

6 –Container concept Assignment

Name: Rahul R

Training Id: WC032

Branch: Chennai

Mail Id: rahulrengi12@gmail.com

Github repo: <https://github.com/rahulrengesh/docker-assignment>

What is docker:

A software platform called Docker enables programmers to build, distribute, and operate applications within containers. A container is a small, independent executable software package that contains all of the configuration files, code, libraries, and dependencies required to run a programme. It is simpler to deploy and manage programmes across various contexts, such as development, testing, and production, thanks to Docker, which offers a portable and uniform approach to package and distribute applications. Because of its adaptability, scalability, and usability, open-source software called Docker has grown in popularity in recent years.

What is container in docker:

A container in Docker is a small, portable executable package that contains all of the components required to run an application, including the code, runtime, system tools, libraries, and settings. Applications may run in an isolated environment with the help of containers, which make sure they only have access to the resources and dependencies they require to work properly.

Images, which are read-only templates containing the application and all of its dependencies, serve as the foundation for containers in Docker. Containers are built using images, and each one has its own file system, network interface, and process space.

Docker containers are segregated from the host system and other containers, enabling the conflict-free operation of several applications on the same host. Additionally, because containers can be readily transferred between various

contexts, including development, testing, and production, they offer a fast and effective approach to deploy programmes.

What is multi container concept:

An strategy to creating and deploying applications that makes use of numerous containers is known as a multi-container architecture. Each container in this design is in charge of a particular service or component of the programme, such as a web server, database server, or messaging system.

Multi-container architectures are frequently employed in complicated systems that ask for the interaction of several services or components. A web application, for instance, could need a web server, application server, and database server to work properly. These components each have the option to be deployed in their own container, enabling portability, scalability, and isolation.

An important advantage of a multi-container design is that it makes managing and scaling applications simpler. According to the unique requirements of the application, each container may be scaled individually, which enables developers to maximise performance and resource use. Additionally, containers may be set up on many hosts, offering fault tolerance and high availability.

What is docker compose:

A tool called Docker Compose is used to create and execute multi-container Docker applications. Developers may represent the parts and services of their application as a collection of linked Docker containers since it offers a straightforward method for defining and managing the interactions between various containers.

To specify the services, networks, and volumes needed for an application, Docker Compose employs a YAML file. Numerous configuration settings, including image names, container names, environment variables, ports, and volumes, may be specified in the YAML file.

Developers can quickly launch and manage their whole application stack with a single command by using Docker Compose. The application's services are started, stopped, and scaled by Compose, who also makes sure that all containers are correctly connected and interacting with one another.

TASK:

Design and develop an application with multi container architecture using Docker compose whereas a front end application running on one container gets the registration number of a student from user and communicate the other container which is running the database service to collect the COVID vaccinated status of the student. • The Web App should ask a user to enter the registration number and forward the registration number to the database container to get the vaccination details of the given student. Assume that the database service already maintain the details of the students and their vaccination details such as (RegNo,Name and Vaccination_Status(Yes/No)). • Use suitable dockerfile and docker-compose yaml/json file and then launch application • Test the application through the front-end application exposed to the port 5000 of the localhost Create a separate folder for your application in the docker and then create the dockerfile,docker compose file, python front end and database table creation and row insertion. Create a volume and mount the same for storing and sharing the python from end script

Tech stack used:

Front-end technology: Python (Flask)

Back-end technology: MySQL

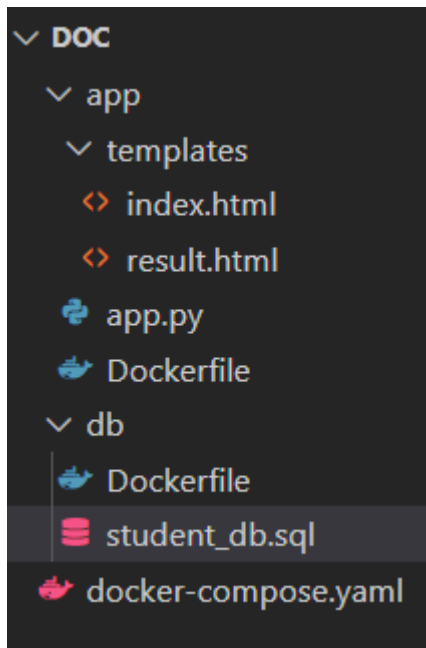
How to install docker:

- `sudo apt-get update`
- `sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg`
- `echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`
- `sudo apt-get update`

- `sudo apt-get install docker-ce docker-ce-cli containerd.io`
- `sudo docker run hello-world`

Directory:

```
app---->templates---->index.html
                        result.html
    app.py
    Dockerfile
    requirements.txt
db----->Dockerfile
        student_db.sql
docker-compose.yaml
README.txt
```



Front-end part:

The purpose of front end is to get the registration number of a student from user and communicate the other container which is running the database service to collect the COVID vaccinated status of the student. The Web App should ask a user to enter the registration number and forward the registration number to the database container to get the vaccination details of the given student. Assume that the database service already maintain the details of the

students and their vaccination details such as (RegNo,Name and Vaccination_Status(Yes/No))

Requirements for front-end part:

Flask==2.0.1

mysql-connector-python==8.0.26

What is Flask:

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

App.py:

app.py is typically the main file for a Flask application. It contains the Python code that defines the routes, views, and logic of the web application.

In the context of the above question, app.py defines the route for the vaccination status API endpoint. When a request is made to the endpoint, the function defined in app.py retrieves the registration number from the request data, queries the database to retrieve the vaccination status for that student, and returns the result as a JSON response.

App.py also includes the configuration settings for the Flask application, such as the secret key and the database connection details. It can also include any additional logic, such as authentication or error handling, that is required for the web application to function correctly.

Overall, app.py plays a central role in defining the functionality of the Flask web application and responding to user requests.

Dockerfile for front-end part:

The Dockerfile of the frontend specifies how to build a Docker image for the frontend application. It contains instructions for building and configuring the environment required to run the frontend application.

The Dockerfile usually starts with a base image, which is an existing image that provides a starting point for the new image. Then, it installs any necessary dependencies for the frontend application using package managers such as pip. It also copies the source code of the frontend application into the image.

The Dockerfile also includes instructions for running the frontend application when the image is launched as a container. These instructions typically specify the command to start the application, the port to expose, and any environment variables needed for the application to run properly.

Once the Dockerfile is built into a Docker image, it can be used to launch containers that run the frontend application in a consistent and portable manner across different environments.

Back-end part:

The task of backend part is to create a table for storing vaccination details of students along with their name and reg.no.

Mysql file:

The SQL statements are executed when the MySQL service starts up, as specified in the Dockerfile and docker-compose.yml files. The table creation statement defines the structure of the table, including the column names and data types. The data insertion statement adds some example data to the table, which can be queried by the Flask application when a request is made to the vaccination status endpoint.

Overall, the MySQL file is a crucial component of the Docker-based application stack, as it defines the database schema and initial data for the application to function correctly.

Docker-compose file:

The Docker Compose file is used to define and run a multi-container Docker application. It specifies the services that the application requires and how they should be configured and connected to each other.

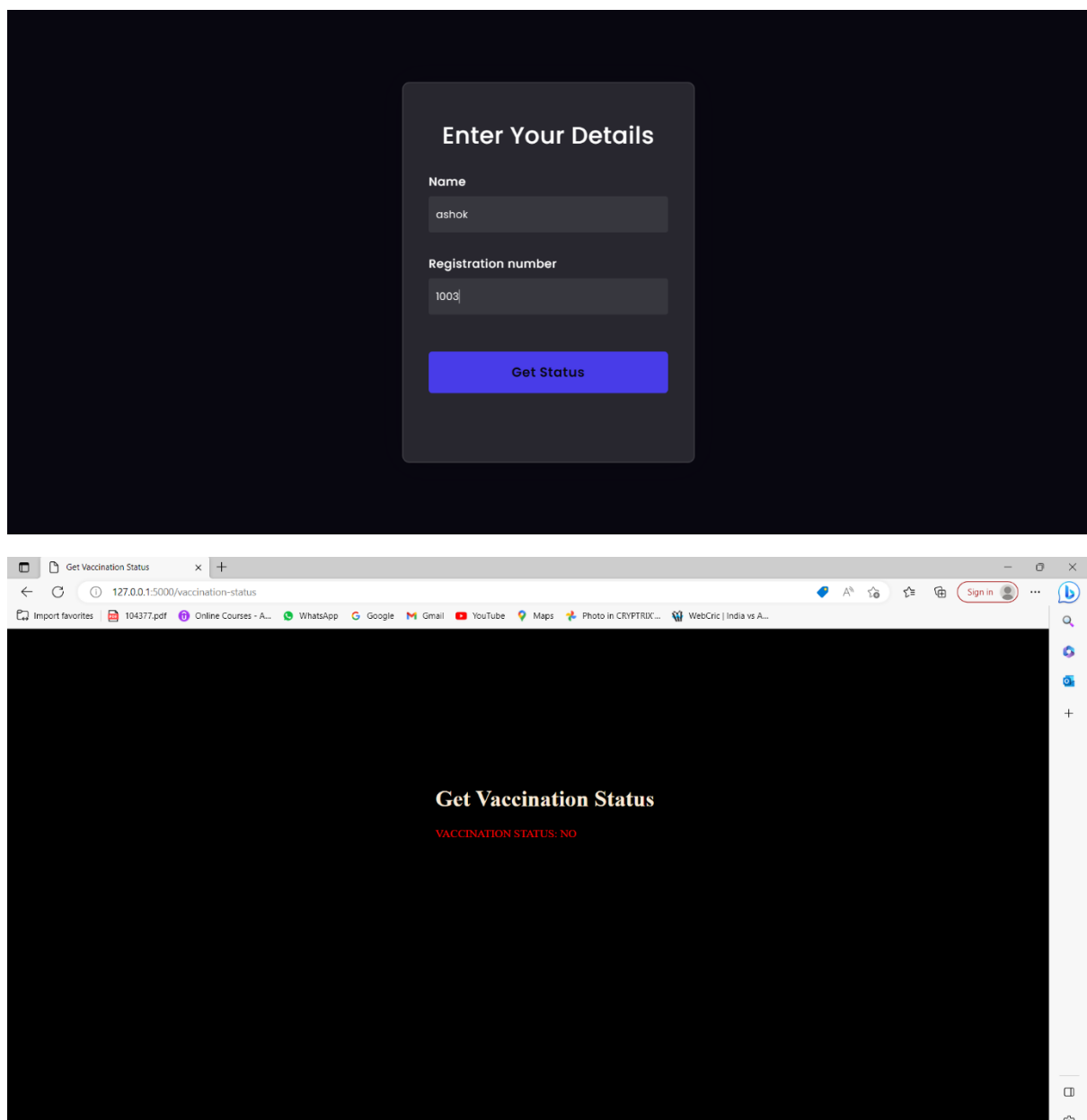
In the context of the above question, the docker-compose.yml file defines two services: the frontend service, which runs the Flask web application, and the db service, which runs the MySQL database server. It also specifies how these

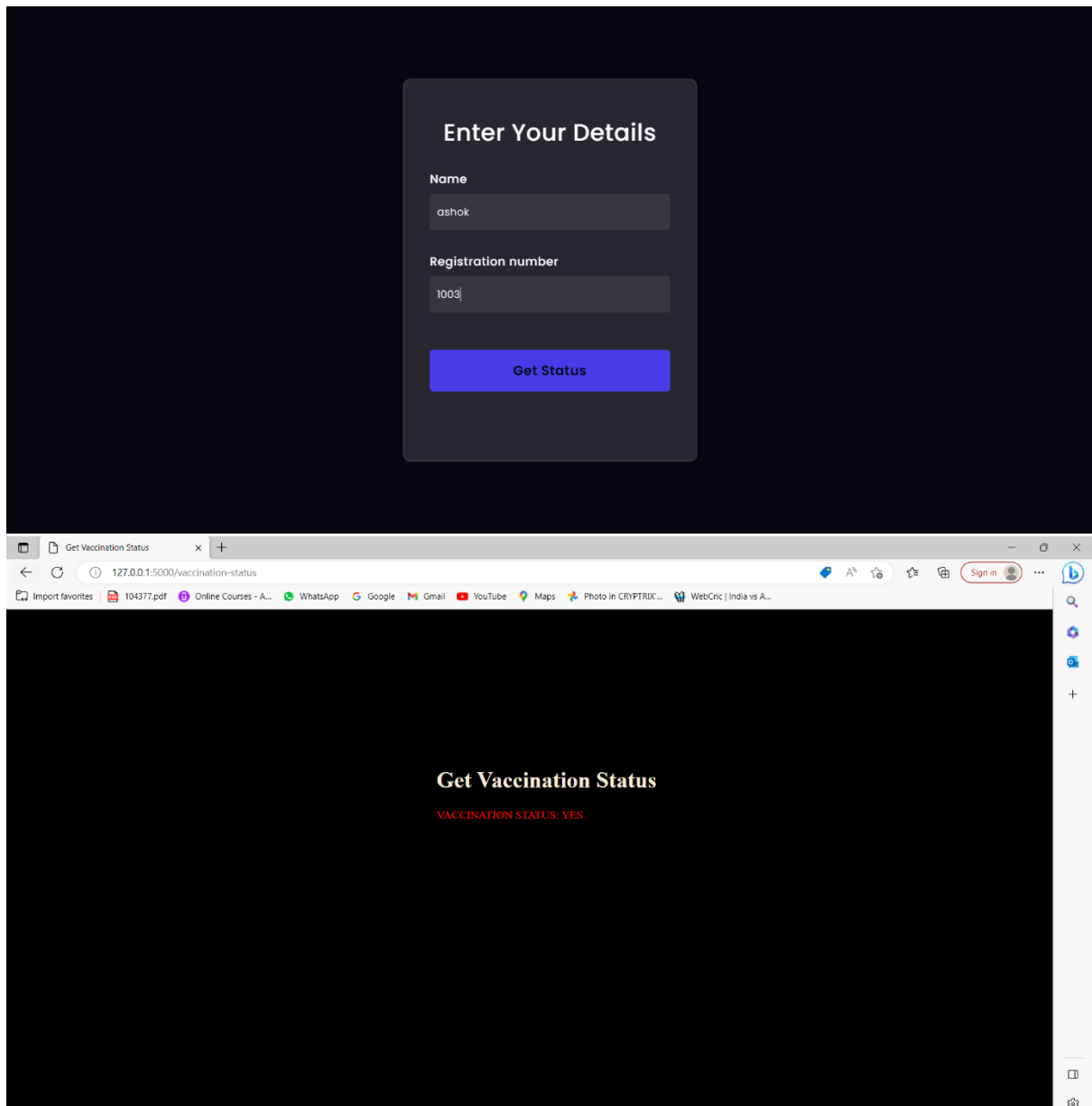
services should interact with each other by defining the network connections and environment variables.

The Docker Compose file is used to simplify the deployment and management of multi-container applications by allowing developers to define the infrastructure and dependencies required for the application in a single file. By running the docker-compose command, the entire application stack can be started up and managed as a single unit, making it easy to deploy and scale the application.

The output will be executed in localhost 5000 port.

Output Screenshots:



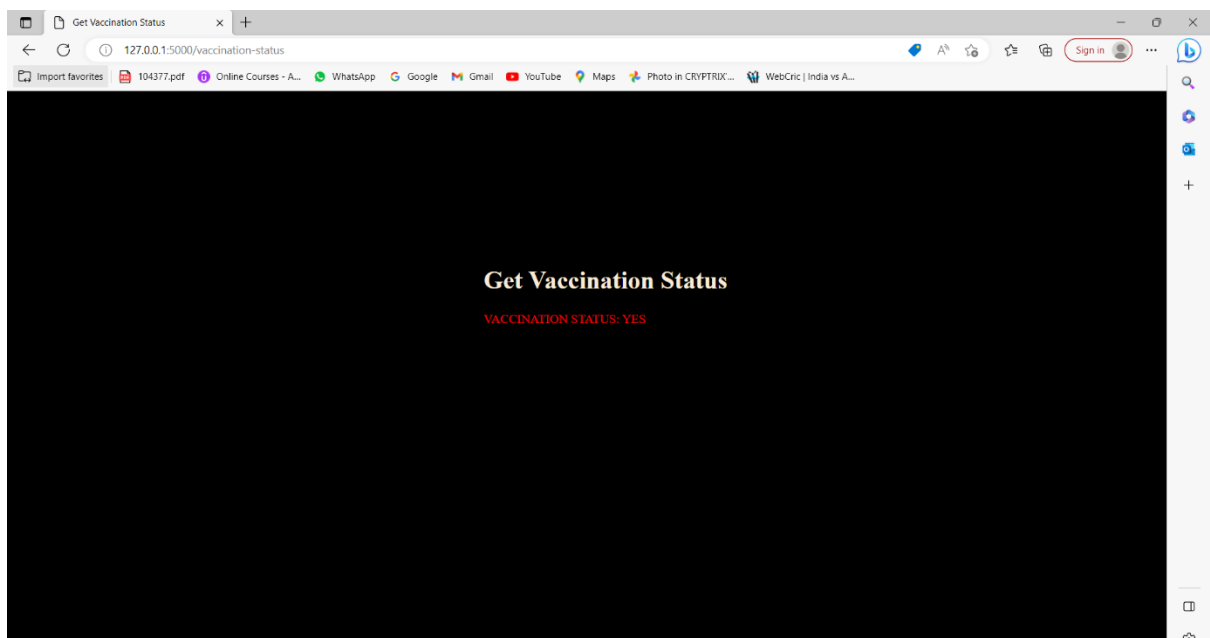


Enter Your Details

Name

Registration number

Get Status




```
Welcome index.html X
C: > Users > Rahul > index.html > html > body > form > input#reg_no

115 .social i{
116     margin-right: 4px;
117 }
118
119 </style>
120 </head>
121 <body>
122     <div class="background">
123         <div class="shape"></div>
124         <div class="shape"></div>
125     </div>
126     <form>
127         <h3>Enter Your Details</h3>
128
129         <label for="username">Name</label>
130         <input type="text" placeholder="Name" id="name">
131
132         <label for="text">Registration number</label>
133         <input type="text" placeholder="Register" id="reg_no">
134
135         <button>Get Status</button>
136     </form>
137 </body>
138 </html>
139
140
```

```
docker-compose.yaml
1  version: '3'
2
3  services:
4      web:
5          build: .
6          ports:
7              - "5000:5000"
8          volumes:
9              - ./app
10         depends_on:
11             - db
12         environment:
13             - DATABASE_HOST=db
14         networks:
15             - mynet
16
17     db:
18         image: mysql:5.7
19         environment:
20             - MYSQL_ROOT_PASSWORD=mysecretpassword
21             - MYSQL_DATABASE=student_vaccination_db
22         volumes:
23             - dbdata:/var/lib/mysql
24         networks:
25             - mynet
26
27     volumes:
28         dbdata:
29
30     networks:
31         mynet:
32
```

```
1 FROM mysql:8.0.25
2 COPY ./student_db.sql /docker-entrypoint-initdb.d/
3 ENV MYSQL_ROOT_PASSWORD=password
4 ENV MYSQL_DATABASE=student_vaccination_db
5 EXPOSE 3306
6
```

app >  Dockerfile > ...

```
1 FROM python:3.8-slim-buster
2 WORKDIR /app
3 COPY requirements.txt .
4 RUN pip install --no-cache-dir -r requirements.txt
5 COPY . .
6 EXPOSE 5000
7 CMD ["python", "app.py"]
8
```

```

1  from flask import Flask, request, jsonify, render_template
2  import mysql.connector
3
4  app = Flask(__name__)
5
6  @app.route('/', methods=['GET'])
7  def index():
8      return render_template('index.html')
9
10 @app.route('/vaccination-status', methods=['GET'])
11 def get_vaccination_status():
12     reg_no = request.form.get('reg_no')
13
14     db = mysql.connector.connect(
15         host='db',
16         user='root',
17         password='password',
18         database='student_vaccination_db'
19     )
20
21     cursor = db.cursor()
22     cursor.execute(f"SELECT Vaccination_Status FROM Students WHERE RegNo='{reg_no}'")
23     result = cursor.fetchone()
24
25     if result:
26         return render_template('result.html', vaccination_status=result[0])
27     else:
28         return render_template('result.html', error='Invalid registration number')
29
30 if __name__ == '__main__':
31     app.run(host='0.0.0.0', port=5000)
32

```

```

</head>
<body>
    <h1>Get Vaccination Status</h1>
    {% if error %}
    <p>{{ error }}</p>
    {% else %}
    <p>VACCINATION STATUS: {{ vaccination_status }}</p>
    {% endif %}
</body>

```

The github repo: <https://github.com/rahulrengesh/docker-assignment>

Outcome of this assignment:

By this assignment I learnt what is docker what is docker container and how it works also I learnt how to build an application with multi container architecture and how to use docker compose to combine the functionalities of front end and back-end part.

Code:

app.py

```

from flask import Flask, request, jsonify, render_template
import mysql.connector

app = Flask(__name__)

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/vaccination-status', methods=['POST'])
def get_vaccination_status():
    data = request.form.to_dict()
    reg_no = data['reg_no']

    db = mysql.connector.connect(
        host='db',
        user='root',
        password='password',
        database='student_vaccination_db'
    )

    cursor = db.cursor()

    cursor.execute(f"SELECT Vaccination_Status FROM Students WHERE
RegNo='{reg_no}'")

    result = cursor.fetchone()

```

```
if result:
    return render_template('result.html', vaccination_status=result[0])
else:
    return render_template('result.html', error='Invalid registration number')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

dockerfile for front-end:

```
FROM python:3.8-slim-buster
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python", "app.py"]
```

Students_db.sql:

```
CREATE TABLE students (
    reg_no VARCHAR(10) NOT NULL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    vaccination_status VARCHAR(5) NOT NULL
);

INSERT INTO students (reg_no, name, vaccination_status)
VALUES
```

```
('1001', 'rahul', 'YES'),  
('1002', 'ram', 'NO'),  
('1003', 'ashok', 'YES');
```

Dockerfile for backend (db):

```
FROM mysql:8.0.25  
COPY ./student_db.sql /docker-entrypoint-initdb.d/  
ENV MYSQL_ROOT_PASSWORD=password  
ENV MYSQL_DATABASE=student_vaccination_db  
EXPOSE 3306
```

Docker_compose.yaml:

```
version: '3'
```

```
services:
```

```
  web:
```

```
    build: .
```

```
    ports:
```

```
      - "5000:5000"
```

```
    volumes:
```

```
      - ./app
```

```
    depends_on:
```

```
      - db
```

```
    environment:
```

```
      - DATABASE_HOST=db
```

```
    networks:
```

```
      - mynet
```

db:

image: mysql:5.7

environment:

- MYSQL_ROOT_PASSWORD=mysecretpassword
- MYSQL_DATABASE=student_vaccination_db

volumes:

- dbdata:/var/lib/mysql

networks:

- mynet

volumes:

dbdata:

networks:

mynet:

index.html:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Get Vaccination Status</title>
```

```
<style>
```

```
*{
```

```
background-color: aqua;
```

```
}
```

```
form{
```



```
position: relative;
left: 600px;
top:200px;
border: 1cm;

}
h1{
color: blueviolet;
position: relative;
left: 600px;
top:200px;
}
h2{
color: rgb(226, 116, 43);
position: relative;
left: 600px;
top:200px;
}
</style>
<link rel="stylesheet"
href=
"https://unpkg.com/purecss@1.0.0/build/pure-min.css"
integrity=
"sha384-
nn4HPE8lTHyVtfCBi5yW9d20FjT8BJwUXyWZT9InLYax14RDjBj46LmSztkmNP9w"
crossorigin="anonymous"/>
```

</head>

<body>

<h2>WORLD LINE ASSIGNMENT</h2>

<h1>Get Vaccination Status</h1>

<form class="pure-form" method="POST" action="/vaccination-status">

<label for="reg_no">Registration Number:</label>

<input type="text" id="reg_no" name="reg_no" required>

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<button type="submit">Submit</button>

</form>

</body>

</html>

Result.html:

<!DOCTYPE html>

<html>

<head>

<title>Get Vaccination Status</title>

<style>

```

*{
    background-color: black;

}

h1{
    color: antiquewhite;
    position: relative;
    left: 600px;
    top:200px;
}

p{
    color: red;
    position: relative;
    left: 600px;
    top:200px;
}

</style>
</head>
<body>
    <h1>Get Vaccination Status</h1>
    {% if error %}
        <p>{{ error }}</p>
    {% else %}
        <p>VACCINATION STATUS: {{ vaccination_status }}</p>
    {% endif %}
</body>

```

</html>