysql-1' and 'getting-started-app-app-1'. The MySQL container logs indicate the entrypoint script for MySQL Server started, switching to the 'mysql' user, and initializing database files. The application container logs show the yarn install process and the nodemon start command.

```

Error response from daemon: page not found
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time% %USERNAME%
Sun 02/11/2024 21:25:36.36; biraa
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker compose up -d
[+] Running 2/4
  - Network getting-started-app_default          Created
  - Volume "getting-started-app_todo-mysql-data" Created
  ✓ Container getting-started-app-mysql-1        Started
  ✓ Container getting-started-app-app-1          Started
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker compose logs -f
app-1 | yarn install v1.22.19
app-1 | [1/4] Resolving packages...
app-1 | success Already up-to-date.
app-1 | Done in 0.44s.
app-1 | yarn run v1.22.19
app-1 | $ nodemon -L src/index.js
app-1 | [nodemon] 2.0.20
app-1 | [nodemon] to restart at any time, enter `rs`
app-1 | [nodemon] watching path(s): *.*
app-1 | [nodemon] watching extensions: js,mjs,json
app-1 | [nodemon] starting `node src/index.js`
app-1 | Waiting for mysql:3306.....
app-1 | Connected!
app-1 | Connected to mysql db at host mysql
app-1 | Listening on port 3000
mysql-1 2024-02-12 03:25:44+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.36-1.el8 started.
mysql-1 2024-02-12 03:25:45+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-1 2024-02-12 03:25:45+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.36-1.el8 started.
mysql-1 2024-02-12 03:25:45+00:00 [Note] [Entrypoint]: Initializing database files
mysql-1 2024-02-12T03:25:45.308Z [Warning] [MY-011068] [Server] The syntax '--skin-host-cache' is deprecated and will be removed in a future re

```



- iv. Destroying all containers using “**docker compose down**”. It kills all services that are running in the application.

The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying several commands and their outputs:

- `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%`
Output: Sun 02/11/2024 21:27:56.96; biraa
- `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps`
Output:

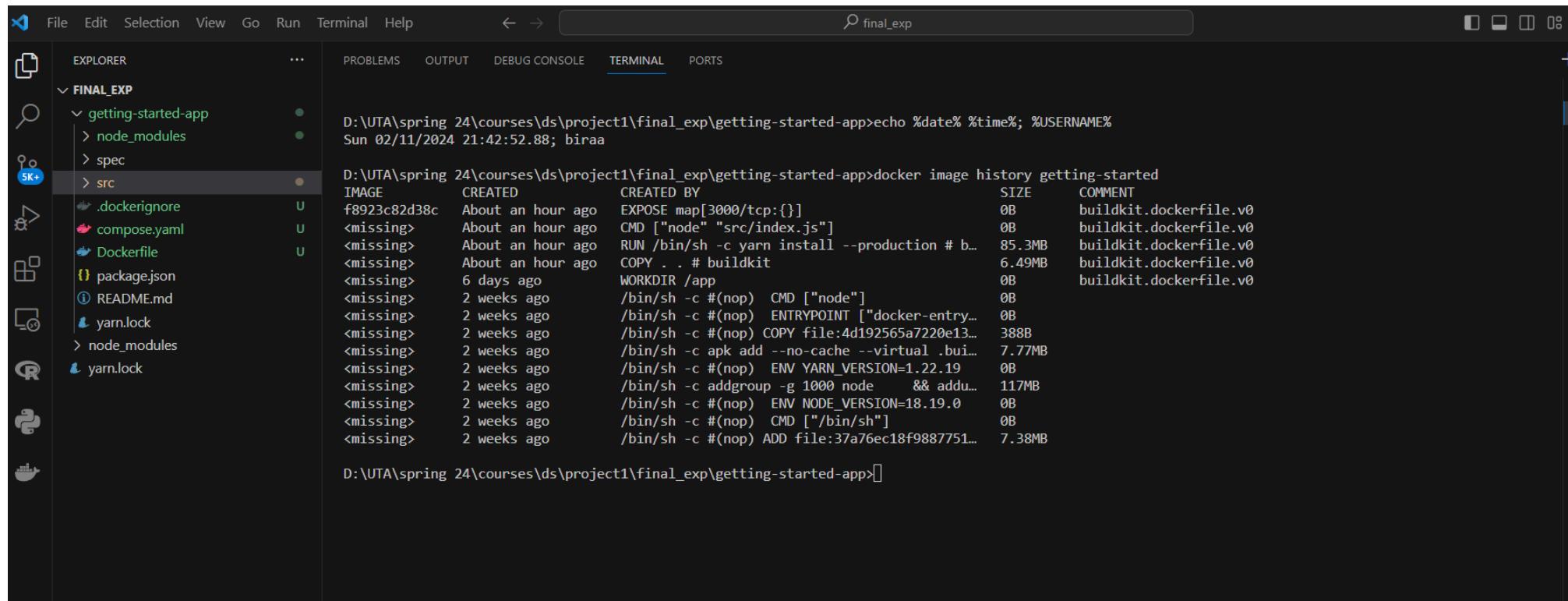
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8e5e3154363c	mysql:8.0	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	3306/tcp, 33060/tcp	getting-started-app-mysql-1
97ed29645dc5	node:18-alpine	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	127.0.0.1:3000->3000/tcp	getting-started-app-app-1
- `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker compose down`
Output: [+] Running 3/3
✓ Container getting-started-app-mysql-1 Removed
✓ Container getting-started-app-app-1 Removed
✓ Network getting-started-app_default Removed
- `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker ps`
Output:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
- `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>`

j. Part 9: image building best practices

i. Image layering

1. Checking history using **docker image history**. Here it shows the layers in the getting-started image.



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command `docker image history getting-started` and its output, which lists several Docker images with their creation details and sizes. The terminal prompt at the bottom is `D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>`.

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
f8923c82d38c	About an hour ago	EXPOSE map[3000/tcp:{}]	0B	buildkit.dockerfile.v0
<missing>	About an hour ago	CMD ["node" "src/index.js"]	0B	buildkit.dockerfile.v0
<missing>	About an hour ago	RUN /bin/sh -c yarn install --production # b...	85.3MB	buildkit.dockerfile.v0
<missing>	About an hour ago	COPY . . # buildkit	6.49MB	buildkit.dockerfile.v0
<missing>	6 days ago	WORKDIR /app	0B	buildkit.dockerfile.v0
<missing>	2 weeks ago	/bin/sh -c #(nop) CMD ["node"]	0B	
<missing>	2 weeks ago	/bin/sh -c #(nop) ENTRYPOINT ["docker-entry...	0B	
<missing>	2 weeks ago	/bin/sh -c #(nop) COPY file:4d19256a7220e13...	388B	
<missing>	2 weeks ago	/bin/sh -c apk add --no-cache --virtual .bui...	7.77MB	
<missing>	2 weeks ago	/bin/sh -c #(nop) ENV YARN_VERSION=1.22.19	0B	
<missing>	2 weeks ago	/bin/sh -c addgroup -g 1000 node && addu...	117MB	
<missing>	2 weeks ago	/bin/sh -c #(nop) ENV NODE_VERSION=18.19.0	0B	
<missing>	2 weeks ago	/bin/sh -c #(nop) CMD ["/bin/sh"]	0B	
<missing>	2 weeks ago	/bin/sh -c #(nop) ADD file:37a76ec18f9887751...	7.38MB	

2. Updating Dockerfile to decrease build time for which we need to support caching of dependencies. So here copying **package.json** first will help stop reinstalling yarn dependencies.

The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. In the top left, there's a navigation bar with File, Edit, Selection, View, Go, Run, Terminal, and Help. To the right of the bar is a search field containing "final_exp". Below the search field is a tab bar with three tabs: "Dockerfile U X" (highlighted in blue), "app.js M", and "compose.yaml U".

The main area is the Explorer sidebar, which lists the project structure under "FINAL_EXP". The structure includes a "getting-started-app" folder containing "node_modules", "spec", "src", ".dockerignore", and "compose.yaml". It also lists "Dockerfile", "package.json", "README.md", "yarn.lock", and two "yarn.lock" files. The "Dockerfile" is currently selected, as indicated by the highlighted row.

The code editor pane displays the content of the "Dockerfile". The code is:

```
1 # syntax=docker/dockerfile:1
2 FROM node:18-alpine
3 WORKDIR /app
4 COPY package.json yarn.lock ./ 
5 RUN yarn install --production
6 COPY . .
7 CMD ["node", "src/index.js"]
```

At the bottom of the screen, there's a "TERMINAL" tab. The terminal window shows two lines of output:

```
D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>echo %date% %time%; %USERNAME%
Sun 02/11/2024 21:42:52.88; biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker image history getting-started
IMAGE          CREATED          CREATED BY
f8923c82d38c  About an hour ago   EXPOSE map[3000/tcp:{}]
<missing>      About an hour ago   CMD ["node" "src/index.js"]
```

3. Used `.dockerignore` file to copy only relevant image files by excluding `node_modules`
4. Built the getting started image once and then made some changes to `index.html` and then built it again.
So we see many files being cached during the second build.

The screenshot shows the VS Code interface with the terminal tab selected. The terminal output displays the Docker build process for a React application named 'getting-started-app'. The build command used was `docker build -t getting-started .`. The log shows the following steps:

```

D:\UTA\spring 24\courses\ds\project1\final_exp\getting-started-app>docker build -t getting-started .
[+] Building 13.7s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 205B
=> resolve image config for docker.io/docker/dockerfile:1
=> [auth] docker/dockerfile:pull token for registry-1.docker.io
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load .dockerignore
=> => transferring context: 53B
=> [1/5] FROM docker.io/library/node:18-alpine
=> [internal] load build context
=> => transferring context: 5.24kB
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package.json yarn.lock ./
=> [4/5] RUN yarn install --production
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:b242be5c2d4289b7a90ac06580e0125522acb4eb003e712b30d2028d31d1ec46
=> => naming to docker.io/library/getting-started

View build details: docker-desktop://dashboard/build/default/default/194tjs1yx86qyo7exqpn7y3j3

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

```

The build completed successfully with a total duration of 13.7 seconds. The Dockerfile used is located at `Dockerfile`, and the resulting image is tagged `getting-started`.

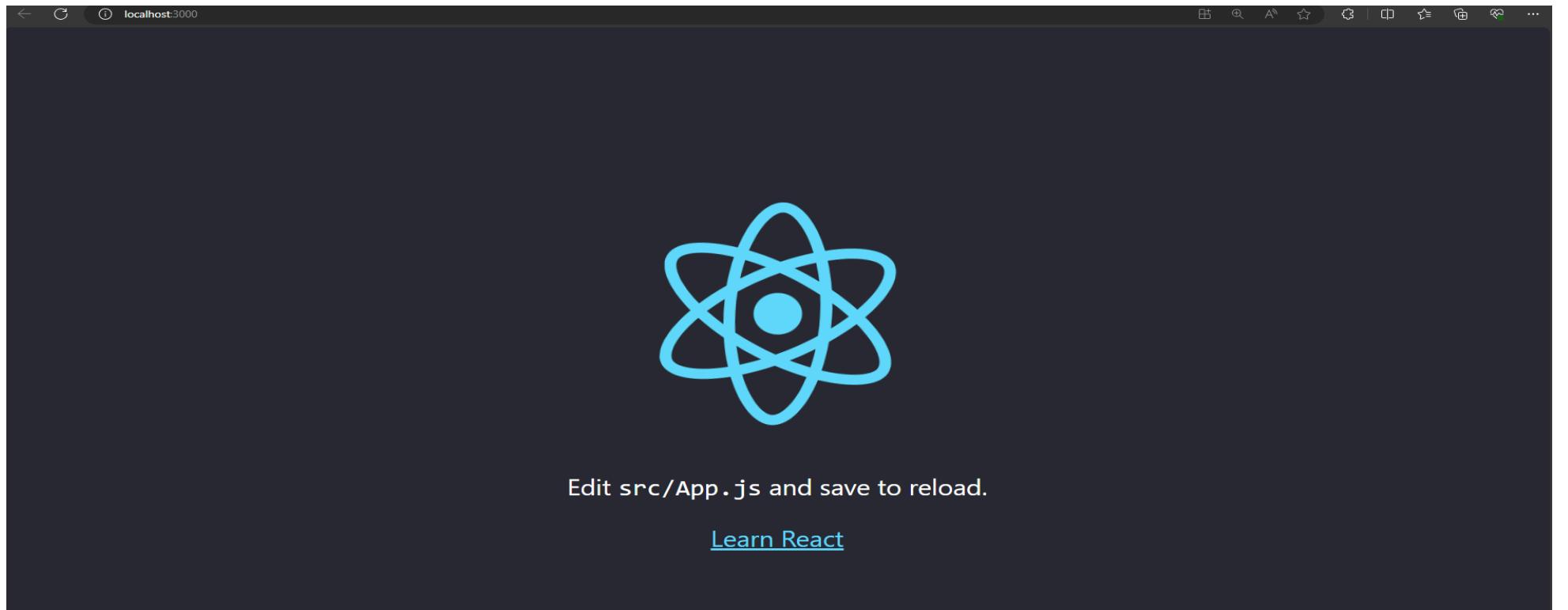
Below the terminal, there are two additional build logs for the same directory, both completed in 0.8 seconds. The first one corresponds to the build shown above, and the second one is identical.

ii. Multi-stage builds

1. React example

- a. Created a react app using command “**yarn create react-app my-app**” as this create a new react app called my-app and started it using “**npm start**” (ref:[Getting Started | Create React App \(create-react-app.dev\)](#))

```
C:\Windows\system32\cmd.exe: X + ^ D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn>yarn --version  
1.22.21  
  
D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn>yarn create react-app my-app  
yarn create v1.22.21  
[1/4] Resolving packages...  
warning create-react-app > tar-pack > tar@2.2.2: This version of tar is no longer supported, and will not receive security updates. Please upgrade asap.  
[2/4] Fetching packages...  
[3/4] Linking dependencies...  
[4/4] Building fresh packages...  
success Installed "create-react-app@5.0.1" with binaries:  
- create-react-app  
  
Creating a new React app in D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn\my-app.  
  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts with cra-template...
```



- i. Created a docker file as below referring to the tutorial for react apps in docker for Multi-stage builds

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a file tree for a project named 'FINAL_EXP'. The tree includes a 'getting-started-app' folder, a 'react_app' folder, and a 'react-app-yarn\my-app' folder which contains 'node_modules', 'public', and 'src' subfolders, along with files like '.dockerignore', '.gitignore', 'Dockerfile', 'package.json', 'README.md', and 'yarn.lock'. The 'Dockerfile' file is currently selected in the editor area on the right. The code in the editor is as follows:

```
1 # syntax=docker/dockerfile:1
2 FROM node:18 AS build
3 WORKDIR /app
4 COPY package* yarn.lock ./
5 RUN yarn install
6 COPY public ./public
7 COPY src ./src
8 RUN yarn run build
9
10 FROM nginx:alpine
11 COPY --from=build /app/build /usr/share/nginx/html
```

- ii. Created build, Ran the image container and the output is shown below.
1. As specified in Dockerfile above. Here we ship the static resources in a static nginx container as we are not doing any server side rendering.

```
C:\Windows\system32\cmd.e: × + ▾
Compiled successfully!

You can now view my-app in the browser.

Local:          http://localhost:3000
On Your Network:  http://10.5.2.4:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully

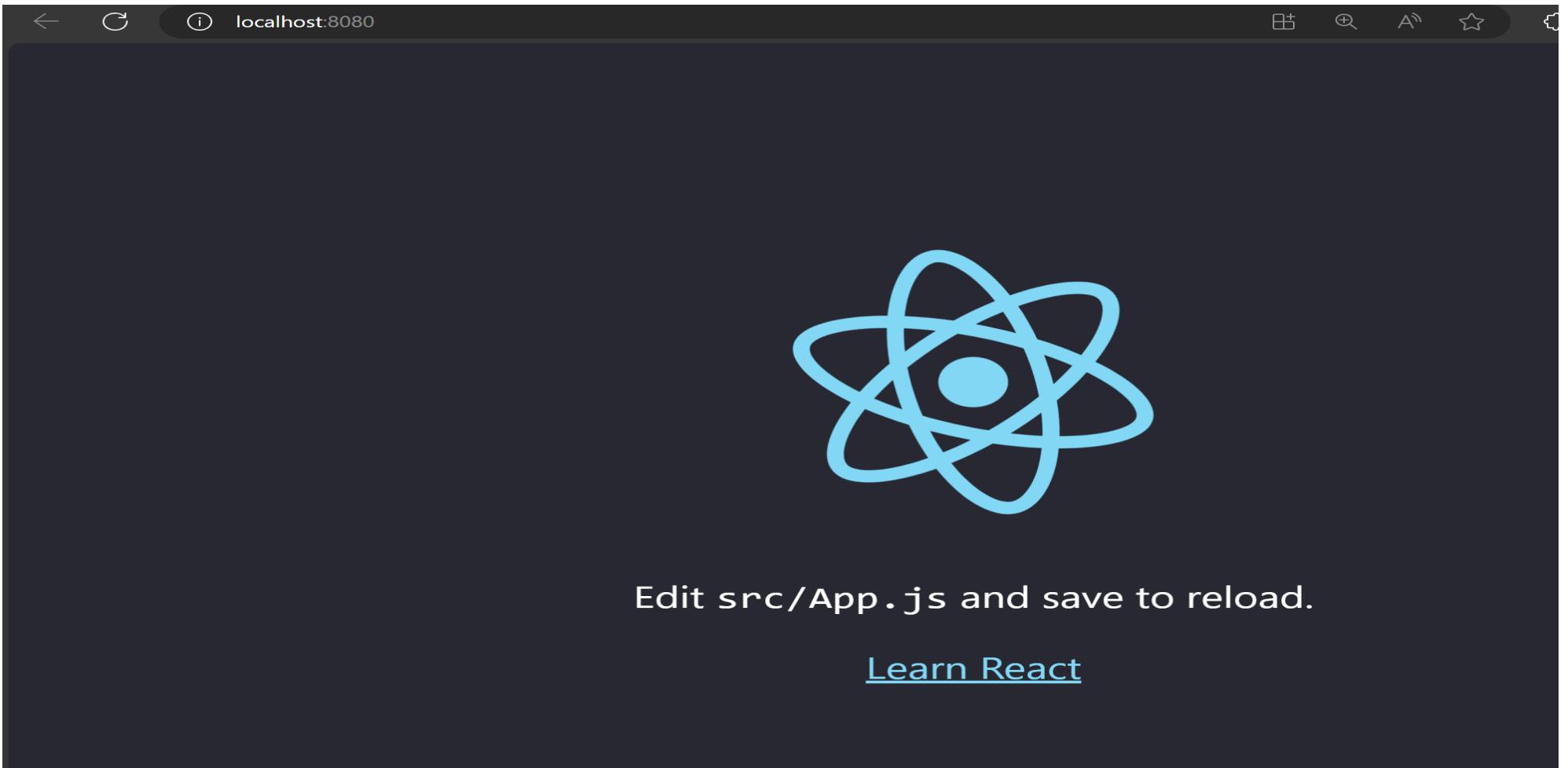
D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn\my-app>docker build -t my-react-app:latest .
[+] Building 60.0s (20/20) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 281B
=> resolve image config for docker.io/docker/dockerfile:1
=> [auth] docker/dockerfile:pull token for registry-1.docker.io
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
=> [internal] load metadata for docker.io/library/node:18
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 52B
=> CACHED [stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:f2802c2a9d09c7aa3ace27445dfc5656ff24355da28e7b958074a0111e3fc076
=> [build 1/7] FROM docker.io/library/node:18@sha256:2a13079c6393cd19adfd8d362fac004b2d0eed462f3c3fedfad2c0d0de17b429
=> [internal] load build context
=> => transferring context: 498.56kB
=> CACHED [build 2/7] WORKDIR /app
=> [build 3/7] COPY package* yarn.lock ./
=> [build 4/7] RUN yarn install
=> [build 5/7] COPY public ./public
=> [build 6/7] COPY src ./src
=> [build 7/7] RUN yarn run build
=> [stage-1 2/2] COPY --from=build /app/build /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:eb4eeefe24ff8977c4200c31d21e1d6935711f86d6e57e0f42cc7601df10ae176
=> => naming to docker.io/library/my-react-app:latest

View build details: docker-desktop://dashboard/build/default/default/8xdwyu9pbqwg9dn7hmt1clszj

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

D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn\my-app>docker run -d -p 8080:80 my-react-app:latest
e2aba0ed16e9056f5b2deb416c4c59d9e915ea33dc1eaf2afbc6a6d493e06b29

D:\UTA\spring 24\courses\ds\project1\final_exp\react-app-yarn\my-app>
```



2. Python language-specific guide

a. Description:

- i. Here we are building images, running containers and developing an app using python and flask.

b. Containerize a Python application:

- i. Cloned the same application from git using “git clone <https://github.com/docker/python-docker>” and Intialized docker application using “docker init”

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** final_exp
- Terminal Tab:** TERMINAL
- Terminal Content:**

```
D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app>echo %date% %time% %USERNAME%
Mon 02/12/2024 7:07:08.78 biraa

D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app>git clone https://github.com/docker/python-docker
Cloning into 'python-docker'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 11 (delta 2), reused 1 (delta 1), pack-reused 5
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.

D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app>docker init
Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
- Dockerfile
- compose.yaml
- README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.5)

? What version of Python do you want to use? 3.11.5
? What port do you want your app to listen on? (8000) 5000

? What port do you want your app to listen on? 5000
? What is the command you use to run your app? (gunicorn 'python-docker.app:app' --bind=0.0.0.0:5000) python3 -m fl
? What is the command you use to run your app? python3 -m flask run --host=0.0.0.0

CREATED: .dockerignore
CREATED: Dockerfile
```
- Explorer View:** Shows a file tree with the following structure:
 - FINAL_EXP
 - > getting-started-app
 - > python-docker-app
 - > python-docker
 - app.py
 - README.md
 - requirements.txt
 - .dockerignore
 - compose.yaml
 - Dockerfile
 - README.Docker.md
 - > react_app
 - > react-app-yarn
- Sidebar:** Includes icons for GitHub, Git, Python, Docker, and other development tools.
- Bottom Status Bar:** OUTLINE, TIMELINE

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar contains icons for Explorer, Search, Problems, and other extensions like Python and Docker. The Explorer view shows a project structure under 'FINAL_EXP': 'getting-started-app' (expanded) and 'python-docker-app' (selected). Inside 'python-docker-app', there are files: 'app.py', 'README.md', 'requirements.txt', '.dockerignore', 'compose.yaml', 'Dockerfile', and 'README.Docker.md'. The main terminal tab is active, displaying the following output:

```
- Dockerfile
- compose.yaml
- README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.5)

? What version of Python do you want to use? 3.11.5
? What port do you want your app to listen on? (8000) 5000

? What port do you want your app to listen on? 5000
? What is the command you use to run your app? (gunicorn 'python-docker.app:app' --bind=0.0.0.0:5000) python3 -m fl
? What is the command you use to run your app? python3 -m flask run --host=0.0.0.0

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✓ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

WARNING: No requirements.txt file found. Be sure to create one that contains the dependencies for your application before running it.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:5000

Consult README.Docker.md for more information about using the generated files.
```

ii. Building the docker image and running it using docker compose

1. **docker compose up --build** (explained in part 8)
 - a. Here --build flag triggers rebuilding images for services with changes in Dockerfiles
2. **docker compose down**(explained in part 8)

```

Mon 02/12/2024 7:25:37.91 biraa;

(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app\python-docker>docker compose up --build
[+] Building 0.0s (0/0) docker:default
[+] Building 20.5s (14/14) FINISHED
      docker:default
-> [server internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 1.68kB                            0.0s
=> [server] resolve image config for docker.io/docker/dockerfile:1  0.9s
=> [server auth] docker/dockerfile:pull token for registry-1.docker.io 0.0s
=> CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50 0.0s
=> [server internal] load metadata for docker.io/library/python:3.11.4-slim 0.9s
=> [server auth] library/python:pull token for registry-1.docker.io 0.0s
=> [server internal] load .dockerignore 0.0s
=> => transferring context: 667B 0.0s
=> [server base 1/5] FROM docker.io/library/python:3.11.4-slim@sha256:17d62d681d9ecef20aae6c6605e9cf83b0ba3dc247013e2f43 12.5s
=> => sha256:17d62d681d9ecef20aae6c6605e9cf83b0ba3dc247013e2f43e1b5a045ad4901 1.65kB / 1.65kB 0.0s
=> => sha256:0275089b5b654bb33931fc239a447db9fdd1628bc9d1482788754785d6d9e464 1.37kB / 1.37kB 0.0s
=> => sha256:596e0d6b34dfa7ed330941075bcd38b376b3eba8e5b63a1da38bf04fe08bdd3 6.92kB / 6.92kB 0.0s
=> => sha256:52d2b7f179e32b4cbd579ee3c4958027988f9a8274850ab0c7c24661e3adaac5 29.12MB / 29.12MB 3.6s
=> => sha256:2b8a9a2240c1224b34f6aafbc3310f9a3fe65bd6893050906d02e89fc8326aa9 3.50MB / 3.50MB 0.3s
=> => sha256:051d6521462a7eb4ca0374e97701d6eec68eb51b118d3ef5d002798b498fb12e 17.86MB / 17.86MB 3.1s
=> => sha256:fce84b1f897c621e9474bd4d5a49e2e22fa35e248e78e754010d34ec3d2d28cd 245B / 245B 0.6s
=> => sha256:46233543d8c2dc599bdb9d522180ca9e14cad4ac2017a5dc481660bfa4aa3ed9 3.38MB / 3.38MB 2.2s
=> => extracting sha256:52d2b7f179e32b4cbd579ee3c4958027988f9a8274850ab0c7c24661e3adaac5 3.4s
=> => extracting sha256:2b8a9a2240c1224b34f6aafbc3310f9a3fe65bd6893050906d02e89fc8326aa9 0.4s
=> => extracting sha256:051d6521462a7eb4ca0374e97701d6eec68eb51b118d3ef5d002798b498fb12e 3.5s
=> => extracting sha256:fce84b1f897c621e9474bd4d5a49e2e22fa35e248e78e754010d34ec3d2d28cd 0.0s
=> => extracting sha256:46233543d8c2dc599bdb9d522180ca9e14cad4ac2017a5dc481660bfa4aa3ed9 1.1s
=> [server internal] load build context 0.0s
=> => transferring context: 1.20kB 0.0s
=> [server base 2/5] WORKDIR /app 0.1s
=> [server base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" 0.8s
=> [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target= 4.7s
=> [server base 5/5] COPY . . 0.1s
=> [server] exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:0cb6d854b909db819f7deedb7a69814dd3f1bbcdc59b2381dff3d2625bcd2093 0.0s
=> => naming to docker.io/library/python-docker-server 0.0s
[+] Running 2/2
  ✓ Network python-docker_default Created 0.1s
  ✓ Container python-docker-server-1 Created 0.1s
Attaching to server-1
server-1 | * Debug mode: off

```



Hello, Docker!

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS    cmd ▲ + ▾ ⊞ ... ▾ ×  
=> [server internal] load build context  
=> => transferring context: 1.20kB  
=> [server base 2/5] WORKDIR /app  
=> [server base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin"  
=> [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target=  
=> [server base 5/5] COPY . .  
=> [server] exporting to image  
=> => exporting layers  
=> => writing image sha256:0cb6d854b909db819f7deedb7a69814dd3f1bbcd59b2381dff3d2625bcd2093  
=> => naming to docker.io/library/python-docker-server  
[+] Running 2/2  
✓ Network python-docker_default Created 0.0s  
✓ Container python-docker-server-1 Created 0.0s  
Attaching to server-1 0.1s  
server-1 | * Debug mode: off 0.1s  
server-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. 0.0s  
server-1 | * Running on all addresses (0.0.0.0) 0.0s  
server-1 | * Running on http://127.0.0.1:5000 0.0s  
server-1 | * Running on http://172.20.0.2:5000 0.0s  
server-1 | Press CTRL+C to quit 0.0s  
server-1 | 172.20.0.1 - - [12/Feb/2024 13:28:25] "GET / HTTP/1.1" 200 - 0.0s  
server-1 | 172.20.0.1 - - [12/Feb/2024 13:28:25] "GET /favicon.ico HTTP/1.1" 404 - 0.0s  
Gracefully stopping... (press Ctrl+C again to force)  
[+] Stopping 1/1 0.0s  
✓ Container python-docker-server-1 Stopped 10.5s  
canceled 0.0s  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app\python-docker>docker compose down 0.0s  
[+] Running 2/2 0.0s  
✓ Container python-docker-server-1 Removed 0.0s  
✓ Network python-docker_default Removed 0.3s  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-app\python-docker>
```

c. Develop your App

i. Use containers for Python development

1. Cloned the sample application from “git clone <https://github.com/docker/python-docker-dev>”
2. Initialized the docker files with “docker init”. This command Initialize a project with the files necessary to run the project in a container.

The screenshot shows the Visual Studio Code interface with the terminal tab active. The search bar at the top contains "final_exp". The terminal window displays the following command-line session:

```
(base) D:\UTA\spring 24\courses\ds\project1\final_exp>echo %date% %time% %USERNAME%;  
Mon 02/12/2024 7:40:38.77 biraa;  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp>git clone https://github.com/docker/python-docker-dev  
Cloning into 'python-docker-dev'...  
remote: Enumerating objects: 18, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (18/18), done.  
remote: Total 18 (delta 5), reused 6 (delta 0), pack-reused 0  
Receiving objects: 100% (18/18), 5.63 KiB | 5.63 MiB/s, done.  
Resolving deltas: 100% (5/5), done.  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp>cd python-docker-dev  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev>dir  
Volume in drive D is personal  
Volume Serial Number is E047-5B0D  
  
Directory of D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev  
  
02/12/2024 07:40 AM <DIR> .  
02/12/2024 07:40 AM <DIR> ..  
02/12/2024 07:40 AM 1,532 app.py  
02/12/2024 07:40 AM 121 README.md  
02/12/2024 07:40 AM 157 requirements.txt  
3 File(s) 1,810 bytes  
2 Dir(s) 47,398,903,808 bytes free
```

```
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev>docker init
Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
- Dockerfile
- compose.yaml
- README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.5) 3.11.4

? What version of Python do you want to use? 3.11.4
? What port do you want your app to listen on? (8000) 5000

? What port do you want your app to listen on? 5000
? What is the command you use to run your app? (gunicorn 'app:app' --bind=0.0.0.0:5000) python3 -m flask run --host=0.0.0.0

? What is the command you use to run your app? python3 -m flask run --host=0.0.0.0

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✓ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:5000

Consult README.Docker.md for more information about using the generated files.
```

ii. Add a local database and persist data

1. Updating the **compose.yaml** file to have all the database instructions
2. Before running the application we created db folder and a file called **password.txt** to store our password which is “**mysecretpaswword**”

compose.yaml U X password.txt U

```
python-docker-dev > compose.yaml > {} secrets > {} db-password > file
    docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications (compose-spec.json)
1  services:
2    server:
3      build:
4        context: .
5      ports:
6        - 5000:5000
7      environment:
8        - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
9      depends_on:
10        db:
11          condition: service_healthy
12        secrets:
13          - db-password
14    db:
15      image: postgres
16      restart: always
17      user: postgres
18      secrets:
19        - db-password
20      volumes:
21        - db-data:/var/lib/postgresql/data
22      environment:
23        - POSTGRES_DB=example
24        - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
25      expose:
26        - 5432
27      healthcheck:
28        test: [ "CMD", "pg_isready" ]
29        interval: 10s
30        timeout: 5s
31        retries: 5
32      volumes:
33        db-data:
34      secrets:
35        db-password:
36          file: db/password.txt
```

3. After we build and run the docker using “**docker compose up –build**” and try out the below commands and get empty response as our database is empty.
 - a. curl <http://localhost:5000/initdb>
 - b. curl <http://localhost:5000/widgets>
 - c. Curl is a command line tool that enables data exchange between a device and a server through a terminal.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
Mon 02/12/2024 7:50:32.49 biraa;

(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev>docker compose up --build
[+] Running 15/1
✓ db 14 layers [██████████] 0B/0B Pulled
  20.2s
[+] Building 9.2s (14/14) FINISHED
  docker:default
    => [server internal] load build definition from Dockerfile
      0.0s
    => => transferring dockerfile: 1.68kB
      0.0s
    => [server] resolve image config for docker.io/docker/dockerfile:1
      0.7s
    => [server auth] docker/dockerfile:pull token for registry-1.docker.io
      0.0s
    => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
      0.0s
    => [server internal] load metadata for docker.io/library/python:3.11.4-slim
      0.5s
    => [server auth] library/python:pull token for registry-1.docker.io
      0.0s
    => [server internal] load .dockerignore
      0.0s
    => => transferring context: 667B
      0.0s
    => [server base 1/5] FROM docker.io/library/python:3.11.4-slim@sha256:17d62d681d9ecef20aae6c6605e9cf83b0ba3dc247013e2f43e1b5a045ad4901
      0.0s
    => [server internal] load build context
      0.0s
    => => transferring context: 2.73kB
      0.0s
    => CACHED [server base 2/5] WORKDIR /app
      0.0s
    => CACHED [server base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --no-create-home
      - 0.0s
    => [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target=requirements.txt python -m pip ins 7.1s
    => [server base 5/5] COPY . .
      0.1s
    => [server] exporting to image
      0.1s
    => => exporting layers
      0.1s
    => => writing image sha256:dcd0b42b35c96953596b427c09f80c56149d0f5d3f5af68041115eeeb8f8f0ad
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> whoami;
wrathgladiator\biraa
PS D:\UTA\spring 24\courses\ds\project1\final_exp> curl http://localhost:5000/initdb

    StatusCode : 200
    StatusDescription : OK
    Content : init database
    RawContent : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 13
                  Content-Type: text/html; charset=utf-8
                  Date: Mon, 12 Feb 2024 13:52:01 GMT
                  Server: Werkzeug/2.3.6 Python/3.11.4

          init database
Forms : {}
Headers : {[Connection, close], [Content-Length, 13], [Content-Type, text/html; charset=utf-8], [Date, Mon, 12 Feb 2024 13:52:01 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 13

PS D:\UTA\spring 24\courses\ds\project1\final_exp> curl http://localhost:5000/widgets

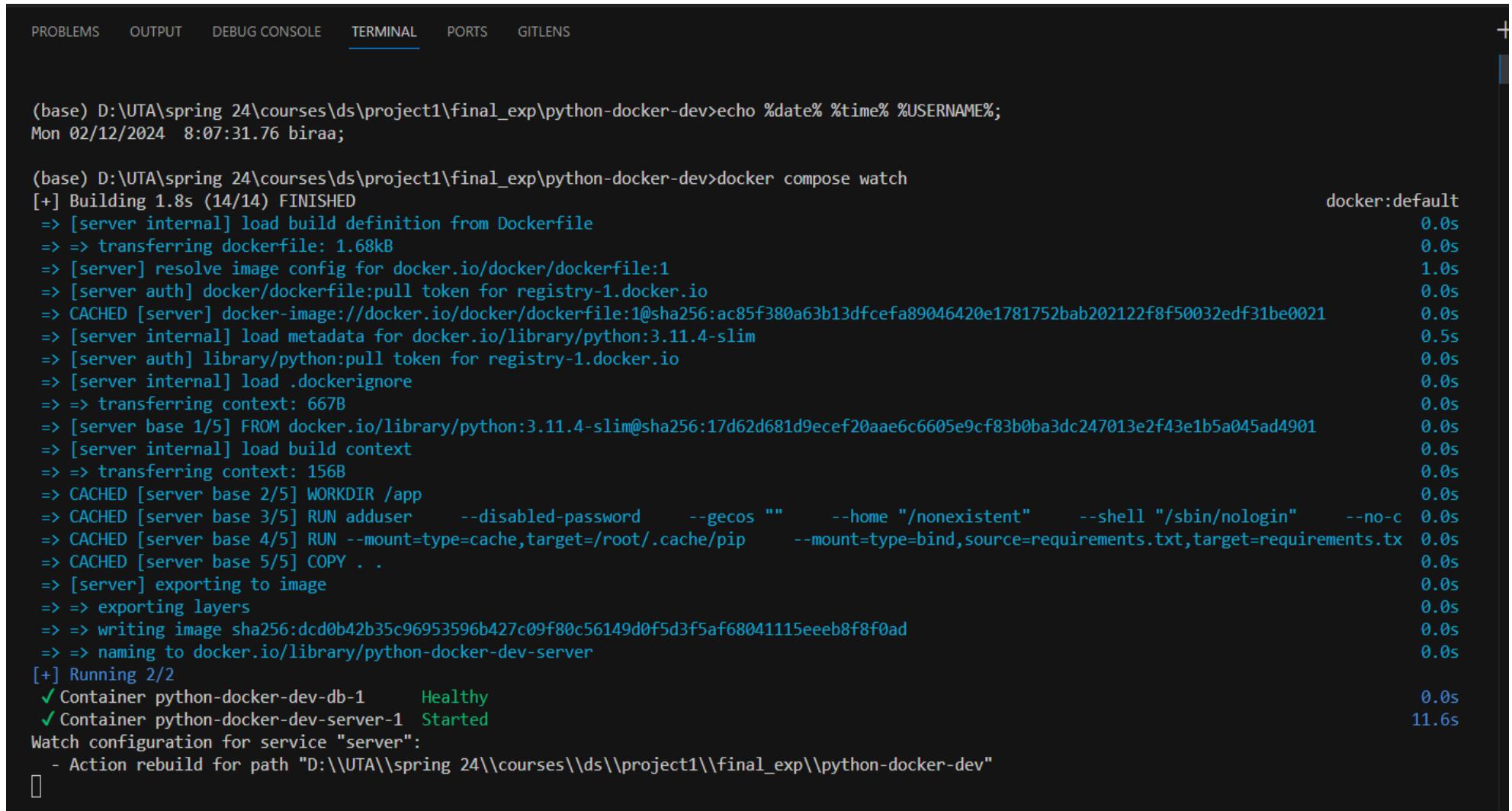
    StatusCode : 200
    StatusDescription : OK
    Content : []
    RawContent : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 2
                  Content-Type: text/html; charset=utf-8
                  Date: Mon, 12 Feb 2024 13:52:36 GMT
                  Server: Werkzeug/2.3.6 Python/3.11.4

        []
Forms : {}
Headers : {[Connection, close], [Content-Length, 2], [Content-Type, text/html; charset=utf-8], [Date, Mon, 12 Feb 2024 13:52:36 GMT]...}
Images : {}
InputFields : {}
Links : {}
```

4. We then stop the application using **ctrl+c**

iii. Automatically update services

- Used compose watch to automatically update running compose services as we edit and save our code. To enable that we modified **compose.yaml** and then run “**docker compose watch**” to watch for changes.



The screenshot shows a terminal window within a code editor interface. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), PORTS, and GITLENS. The terminal output is as follows:

```
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev>echo %date% %time% %USERNAME%;  
Mon 02/12/2024 8:07:31.76 biraa;  
  
(base) D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev>docker compose watch  
[+] Building 1.8s (14/14) FINISHED docker:default  
=> [server internal] load build definition from Dockerfile 0.0s  
=> => transferring dockerfile: 1.68kB 0.0s  
=> [server] resolve image config for docker.io/docker/dockerfile:1 1.0s  
=> [server auth] docker/dockerfile:pull token for registry-1.docker.io 0.0s  
=> CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021 0.0s  
=> [server internal] load metadata for docker.io/library/python:3.11.4-slim 0.5s  
=> [server auth] library/python:pull token for registry-1.docker.io 0.0s  
=> [server internal] load .dockerignore 0.0s  
=> => transferring context: 667B 0.0s  
=> [server base 1/5] FROM docker.io/library/python:3.11.4-slim@sha256:17d62d681d9ecef20aae6c6605e9cf83b0ba3dc247013e2f43e1b5a045ad4901 0.0s  
=> [server internal] load build context 0.0s  
=> => transferring context: 156B 0.0s  
=> CACHED [server base 2/5] WORKDIR /app 0.0s  
=> CACHED [server base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --no-c 0.0s  
=> CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target=requirements.tx 0.0s  
=> CACHED [server base 5/5] COPY . . 0.0s  
=> [server] exporting to image 0.0s  
=> => exporting layers 0.0s  
=> => writing image sha256:dcd0b42b35c96953596b427c09f80c56149d0f5d3f5af68041115eeeb8f8f0ad 0.0s  
=> => naming to docker.io/library/python-docker-dev-server 0.0s  
[+] Running 2/2 0.0s  
✓ Container python-docker-dev-db-1 Healthy 0.0s  
✓ Container python-docker-dev-server-1 Started 11.6s  
Watch configuration for service "server":  
- Action rebuild for path "D:\\UTA\\spring 24\\courses\\ds\\project1\\final_exp\\python-docker-dev"  
[]
```

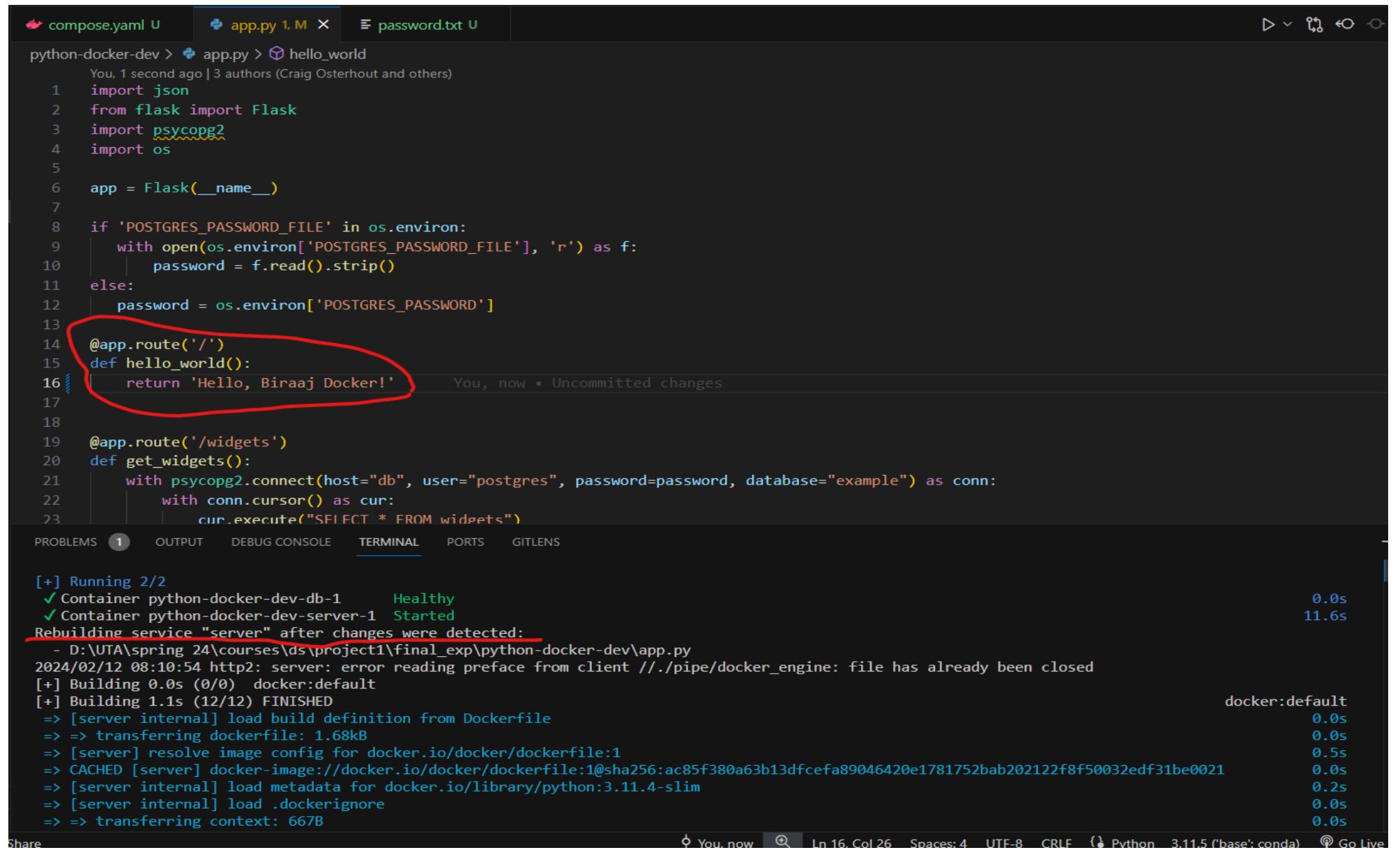
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> whoami;
wrathgladiator\biraa
PS D:\UTA\spring 24\courses\ds\project1\final_exp> curl http://localhost:5000
```

```
StatusCode      : 200
StatusDescription : OK
Content         : Hello, Docker!
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 14
                  Content-Type: text/html; charset=utf-8
                  Date: Mon, 12 Feb 2024 14:09:10 GMT
                  Server: Werkzeug/2.3.6 Python/3.11.4

                  Hello, Docker!
Forms           : {}
Headers         : {[Connection, close], [Content-Length, 14], [Content-Type, text/html; charset=utf-8], [Date, Mon, 12 Feb 2024 14:09:10 GMT]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 14
```

2. After changing the **app.py** we see that the docker compose builds the image again because docker watch is running and we see the modified response when we hit the “[curl http://localhost:5000](http://localhost:5000)”



The screenshot shows a code editor interface with three tabs: compose.yaml, app.py (marked with a blue plus icon), and password.txt. The app.py tab is active, displaying Python code for a Flask application. A red oval highlights the first few lines of the hello_world function.

```
python-docker-dev > app.py > hello_world
You, 1 second ago | 3 authors (Craig Osterhout and others)
1 import json
2 from flask import Flask
3 import psycopg2
4 import os
5
6 app = Flask(__name__)
7
8 if 'POSTGRES_PASSWORD_FILE' in os.environ:
9     with open(os.environ['POSTGRES_PASSWORD_FILE'], 'r') as f:
10         password = f.read().strip()
11 else:
12     password = os.environ['POSTGRES_PASSWORD']
13
14 @app.route('/')
15 def hello_world():
16     return 'Hello, Biraaj Docker!'
17
18
19 @app.route('/widgets')
20 def get_widgets():
21     with psycopg2.connect(host="db", user="postgres", password=password, database="example") as conn:
22         with conn.cursor() as cur:
23             cur.execute("SELECT * FROM widgets")

```

The terminal below shows the output of a docker-compose command, indicating that two containers are running: db (healthy) and server (started). It also shows the process of rebuilding the server service after changes were detected, listing various Docker build steps with their durations.

```
[+] Running 2/2
✓ Container python-docker-dev-db-1      Healthy          0.0s
✓ Container python-docker-dev-server-1 Started        11.6s
Rebuilding service "server" after changes were detected:
- D:\UTA\spring 24\courses\ds\project1\final_exp\python-docker-dev\app.py
2024/02/12 08:10:54 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 0.0s (0/0) docker:default
[+] Building 1.1s (12/12) FINISHED
=> [server internal] load build definition from Dockerfile          docker:default
=> => transferring dockerfile: 1.68kB                                0.0s
=> [server] resolve image config for docker.io/docker/dockerfile:1    0.5s
=> CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021 0.0s
=> [server internal] load metadata for docker.io/library/python:3.11.4-slim   0.2s
=> [server internal] load .dockerignore                                0.0s
=> => transferring context: 667B                                    0.0s
```

PROBLEMS

1

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

GITLENS

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> whoami;
```

```
wrathgladiator\biraa
```

```
PS D:\UTA\spring 24\courses\ds\project1\final_exp> curl http://localhost:5000
```

```
StatusCode      : 200
StatusDescription : OK
Content         : Hello, Biraaj Docker!
RawContent      : HTTP/1.1 200 OK
                  Connection: close
                  Content-Length: 21
                  Content-Type: text/html; charset=utf-8
                  Date: Mon, 12 Feb 2024 14:13:26 GMT
                  Server: Werkzeug/2.3.6 Python/3.11.4

                  Hello, Biraaj Docker!
Forms          : {}
Headers        : {[Connection, close], [Content-Length, 21], [Content-Type, text/html; charset=utf-8], [Date, Mon, 12 Feb 2024 14:13:26 GMT]...}
Images         : {}
InputFields    : {}
Links          : {}
ParsedHtml     : mshtml.HTMLDocumentClass
RawContentLength : 21
```

References:

1. <https://docs.docker.com/get-started/overview/>
2. <https://docs.docker.com/language/python/>
3. [Git Guides - git clone \(github.com\)](#)
4. <https://docs.docker.com/engine/reference/builder/>
5. <https://create-react-app.dev/docs/getting-started>
6. <https://docs.docker.com/engine/reference/commandline>
7. <https://blog.hubspot.com/website/curl-command>