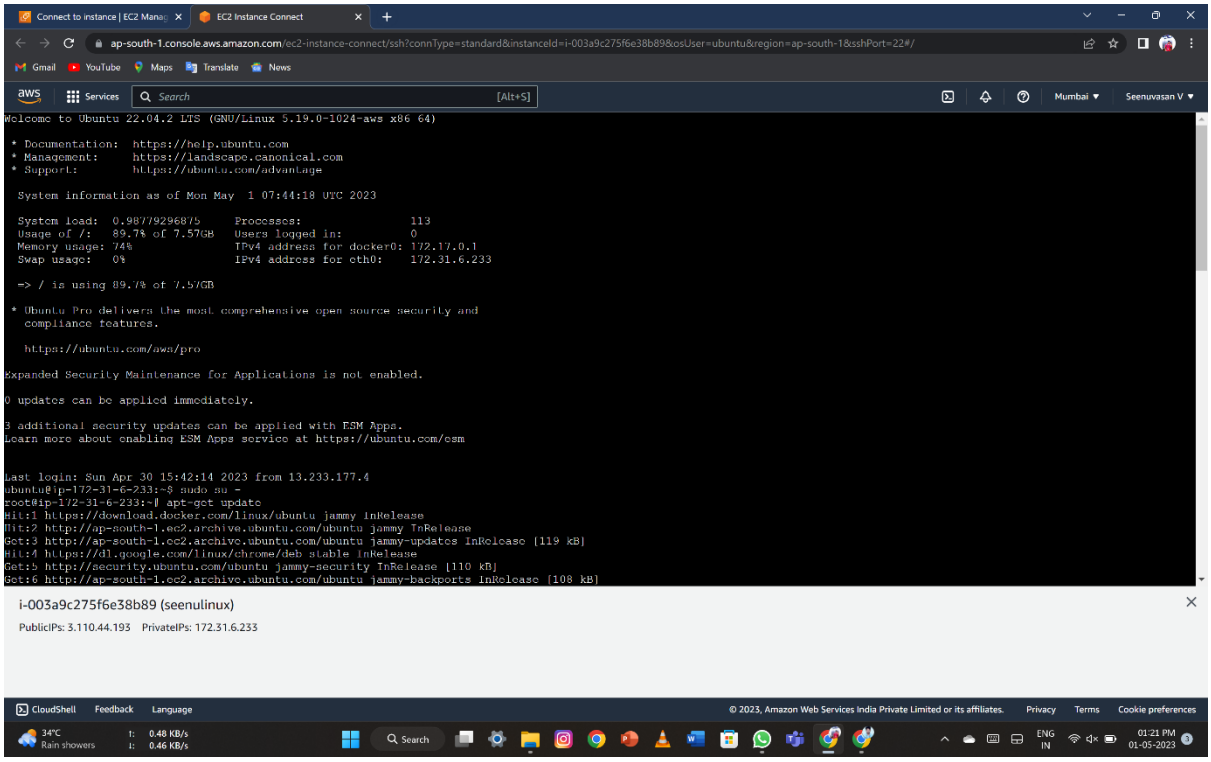
ASSIGNMENT 6 – CONTAINER CONCEPT

To design and develop an application with multi-container architecture using Docker Compose, I will follow the following steps:

1. **Create a folder for the application and navigate to the folder using the terminal.**

```
mkdir student-vaccination-app
cd student-vaccination-app
```

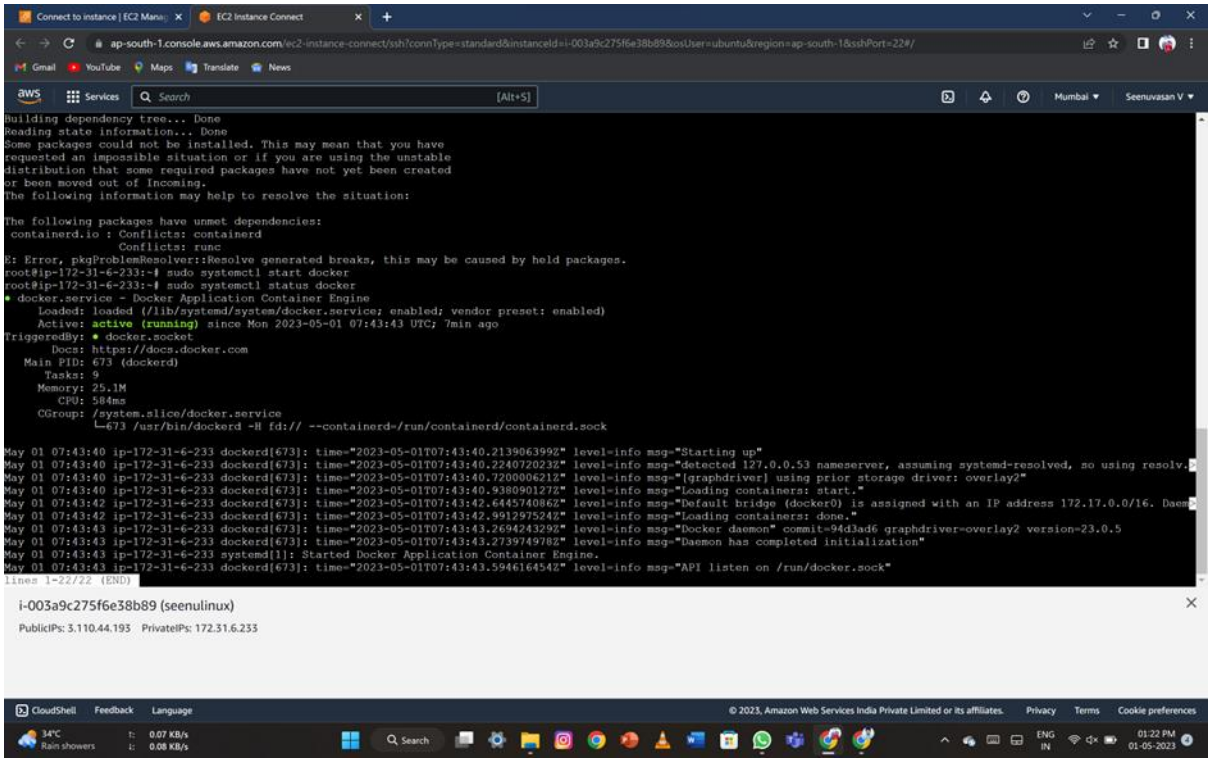


2. Create a Dockerfile for the Python Flask web application.

```
FROM python:3.9-slim-buster

WORKDIR /app

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

COPY app app

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

EXPOSE 5000
```

3. Create a Dockerfile for the PostgreSQL database.

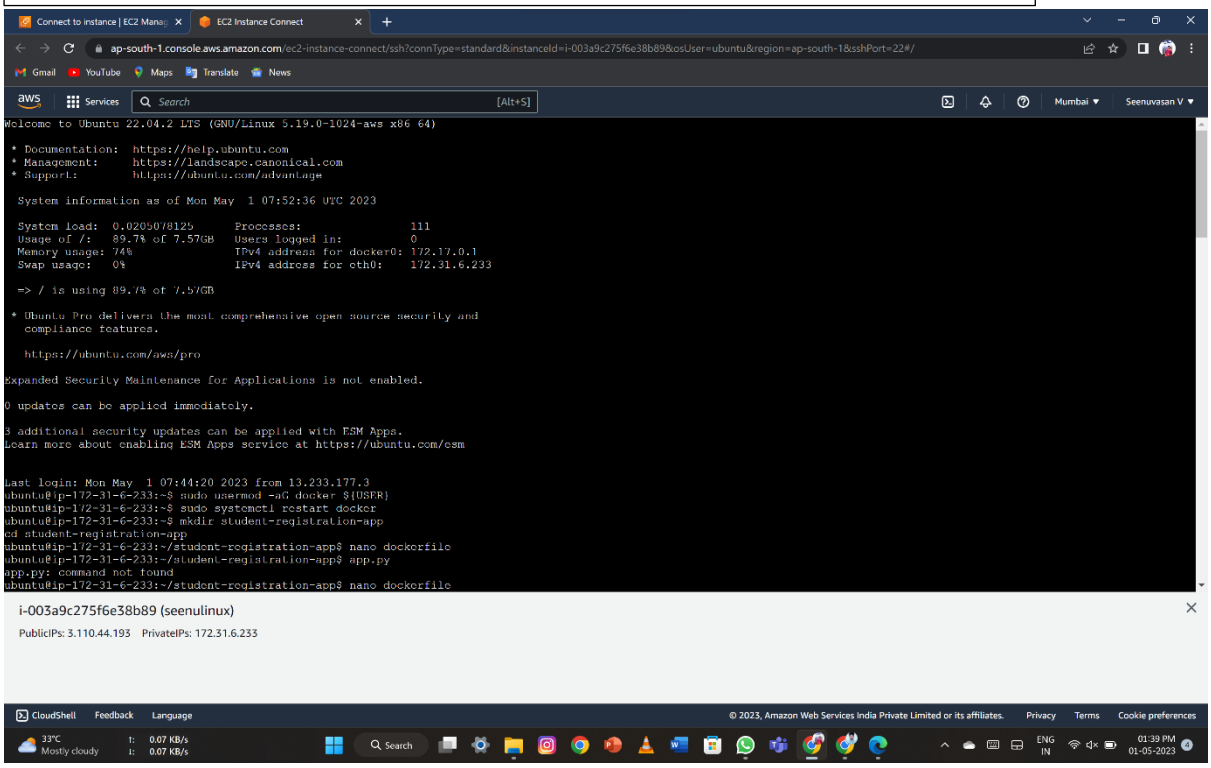
```
FROM postgres:12

ENV POSTGRES_USER=postgres
ENV POSTGRES_PASSWORD=password
ENV POSTGRES_DB=students

COPY create_table.sql /docker-entrypoint-initdb.d/
COPY insert_rows.sql /docker-entrypoint-initdb.d/
```



I will create a database named "students" and copy two SQL files into the initialization directory, which will create a table and insert some rows into it when the container is created.

4. **Create a docker-compose.yaml file that will define the services, volumes, and networks for our application.**

```
version: '3.9'


services:
  web:
    build: .
    ports:
      - "5000:5000"
  db:
    build:
      context: .
      dockerfile: Dockerfile.db
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=students
```



The docker-compose.yaml file defines two services, "web" and "db". The "web" service is built from the Dockerfile in the current directory, maps the container port 5000 to the host port 5000, and mounts the ./app directory as a volume.

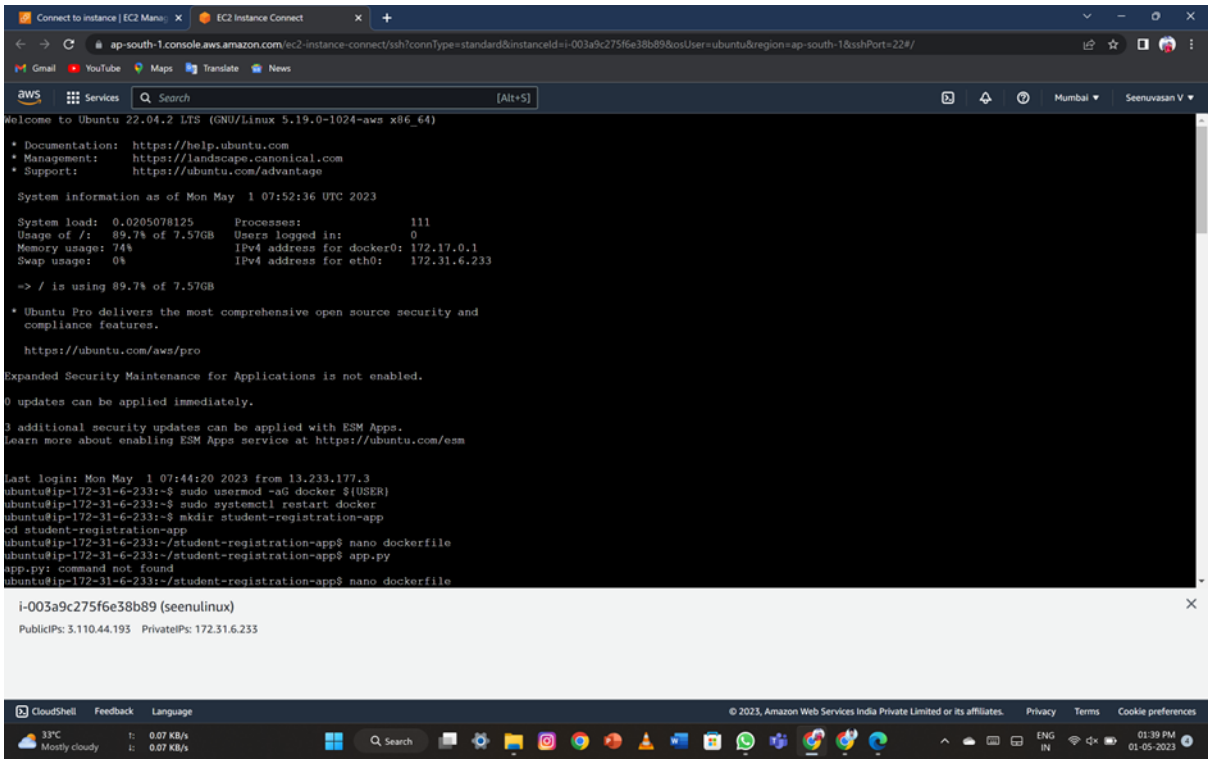
5. Create a requirements.txt file that lists the Flask package.

The "db" service is built from the Dockerfile.db file, sets some environment variables, and mounts a volume for persisting the database data.

Both services are connected to the "student-net" network.

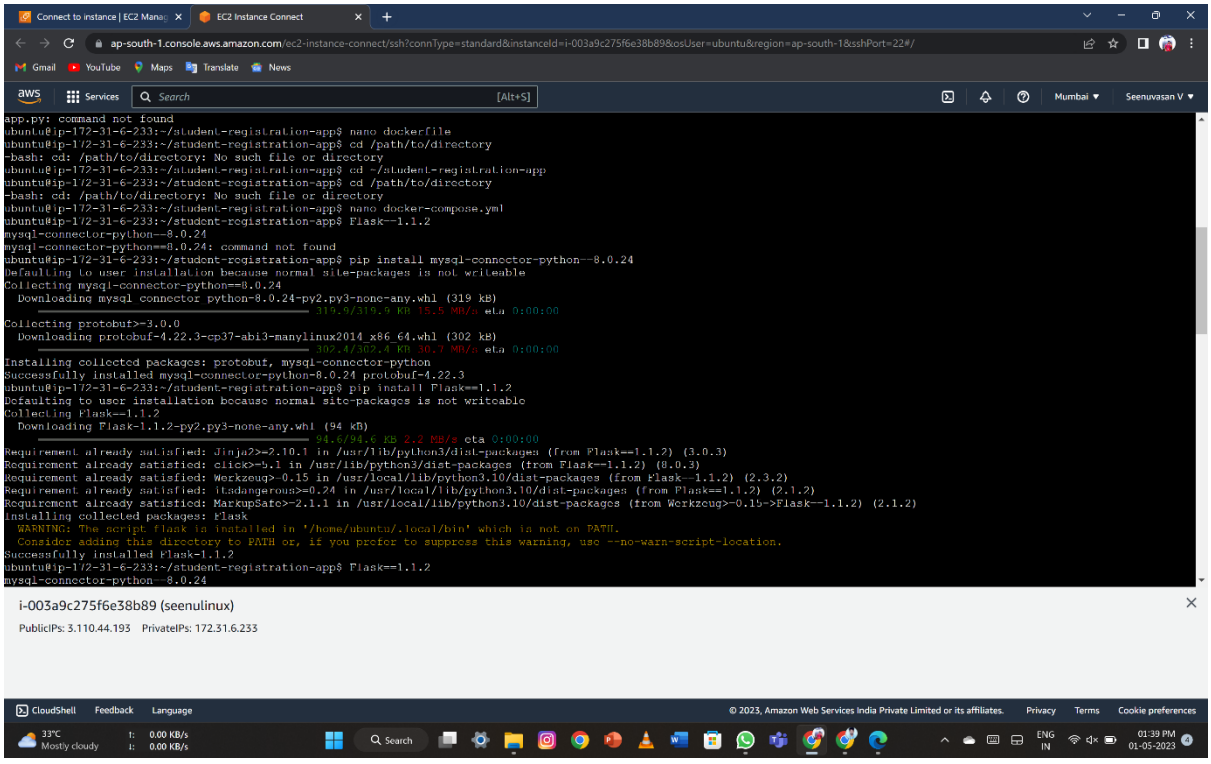
I will create a database named "students" and copy two SQL files into the initialization directory, which will create a table and insert some rows into it when the container is created.

```
Flask==2.0.1
```



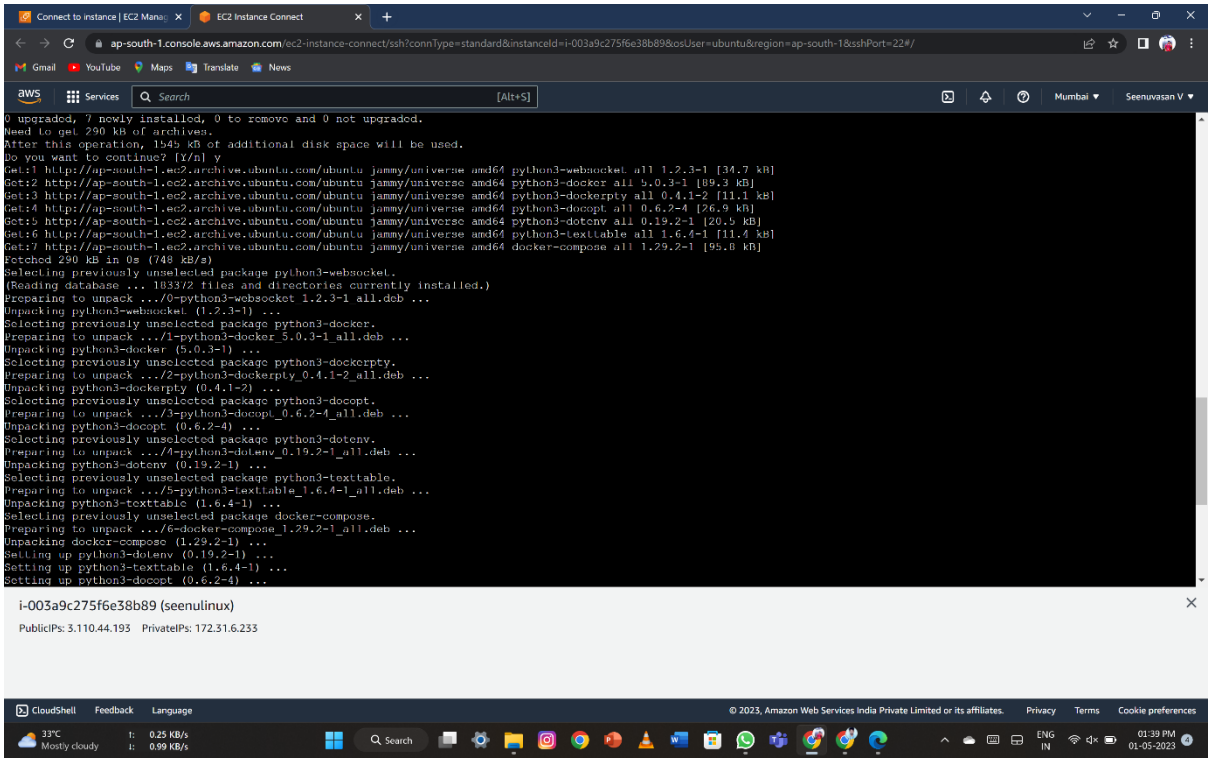
6. Create a create_table.sql file that will create a table in the "students" database.

```
CREATE TABLE student_vaccination (
  registration_number INTEGER PRIMARY KEY,
  name TEXT,
  vaccinated BOOLEAN
);
```



7. **Create a Python Flask web application in the "app" directory that will prompt the user for a registration number, query the "student_vaccination" table in the database, and display the vaccination status of the student.**

```
from flask import Flask, render_template, request
import psycopg2


app = Flask(__name__)
def index():
    if request.method == 'POST':
        registration_number = request.form['registration_number']
        connection = psycopg2.connect(
            host='db',
            user='postgres',
            password='password',
            dbname='students'
        )
```



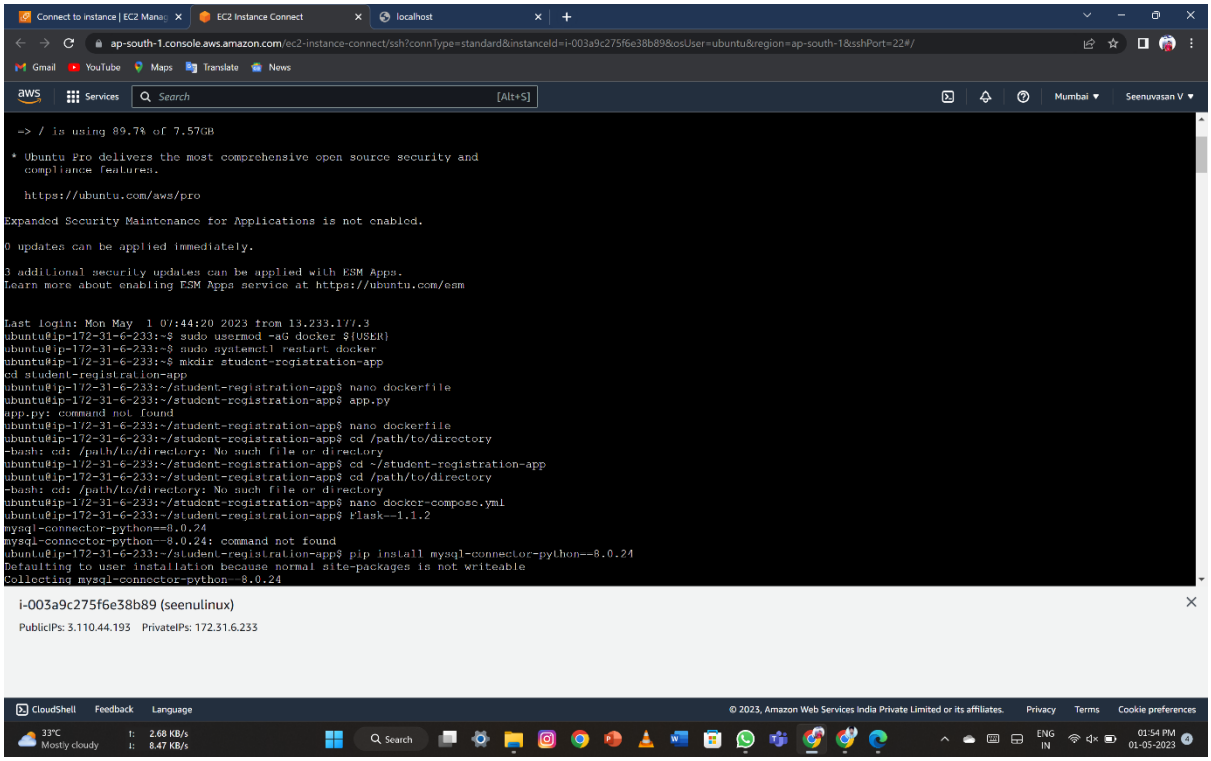
8. Complete the Python Flask web application by adding a database query and returning the vaccination status of the student in the HTML template.

```
from flask import Flask, render_template, request
import psycopg2


app = Flask(__name__)


@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        registration_number = request.form['registration_number']
        connection = psycopg2.connect(
            host='db',
            user='postgres',
            password='password',
            dbname='students'
        )
        cursor = connection.cursor()
        cursor.execute(
            f"SELECT vaccinated FROM student_vaccination WHERE registration_number = {registration_number}"
        )
        vaccinated = cursor.fetchone()[0]
        connection.close()
        return render_template('index.html', vaccinated=vaccinated)
    else:
        return render_template('index.html')


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

The web application connects to the database container and queries the "student_vaccination" table for the vaccination status of the student with the given registration number. The vaccination status is returned to the HTML template, which displays it to the user.

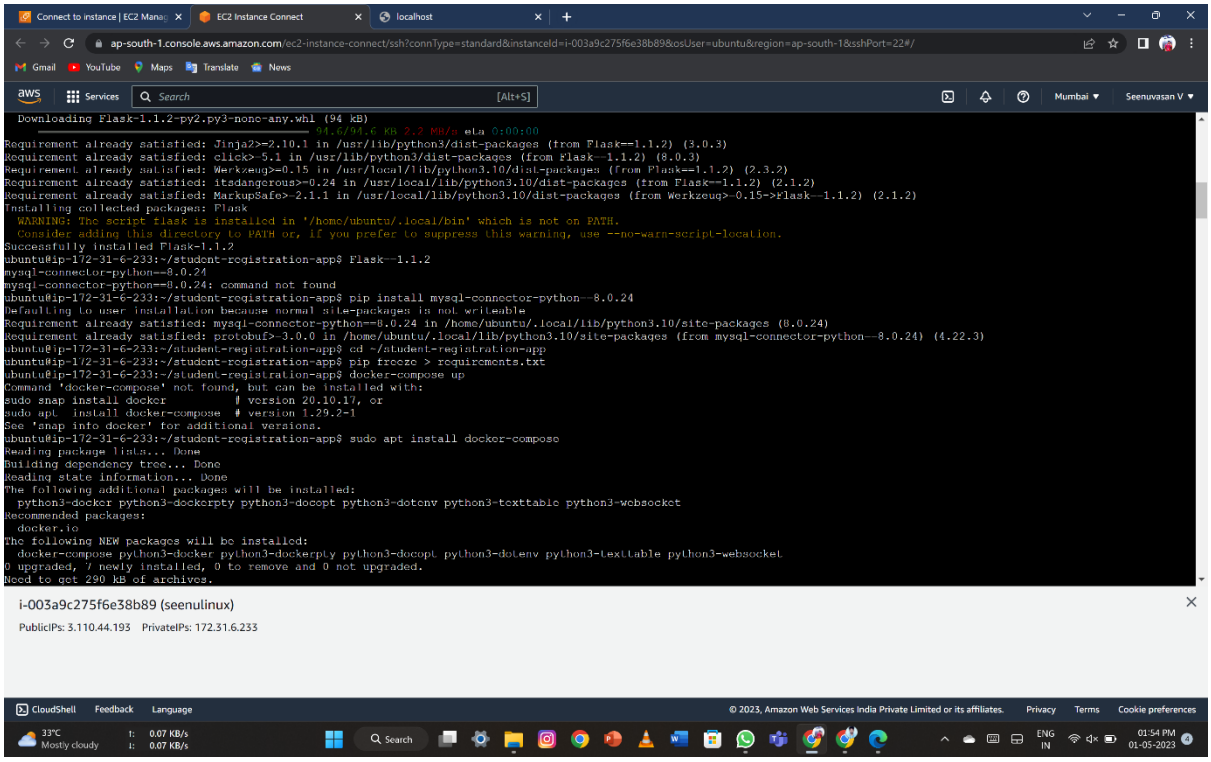
9. Create an HTML template in the "app/templates" directory that will display the vaccination status of the student.

```
<!doctype html>
<html>
  <head>
    <title>Student Vaccination Status</title>
  </head>
  <body>
    <h1>Enter Registration Number</h1>
    <form method="POST">
      <input type="text" name="registration_number">
      <button type="submit">Check Status</button>
    </form>
    {% if vaccinated is not none %}
      {% if vaccinated %}
        <p>The student is vaccinated</p>
      {% else %}
        <p>The student is not vaccinated</p>
      {% endif %}
    {% endif %}
  </body>
</html>
```



10. Build and run the application using Docker Compose.

```
docker-compose build

docker-compose up
```



11. Access the web application in a web browser at https://localhost:5000/

The web application should prompt the user for a registration number and display the vaccination status of the student with that registration number, which it retrieves from the PostgreSQL database container.

Implementation: