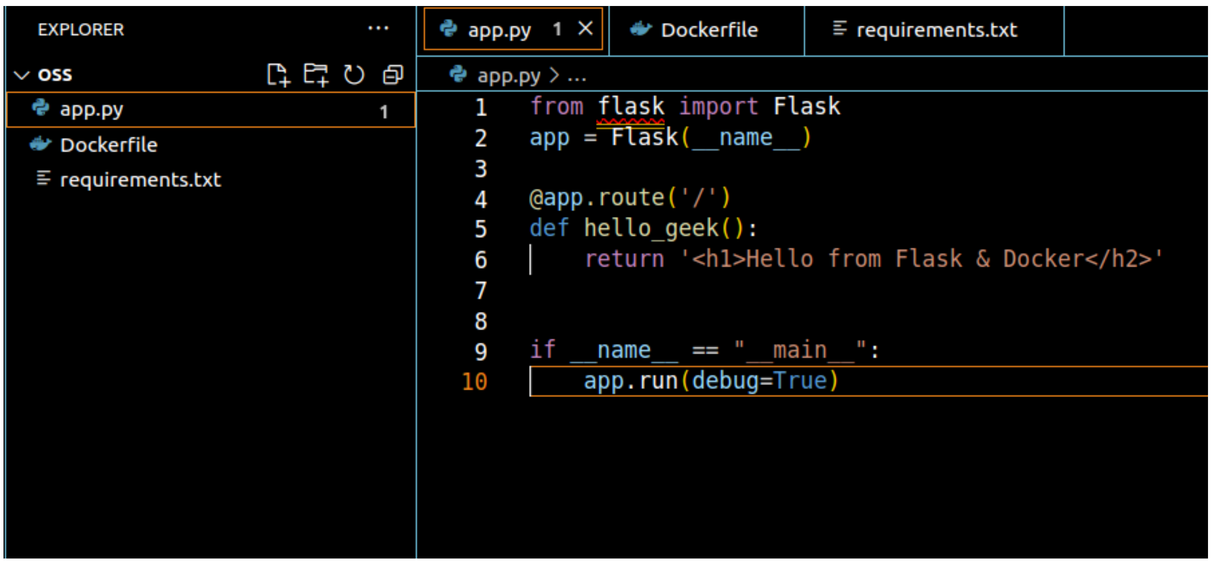
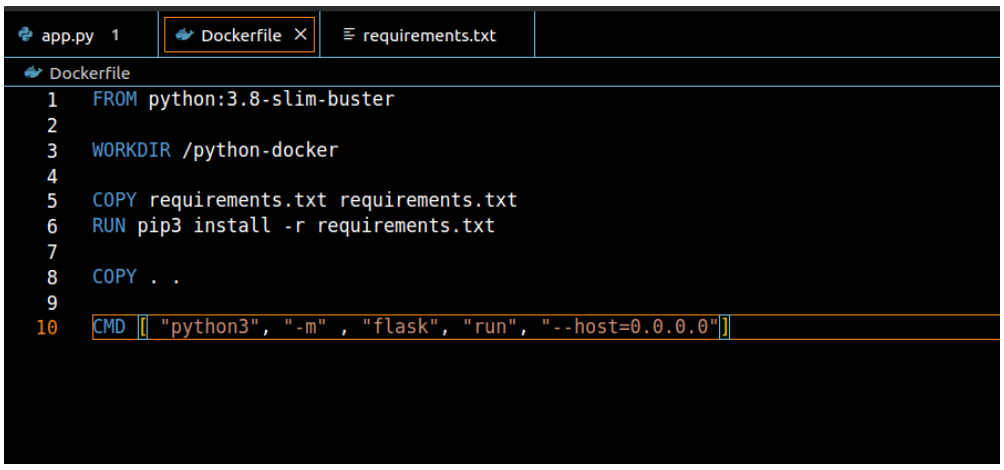
Q.24 A. Create a simple Hello-world python flask application and create the docker image of that Flask application.

B. Run the docker container from recently created image and run that docker container to 5000 port of host system.



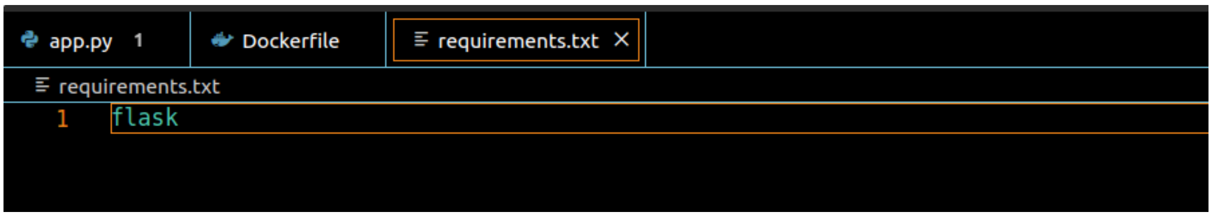
```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_geek():
    return '<h1>Hello from Flask & Docker</h2>'


if __name__ == "__main__":
    app.run(debug=True)
```

```python
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_geek():
    return '<h1>Hello from Flask & Docker</h2>'
if __name__ == "__main__":

    app.run(debug=True)
```



```dockerfile
FROM python:3.8-slim-buster

WORKDIR /python-docker

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

```
syntax=docker/dockerfile:1
FROM python:3.8-slim-buster
WORKDIR /python-docker
COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

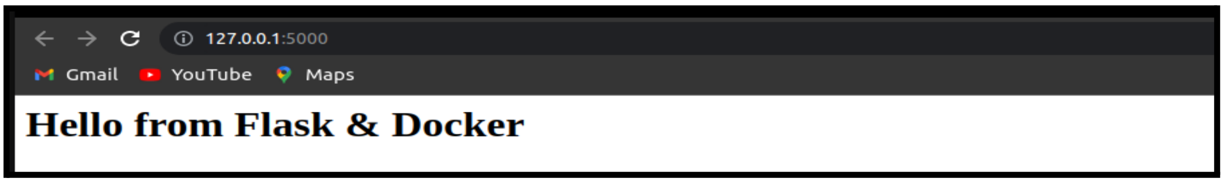


```
flask
```

sudo docker build -t hp .

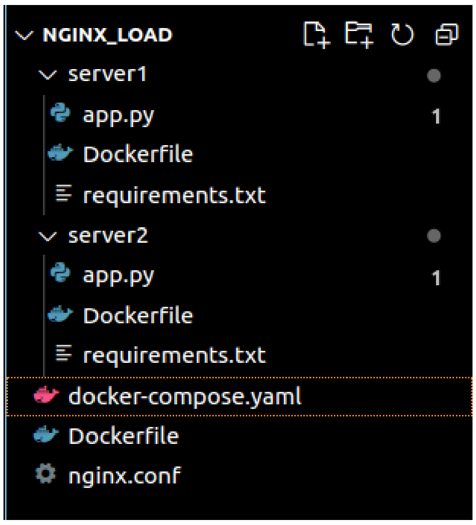
sudo docker run -p 5000:5000 -it hp

Output:



Hello from Flask & Docker

Q.25 Create the 'nginx' container from 'nginx' image. And create the load balancing so that if we go to the address of 'nginx ' it can redirect it to the above created applications (Flask and Wordpress).



Create nginx_load folder
Copy the OSS folder(2 times) from above question
Name 1st copied OSS folder as server1
Name 2nd copied OSS folder as server2

Code for server1(app.py):

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_geek():
   return '<h1>Hello from server1</h2>'
if __name__ == "__main__":
   app.run(debug=True)
```

Code for server2(app.py):

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_geek():
   return '<h1>Hello from Server2</h2>'
if __name__ == "__main__":
   app.run(debug=True)
```

Create docker-compose.yaml in nginx_load:

```
version: '3'

services:
 nginx:
   image: nginx-load-balancer
   ports:
     - "8080:80"
   depends_on:
     - flask-app-1
     - flask-app-2

 flask-app-1:
   image: server1
   ports:
     - "5000:5000"
   # Add other necessary configuration for your Flask app 1

 flask-app-2:
   image: server2
   container_name: my-flask-app-2
   ports:
     - "8000:5000"
   # Add other necessary configuration for your Flask app 2
```

Create Dockerfile in nginx_load:

```
FROM nginx

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Create nginx.conf in nginx_load:

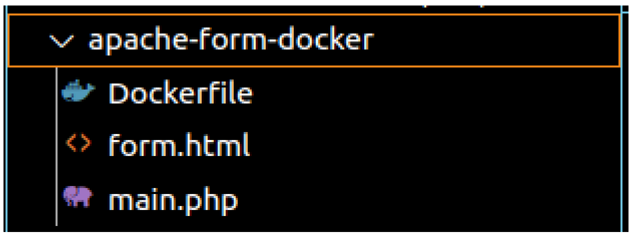
```
# nginx.conf

user  nginx;
worker_processes  1;
```

```
events {
   worker_connections  1024;
}

http {
   include       /etc/nginx/mime.types;
   default_type  application/octet-stream;

   sendfile        on;
   keepalive_timeout  65;

   upstream backend {
       server flask-app-1:5000;
       server flask-app-2:5000;
       # Add more backend servers as needed
   }

   server {
       listen 80;

       location / {
           proxy_pass http://backend;
           proxy_set_header Host $host;
           proxy_set_header X-Real-IP $remote_addr;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
       }
   }
}
```

cd server1
sudo docker build -t server1 .
cd ..
cd server2
sudo docker build -t server2 .
sudo docker images
cd ..
sudo docker build -t nginx-load-balancer .
sudo docker compose up

localhost:8080

Q. 26 Create a web application with simple web page containing login details and create a docker image of the application.(Use Apache Web server)
Q.27 Run the Docker container from recently created image and run the container at port number 80 in host system.



```
apache-form-docker
    Dockerfile
    form.html
    main.php
```

Dockerfile:

```
# Use an official PHP image with Apache as a base
FROM php:apache

# Copy the HTML and PHP files into the document root
COPY form.html /var/www/html/
COPY main.php /var/www/html/

# Expose port 80 for incoming HTTP traffic
EXPOSE 80
```

form.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Login Form</title>
   <style>
       body {
           font-family: Arial, sans-serif;
           background-color: #f4f4f4;
           text-align: center;
```

```
            margin-top: 50px;
        }

        .login-form {
            width: 300px;
            margin: 0 auto;
            background: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        label {
            display: block;
            margin-bottom: 8px;
        }

        input {
            width: 100%;
            padding: 8px;
            margin-bottom: 15px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        input[type="submit"] {
            background-color: #4caf50;
            color: white;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="login-form">
        <h2>Login</h2>
        <form method="post" action="main.php">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"
required>

            <input type="submit" value="Login">
        </form>
    </div>
</body>
</html>
```

main.php:

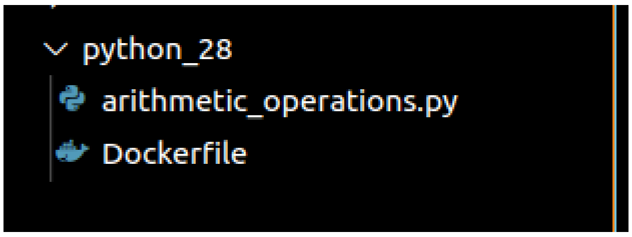
```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // Retrieve the submitted username and password
   $username = $_POST["username"];
   $password = $_POST["password"];

   // Validate the username and password (replace this with your
authentication logic)
   if ($username === "Harshali" && $password === "1234") {
       echo "Login successful!";
   } else {
       echo "Login failed. Please check your username and password.";
   }
}
?>
```

sudo docker build -t harry .
sudo docker run -p 8080:80 harry

localhost:8080/form.html

Q.28 Write a python program to perform arithmetic operations and create Docker image accordingly.

Q.29 Run the Docker container with created image .



```
python_28
    arithmetic_operations.py
    Dockerfile
```

arithmetic_operations.py:

```python
# arithmetic_operations.py

def add(x, y):
   return x + y

def subtract(x, y):
   return x - y

def multiply(x, y):
   return x * y

def divide(x, y):
   if y != 0:
       return x / y
   else:
       return "Error: Division by zero"
if __name__ == "__main__":
   num1 = float(input("Enter the first number: "))
   num2 = float(input("Enter the second number: "))
   print(f"Addition: {add(num1, num2)}")
   print(f"Subtraction: {subtract(num1, num2)}")
   print(f"Multiplication: {multiply(num1, num2)}")
   print(f"Division: {divide(num1, num2)}")
```

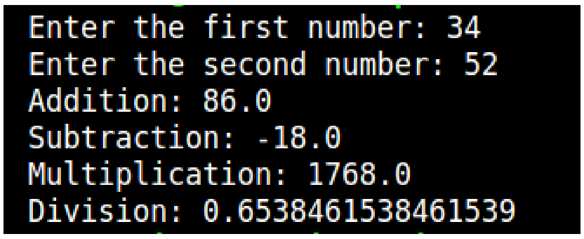
Dockerfile:

```dockerfile
# Dockerfile
# Use an official Python runtime as a parent image
FROM python:3.8-slim
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Run arithmetic_operations.py when the container launches
CMD ["python", "./arithmetic_operations.py"]
```

sudo docker build -t arithmetic_operations_app .

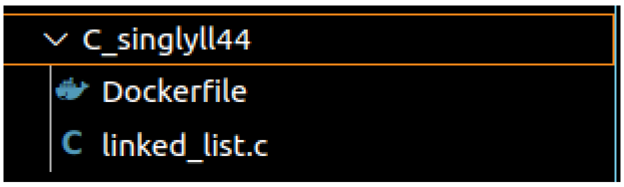
sudo docker run -it arithmetic_operations_app

Output:



```
Enter the first number: 34
Enter the second number: 52
Addition: 86.0
Subtraction: -18.0
Multiplication: 1768.0
Division: 0.6538461538461539
```

Q.44 Write a C program to create singly linked list and containerize it.



linked_list.c:

```c
#include <stdio.h>
#include <stdlib.h>

// Define a structure for a node in the linked list
struct Node {
   int data;
   struct Node* next;
};

// Function to insert a new node at the end of the linked list
void insertEnd(struct Node** head, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct
Node));
   newNode->data = value;
   newNode->next = NULL;

   if (*head == NULL) {
       *head = newNode;
       return;
   }

   struct Node* last = *head;
   while (last->next != NULL) {
       last = last->next;
   }

   last->next = newNode;
}

// Function to print the linked list
void printList(struct Node* node) {
   while (node != NULL) {
       printf("%d -> ", node->data);
       node = node->next;
```

```
    }
    printf("NULL\n");
}

int main() {
    // Initialize an empty linked list
    struct Node* head = NULL;

    // Insert elements into the linked list
    insertEnd(&head, 10);
    insertEnd(&head, 20);
    insertEnd(&head, 30);

    // Print the linked list
    printf("Linked List: ");
    printList(head);

    return 0;
}
```

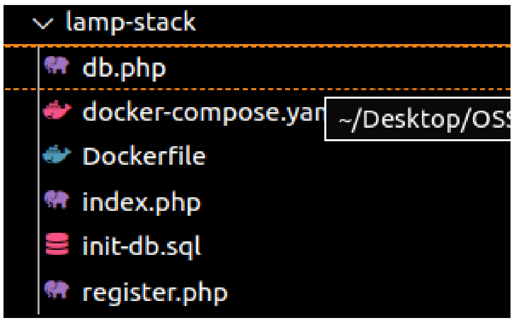
Dockerfile:

```
# Use an official gcc image as a parent image
FROM gcc:latest
# Set the working directory to /app
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Compile the C program
RUN gcc -o linked_list linked_list.c
# Run the executable when the container launches
CMD ["./linked_list"]
```

sudo docker build -t linked_list_app .
sudo docker run -it linked_list_app

Output:
Linked List: 10 -> 20 -> 30 -> NULL

Q. 30 Create a simple web application using LAMP Stack on docker container.
Q.35 Pull the LAMP Stack container from docker hub and host a web application of your own.


```
lamp-stack
  db.php
  docker-compose.yam  ~/Desktop/OS
  Dockerfile
  index.php
  init-db.sql
  register.php
```

db.php:

```
<?php
$conn = mysqli_connect('mysql', 'sample_user', 'sample_password',
'sample_db');
if (!$conn) {
   die('Could not connect: ' . mysqli_connect_error());
}
```

docker-compose.yaml:

```
version: '3'

services:
 lamp-app:
   build:
     context: .
   image: my-lamp-app
   ports:
     - "8080:80"
   depends_on:
     - mysql
   environment:
     MYSQL_HOST: mysql
     MYSQL_ROOT_PASSWORD: root_password
     MYSQL_DATABASE: sample_db
     MYSQL_USER: sample_user
     MYSQL_PASSWORD: sample_password
 mysql:
   image: mysql
   mem_limit: 1000m
   environment:
     MYSQL_ROOT_PASSWORD: root_password
     MYSQL_DATABASE: sample_db
     MYSQL_USER: sample_user
     MYSQL_PASSWORD: sample_password
   volumes:
     - ./init-db.sql:/docker-entrypoint-initdb.d/init-db.sql
```

Dockerfile:

```
# Use an official PHP image with Apache as a base
FROM php:apache
# Install MySQLi extension
RUN docker-php-ext-install mysqli
# Copy PHP files into the document root
COPY index.php /var/www/html/
COPY db.php /var/www/html/
COPY register.php /var/www/html/
# Expose port 80 for incoming HTTP traffic
EXPOSE 80
```

index.php:

```
<?php
include 'db.php';
$result = mysqli_query($conn, 'SELECT * FROM sample_table');
?>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>LAMP Stack Example</title>
</head>
<body>
   <h1>LAMP Stack Example</h1>

   <table border="1">
       <tr>
           <th>ID</th>
           <th>Name</th>
```

```
        </tr>
        <?php while ($row = mysqli_fetch_assoc($result)) : ?>
            <tr>
                <td><?php echo $row['id']; ?></td>
                <td><?php echo $row['name']; ?></td>
            </tr>
        <?php endwhile; ?>
    </table>

    <br>

    <h2>Add Name</h2>
    <form method="POST" action="register.php">
        <label for="new_name">New Name:</label>
        <input type="text" id="new_name" name="new_name" required>
        <input type="submit" value="Add Name">
    </form>
</body>
</html>
```

init-db.sql:

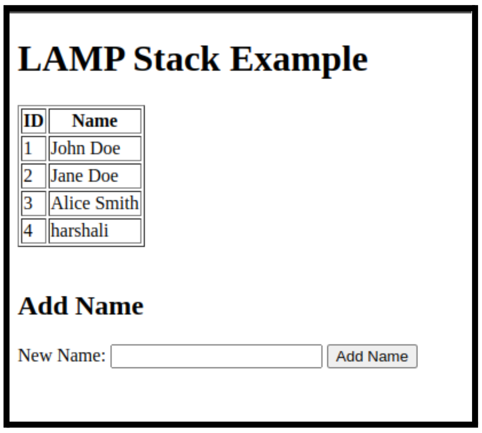
```
-- init-db.sql

CREATE TABLE IF NOT EXISTS sample_table (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(255) NOT NULL
);

INSERT INTO sample_table (name) VALUES
   ('John Doe'),
   ('Jane Doe'),
   ('Alice Smith');
```

register.php:

```
<?php
include 'db.php';

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $newName = mysqli_real_escape_string($conn, $_POST['new_name']);
   // Insert new name into the existing table
   $insertQuery = "INSERT INTO sample_table (name) VALUES
('$newName')";

   if (mysqli_query($conn, $insertQuery)) {
       header("Location: index.php"); // Redirect to index.php after
successful addition
       exit();
   } else {
       echo "Error: " . $insertQuery . "<br>" . mysqli_error($conn);
   }
}
mysqli_close($conn);
?>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Add Name</title>
</head>
<body>
   <!-- No need for additional content here, as it redirects to
index.php -->
</body>
</html>
```

sudo docker compose up

localhost:8080



LAMP Stack Example

ID	Name
1	John Doe
2	Jane Doe
3	Alice Smith
4	harshali

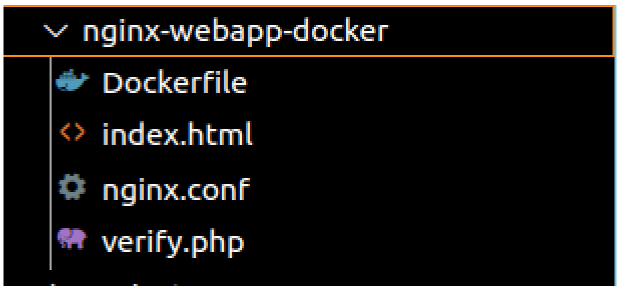
Add Name

New Name:

Q.31 Create a web application with simple web page containing login details and create a docker image of the application.(Use Ngnix Web server)

Q.32 Run the Docker container from recently created image and run the container at port number 80 in host system.

(yat error yeto so ha expt aala trr nginx-webapp-mysql wala karaycha)



Dockerfile:

```
FROM nginx:latest

COPY nginx.conf /etc/nginx/nginx.conf
COPY index.html /usr/share/nginx/html/index.html
COPY verify.php /usr/share/nginx/html/verify.php
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Login Form</title>
</head>
<body>
   <form action="/verify.php" method="post">
       <label for="username">Username:</label>
       <input type="text" id="username" name="username"
required><br>

       <label for="password">Password:</label>
       <input type="password" id="password" name="password"
required><br>
```

```
       <input type="submit" value="Login">
    </form>
</body>
</html>
```

nginx.conf:

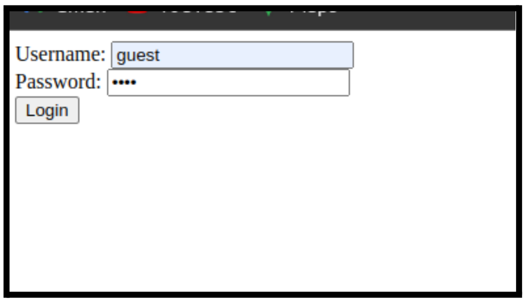
```
events {
    # Configuration for events section (e.g., worker_connections)
}

http {
    # Configuration for HTTP section (e.g., server blocks, location
blocks)
    server {
        listen 80;
        server_name localhost;

        location / {
            root /usr/share/nginx/html;
            index index.html;
        }

        location ~ \.php$ {
            include fastcgi_params;
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root /usr/share/nginx/html;
        }
    }
}
```

verify.php:

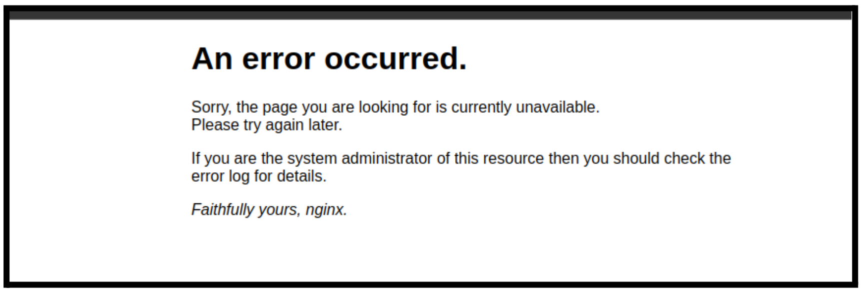
```
<?php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    // Retrieve username and password from the POST request
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Replace the following condition with your actual
verification logic
    if ($username === "example" && $password === "password") {
        echo "User verified!";
    } else {
        echo "Invalid credentials. Please try again.";
    }
} else {
    // Handle non-POST requests if needed
    echo "Invalid request method.";
}
?>
```

sudo docker build -t vir .
sudo docker run -p 8000:80 -it vir

localhost:8000



Username: guest
Password: ••••
Login



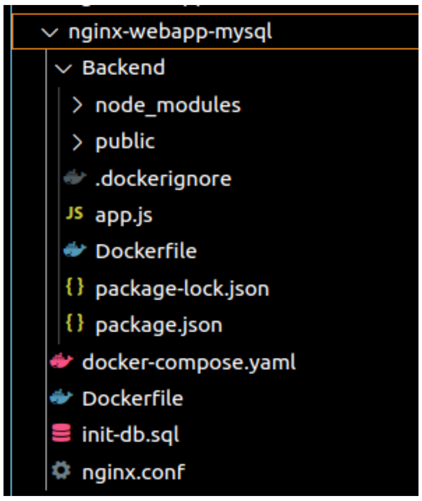
An error occurred.

Sorry, the page you are looking for is currently unavailable.
Please try again later.

If you are the system administrator of this resource then you should check the error log for details.

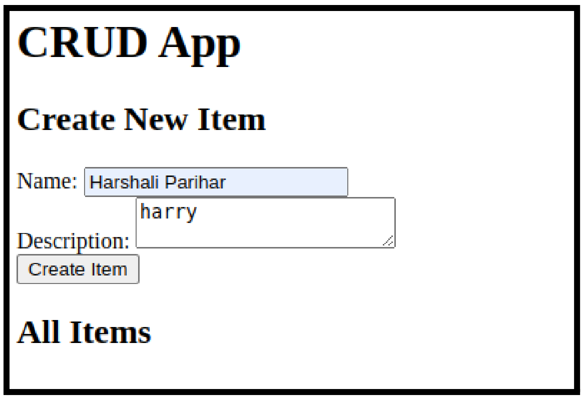
Faithfully yours, nginx.

Q.39 Create a container with ngnix web server and create one more container with mysql.



```
nginx-webapp-mysql
  Backend
    node_modules
    public
    .dockerignore
    app.js
    Dockerfile
    package-lock.json
    package.json
  docker-compose.yaml
  Dockerfile
  init-db.sql
  nginx.conf
```

sudo docker compose up
localhost:8000

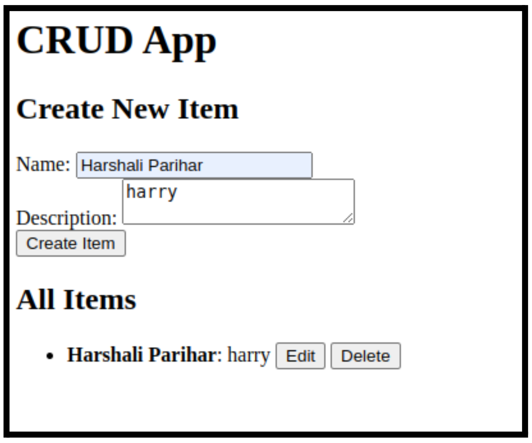


CRUD App

Create New Item

Name: Harshali Parihar
Description: harry
Create Item

All Items



CRUD App

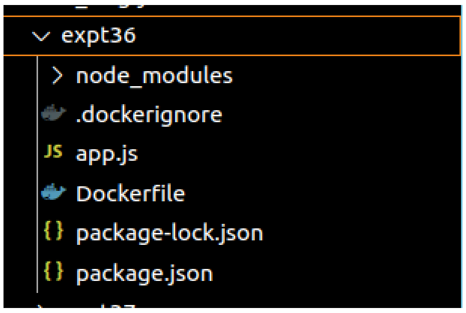
Create New Item

Name: Harshali Parihar
Description: harry
Create Item

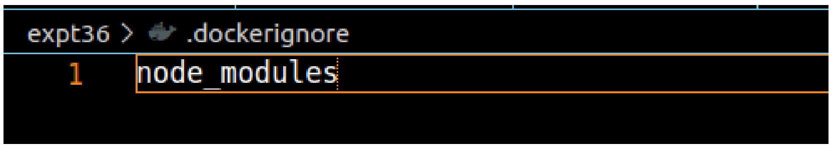
All Items

- **Harshali Parihar**: harry Edit Delete

Q.36 Create a Docker image of simple web application from using HTTP web server at port 5000 in host.



```
expt36
  node_modules
  .dockerignore
  app.js
  Dockerfile
  package-lock.json
  package.json
```



```
expt36 > .dockerignore
node_modules
```

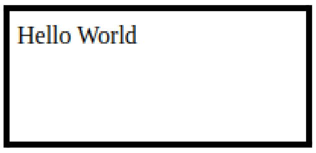
app.js code:

```
const express = require('express' )
const app = express()
const port = 5000
app.get ('/',(req,res) =>{
   res.send('Hello World');
}
)
app.listen(port,()=>{
   console.log('app listening on port 5000');
})
```

Dockerfile:

```
FROM node
COPY . /app
WORKDIR /app
RUN npm install
EXPOSE 5000
CMD ["npm", "start"]
```

sudo docker build -t expt36 .
sudo docker run -p 5000:5000 -it expt36
localhost:5000



```
Hello World
```

Q.37 Create a docker image of simple login form using Flask on port 7000.

Create expt37 folder -> in that create templates folder-> then create login.html

Code in login.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Login Form</title>
</head>
<body>
   <h2>Login Form</h2>
   <form method="post" action="/login">
       <label for="username">Username:</label>
       <input type="text" id="username" name="username"
required><br><br>
       <label for="password">Password:</label>
       <input type="password" id="password" name="password"
required><br><br>
       <input type="submit" value="Login">
   </form>
</body>
</html>
```

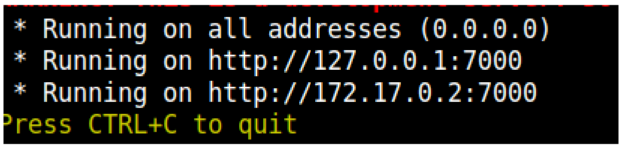
Create app.py in expt37 folder:

```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
   return render_template('login.html')

@app.route('/login', methods=['POST'])
def login():
   username = request.form.get('username')
   password = request.form.get('password')

   # Add your authentication logic here
   # For simplicity, let's just check if username and password are not
empty
   if username and password:
       return f'Login successful! Welcome, {username}!'
   else:
       return 'Invalid login credentials.'

if __name__ == '__main__':
   app.run(host='0.0.0.0', port=7000)
```

Create Dockerfile in expt37:

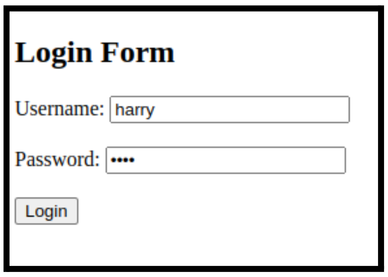
```
# Dockerfile
FROM python:3.9

# Set the working directory
WORKDIR /app

# Copy the application files to the container
COPY . /app

# Install dependencies
RUN pip install Flask

# Expose port 7000
EXPOSE 7000

# Start the Flask app
CMD ["python3", "app.py"]
```

cd expt37
sudo docker build -t flask .
sudo docker run -it flask



```
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:7000
* Running on http://172.17.0.2:7000
Press CTRL+C to quit
```

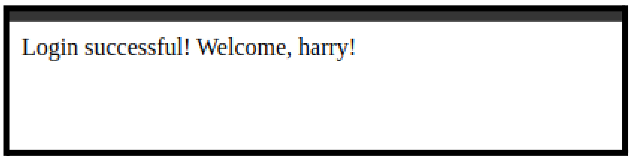
http://172.17.0.2:7000/login



Login Form

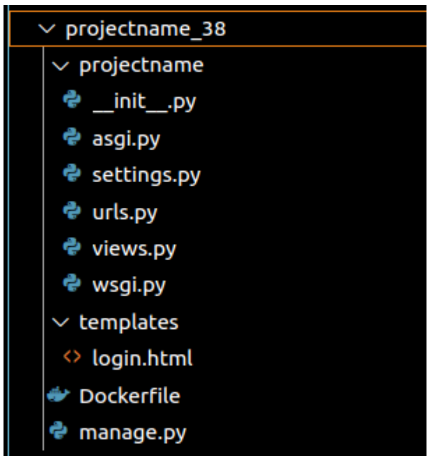
Username: harry

Password: ••••



Login successful! Welcome, harry!

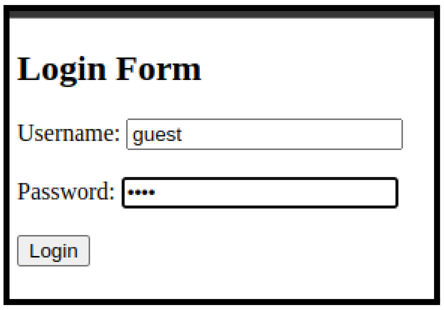
Q.38 Create a docker image of simple login form using django on port 6000.



```
projectname_38
  projectname
    __init__.py
    asgi.py
    settings.py
    urls.py
    views.py
    wsgi.py
  templates
    login.html
  Dockerfile
  manage.py
```

sudo docker build -t expt38 .
sudo docker run -p 8000:8000 -it expt38

localhost:8000



Login Form

Username: guest

Password: ••••

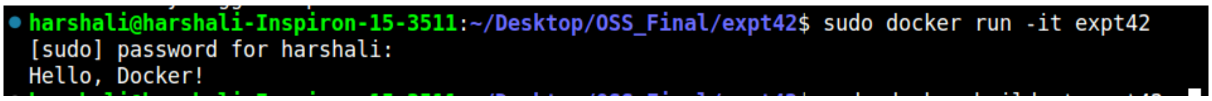
Login successful! Welcome, guest!

Q.42 Write a Docker File to pull the Ubuntu with open jdk and write any java application.

Dockerfile:

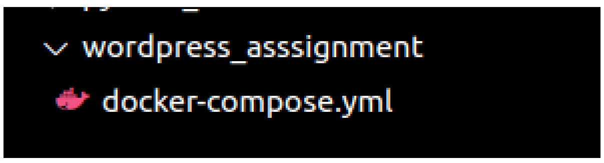
```
# Use the official Ubuntu base image
FROM ubuntu:latest
# Set environment variables
ENV DEBIAN_FRONTEND=noninteractive
# Update the package list and install OpenJDK
RUN apt-get update && \
   apt-get install -y openjdk-11-jdk
# Create a directory for the Java application
WORKDIR /usr/src/app
# Create a simple Java application (HelloWorld.java)
RUN echo 'public class HelloWorld { public static void main(String[]
args) { System.out.println("Hello, Docker!"); } }' > HelloWorld.java
# Compile the Java application
RUN javac HelloWorld.java
# Set the entry point to run the Java application
CMD ["java", "HelloWorld"]
```

sudo docker build -t expt42 .
sudo docker run -it expt42



```
harshali@harshali-Inspiron-15-3511:~/Desktop/OSS_Final/expt42$ sudo docker run -it expt42
[sudo] password for harshali:
Hello, Docker!
```

23. With the help of Docker-compose deploy the 'Wordpress' and 'Mysql' container and access the front end of 'Wordpress'



wordpress_asssignment

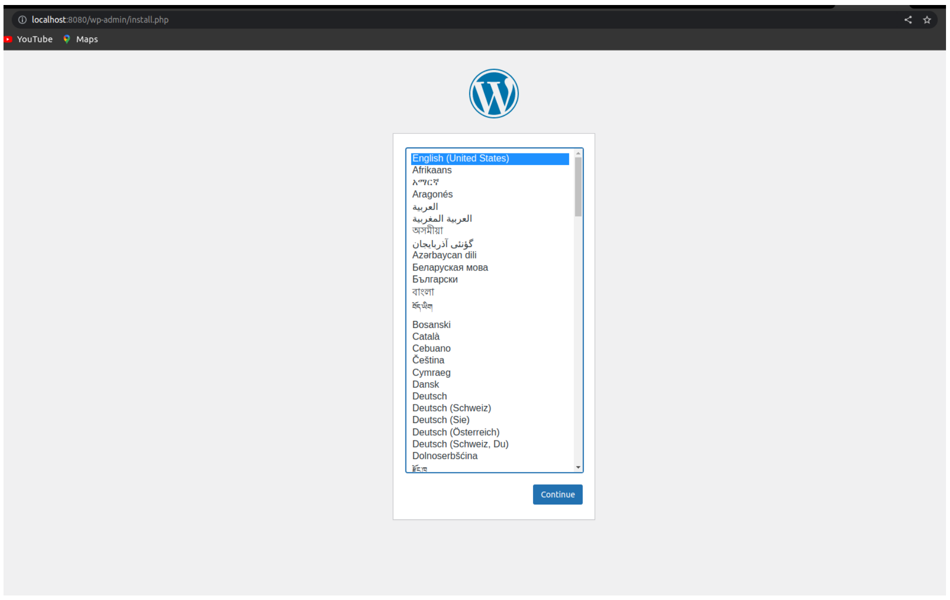
docker-compose.yml

create file docker-compose.yml:

```
version: '3.8'
services:
 wordpress:
   image: wordpress
   ports:
     - "8080:80"
   environment:
     WORDPRESS_DB_HOST: mysql
     WORDPRESS_DB_USER: vrw
     WORDPRESS_DB_PASSWORD: vrw
     WORDPRESS_DB_NAME: vrw
   volumes:
     - wordpress_data:/var/www/html
   depends_on:
     - mysql
 mysql:
   image: mysql
   environment:
     MYSQL_ROOT_PASSWORD: vrw
     MYSQL_DATABASE: vrw
     MYSQL_USER: vrw
     MYSQL_PASSWORD: vrw
   volumes:
     - mysql_data:/var/lib/mysql

volumes:
 wordpress_data:
 mysql_data:
```

sudo docker compose up

localhost:8080



English (United States)
Afrikaans
አማርኛ
Aragonés
العربية
العربية المغربية
অসমীয়া
گؤنئی آذربایجان
Azərbaycan dili
Беларуская мова
Български
বাংলা
བོད་ཡིག
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch
Deutsch (Schweiz)
Deutsch (Sie)
Deutsch (Österreich)
Deutsch (Schweiz, Du)
Dolnoserbšćina
རྫོང་ཁ

Q.40 Create a simple web form to insert the records in mysql database.