Dockerized Healthcare Python Flask Service Report

Course Name: DevOps Fundamentals

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
01.	Vanshika Vyas	EN22CS3011062
02.	Tulsi Chouhan	EN22CS3011038
03.	Swayam Mishra	EN22CS3011009
04.	Ujjawal Goswami	EN22CS3011043
05.	Mukund Mishra	EN22ME304046

Group Name:02D9

Project Number:DO-02

Industry Mentor Name:Mr. Vaibhav

University Mentor Name:Dr. Ritesh Joshi

Academic Year:2025-26

Problem Statement and Objectives

1. **Problem Statement:** Traditional deployment of a Python Flask-based Healthcare Management System can lead to environment inconsistencies, dependency conflicts, and deployment challenges across local and cloud platforms.
   The problem is to containerize the healthcare application using Docker and Docker Compose to ensure portability, consistency, security, and reliable deployment on an AWS EC2 instance, while maintaining optimized image size and persistent data storage.

2. **Project Objectives:**
   - To containerize the Python Flask Healthcare application using Docker.
   - To implement a multi-stage Docker build for optimized and secure images.
   - To use Docker Compose for simplified orchestration and management.
   - To ensure data persistence using Docker volumes with SQLite.
   - To deploy the application on an AWS EC2 Linux instance.
   - To achieve portability, consistency, and reliability across environments.

3. **Scope Of the Project:**
   The scope of the Dockerized Healthcare Python Flask Service project includes:
   - Development of a basic Healthcare Management System using Python Flask and SQLite.
   - Containerization of the application using multi-stage Docker builds following industry best practices.
   - Orchestration of the application using Docker Compose.
   - Deployment of the containerized application on an AWS EC2 Linux instance.
   - Implementation of persistent storage using Docker volumes.

- Ensuring security, portability, and reliability across local and cloud environments.
- The project focuses on containerization and cloud deployment and does not include CI/CD pipeline implementation or Kubernetes-based orchestration.

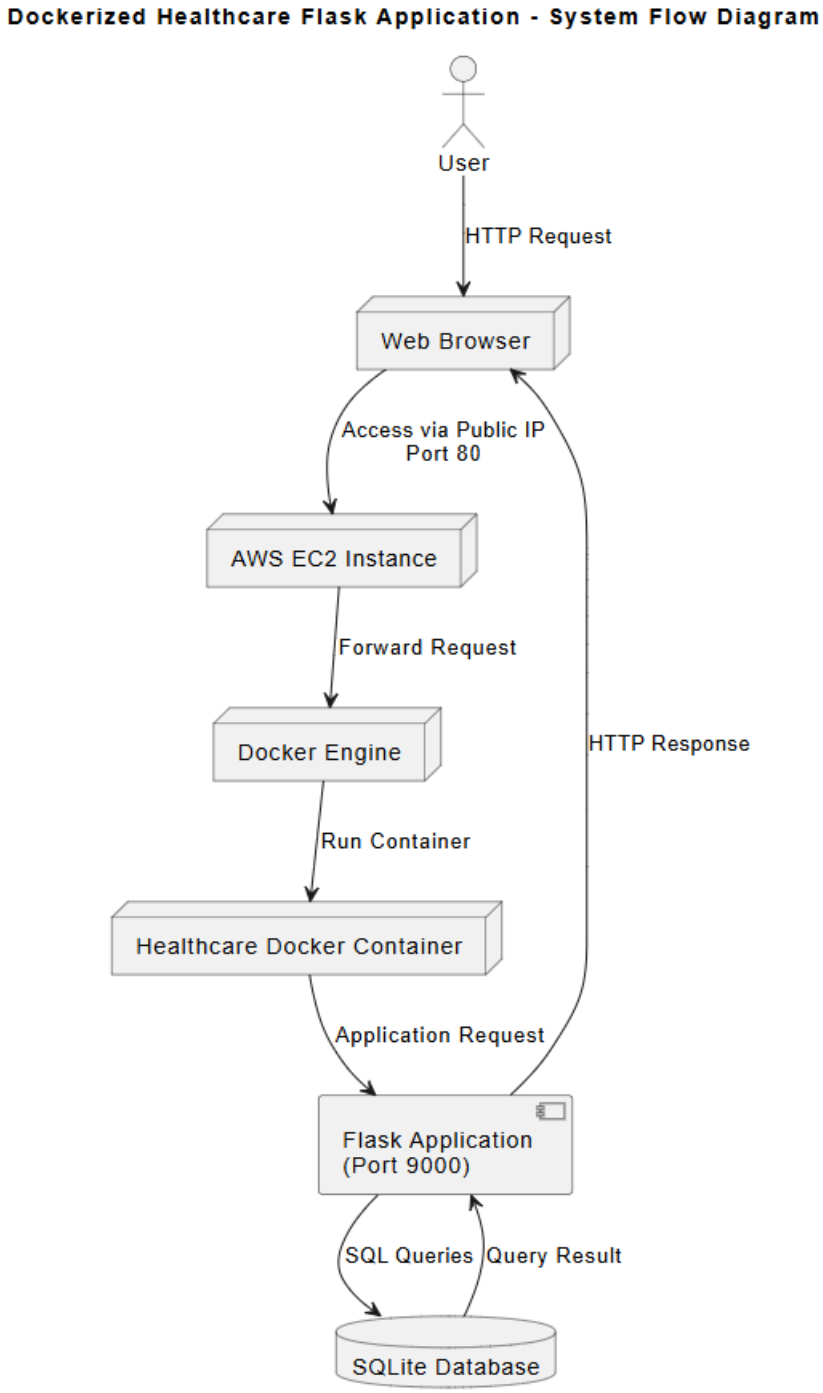
Proposed Solution

1. **Key Features:**
   - Docker-based containerization
   - Multi-stage optimized Docker build
   - Docker Compose orchestration
   - Three-tier architecture (Frontend, Backend, Database)
   - RESTful API for patient management (CRUD operations)
   - SQLite database with Docker volume persistence
   - Deployment on AWS EC2
   - Secure access using AWS Security Groups and SSH keys
   - Portable and environment-independent setup
   - Lightweight and easy-to-deploy architecture

2. **Overall Architecture/Workflow:** The system follows a container-based three-tier architecture deployed on an AWS EC2 instance.

   Architectural Flow:

   User Browser
   ↓
   Internet
   ↓
   AWS EC2 Linux Instance
   ↓
   Docker Engine
   ↓
   Flask Application (Docker Container – Port 9000)
   ↓



Workflow:

- User accesses the application via EC2 Public IP.
- Frontend sends an HTTP request to the Flask backend.

- Flask processes the request and interacts with SQLite.
- Database performs CRUD operations.
- Response is returned and displayed to the user.

The architecture ensures portability, consistency, and secure cloud deployment.

ER Diagram

Entities

Patient

patient_id (Primary Key)

name

age

gender

contact

Doctor

doctor_id (Primary Key)

name

specialization

Appointment

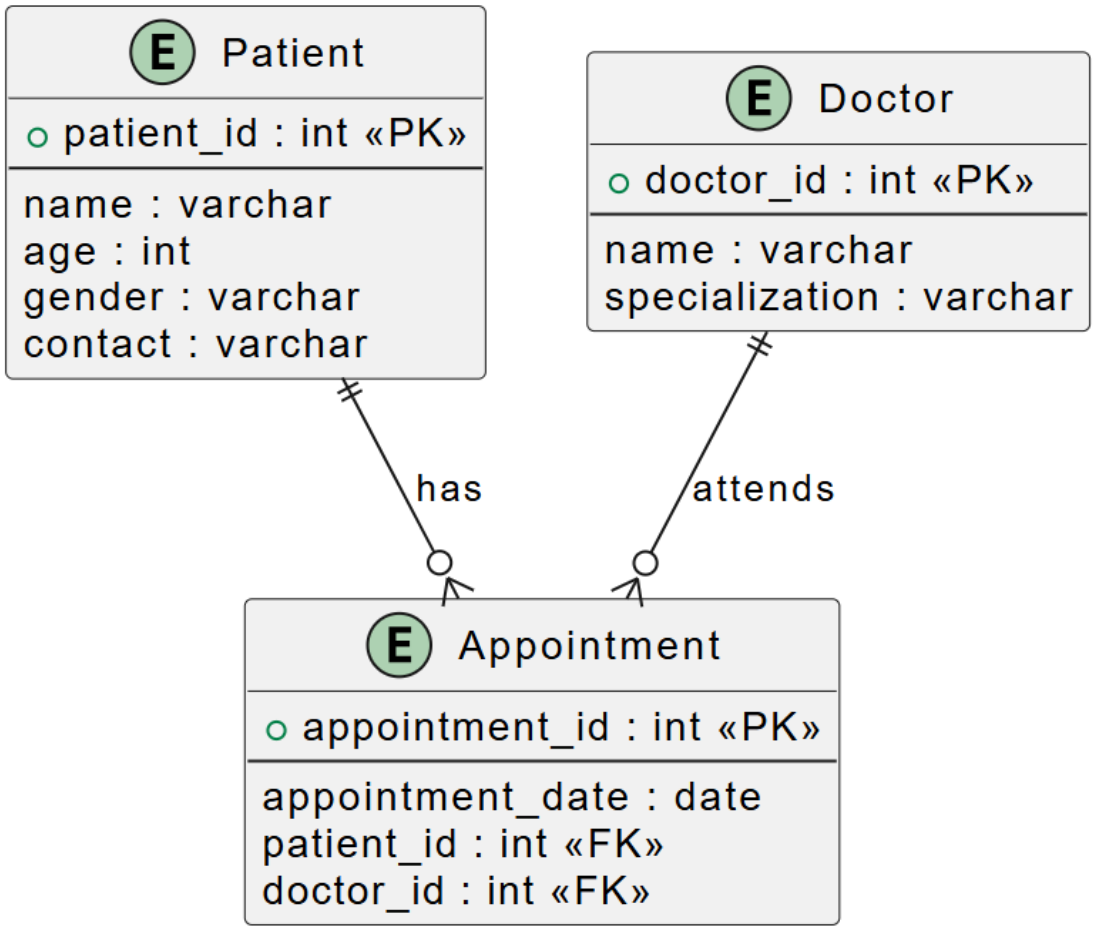
appointment_id (Primary Key)

patient_id (Foreign Key)

doctor_id (Foreign Key)

appointment_date

Healthcare Management System - ER Diagram



3. **Tools and Technologies Used:**
   - **Programming Language:** Python
   - **Framework:** Flask
   - **Frontend:** HTML,CSS
   - **Database:** SQLite
   - **Containerization:** Docker(Multi Stage-Build)
   - **Orchestration:** Docker Compose
   - **Cloud Platform:** AWS EC2
   - **Infrastructure Provisioning:** Terraform
   - **Version Control:** Git

- **Operating System:** Linux(EC2 Instance)

Python Flask

Flask is a lightweight Python web framework used to build RESTful web applications. It is simple, flexible, and suitable for small to medium-scale applications.

SQLite Database

SQLite is a file-based database that does not require a separate server. It is lightweight and ideal for small healthcare systems and learning projects

Docker

Docker is used to package the application and its dependencies into a container, ensuring consistent behavior across environments.

Docker Compose

Docker Compose is used to define and manage multi-container applications. It allows starting and stopping services using a single command.

AWS EC2

Amazon EC2 provides virtual Linux servers on the cloud where the application is deployed.

AWS IAM

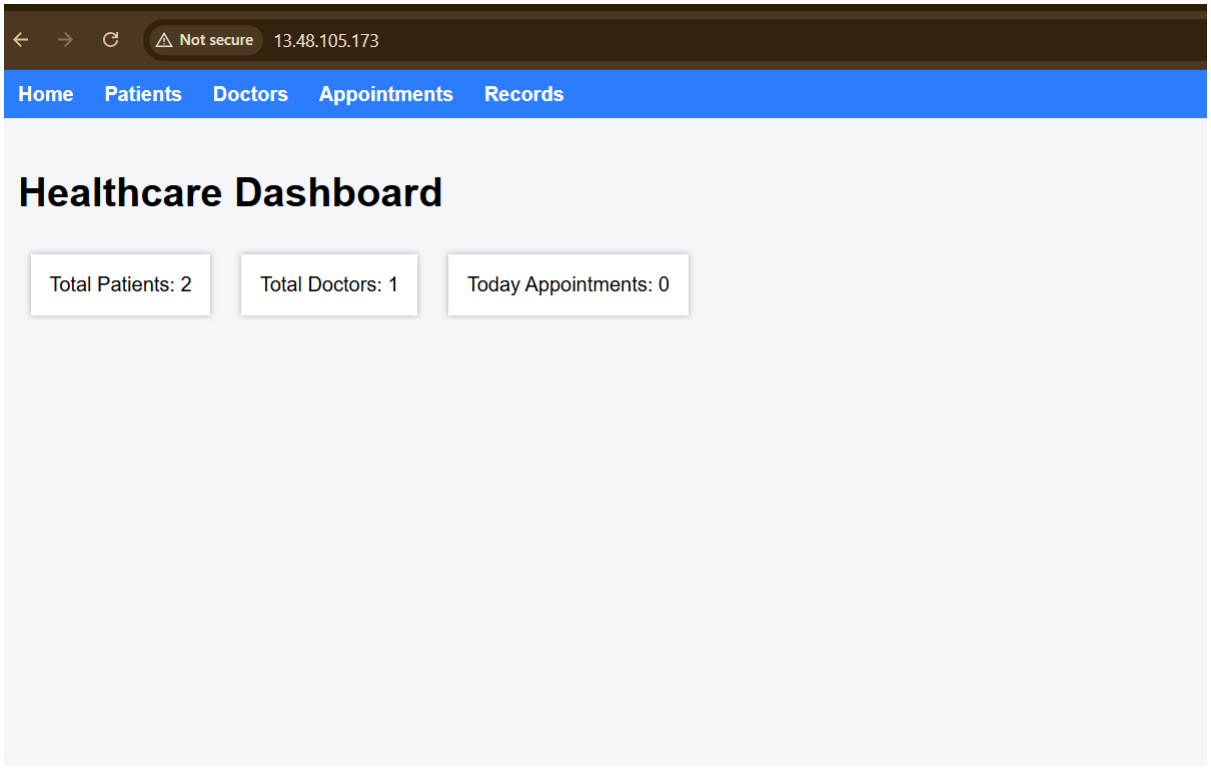
IAM is used to securely manage access to AWS resources using access keys and permissions.

Terraform

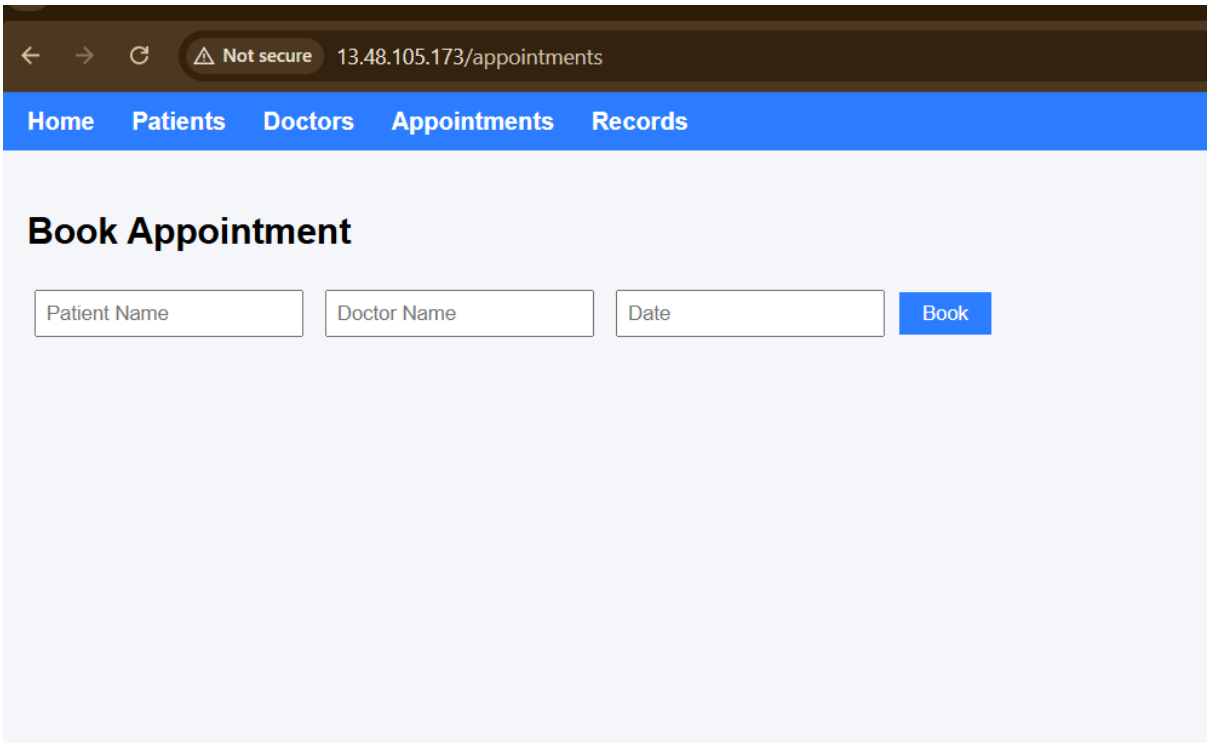
Terraform is used to create cloud infrastructure using code, enabling automation and repeatability.

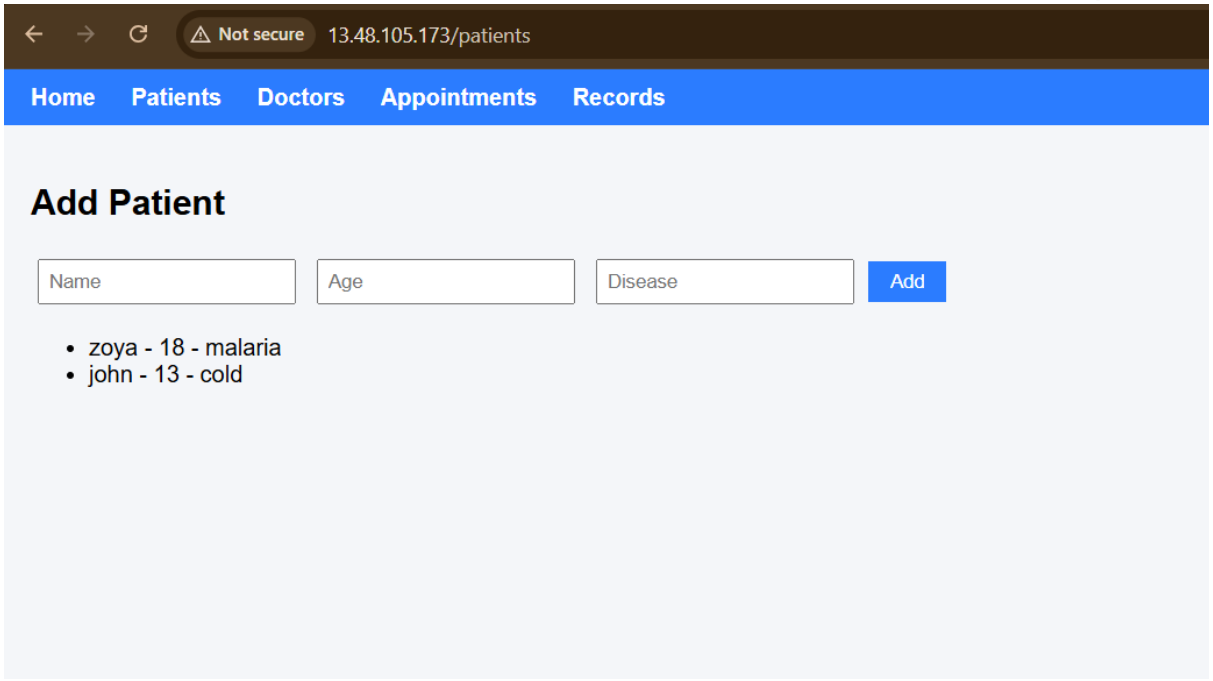
Results & Output

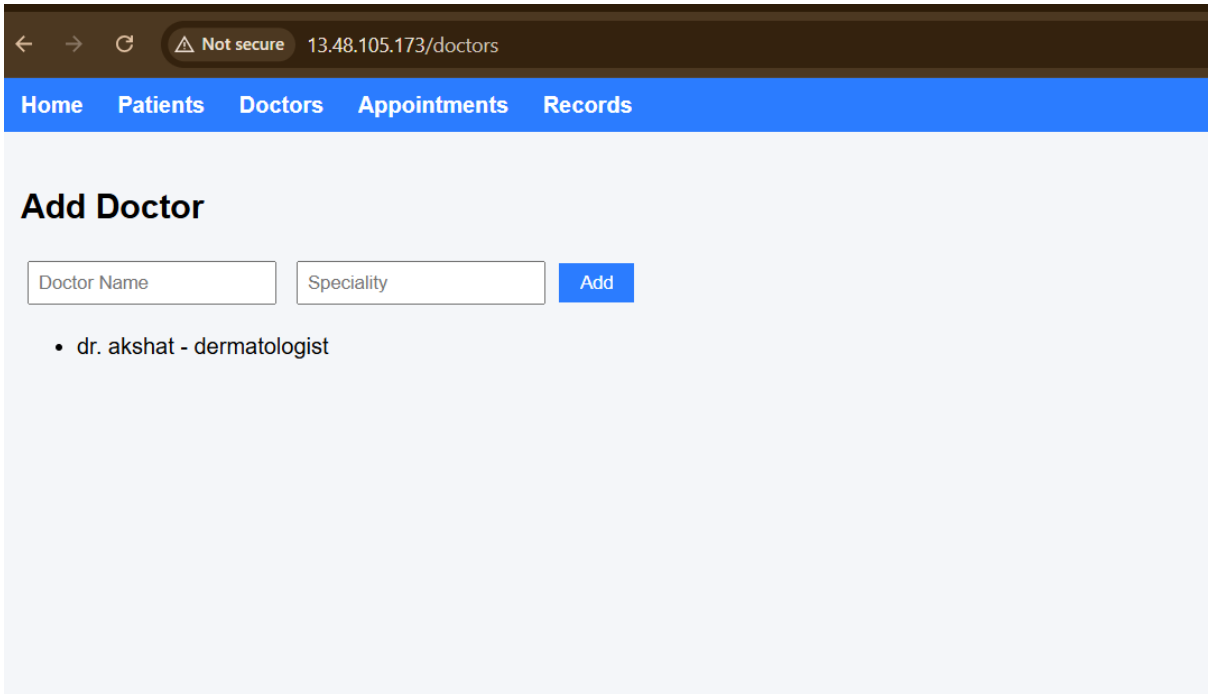
1. Screenshots/Output:

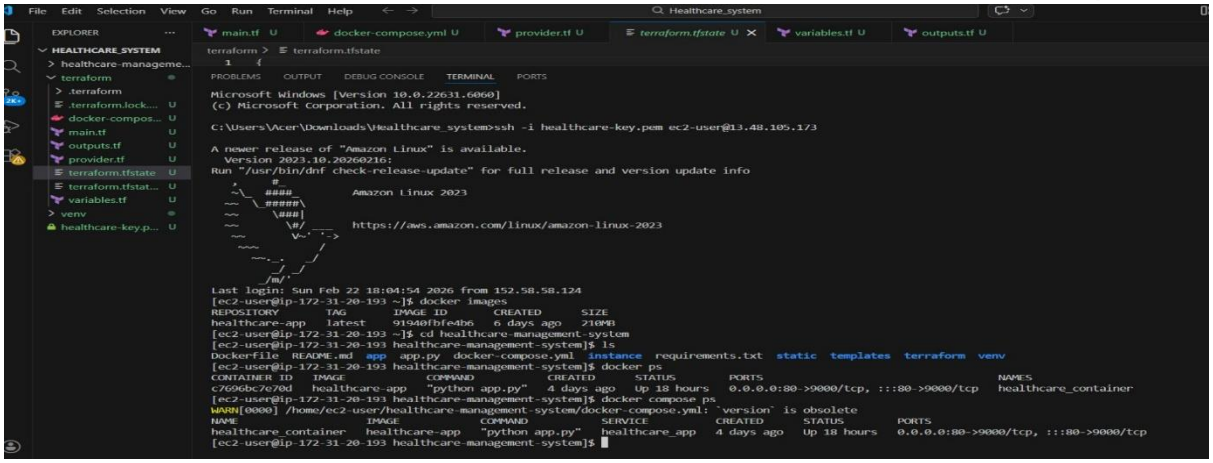


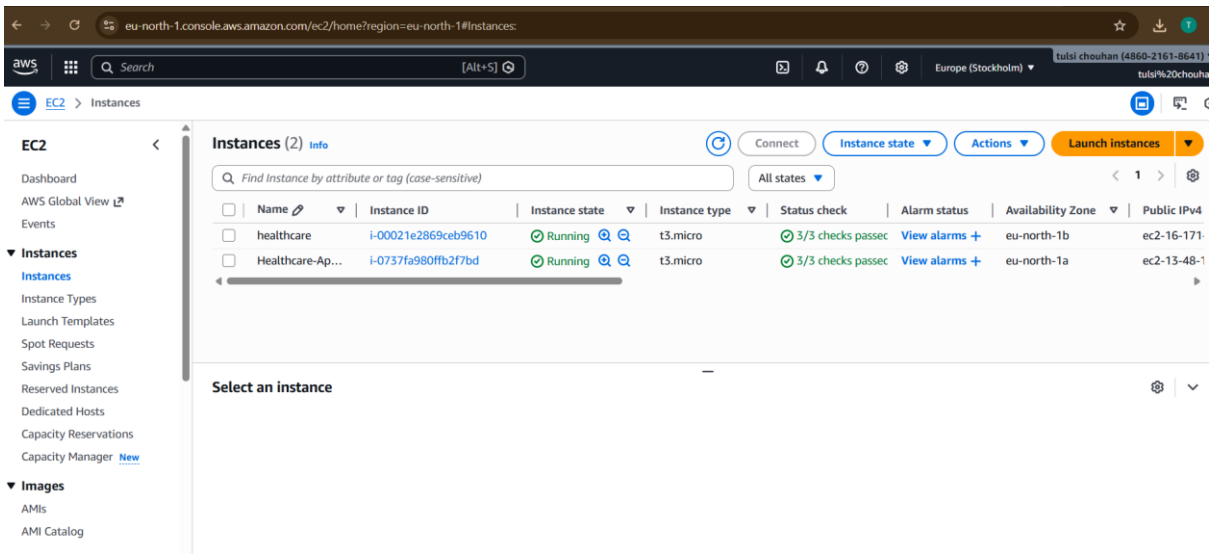
2.




Not secure
13.48.105.173/patients
Home
Patients
Doctors
Appointments
Records
Add Patient
Name
Age
Disease
Add
zoya - 18 - malaria
john - 13 - cold


Not secure
13.48.105.173/doctors
Home
Patients
Doctors
Appointments
Records
Add Doctor
Doctor Name
Speciality
Add
dr. akshat - dermatologist





2. Report/Dashboard/Models:

A. Report:

- Patient records listing (All Patients report)
- Individual patient detail view

- CRUD operation status responses (Add, Update, Delete confirmation)

B. Dashboard:

- Web-based interface displaying patient data
- Simple HTML/CSS frontend
- Displays real-time data fetched from backend APIs

C. Models:

- patient_id (Primary Key)
- name
- age
- disease

3.Key Outcomes:

- Successful containerization of the Healthcare Flask application using Docker
- Implementation of multi-stage Docker build for optimized image size
- Simplified deployment and management using Docker Compose
- Persistent data storage using Docker volumes
- Successful deployment on AWS EC2 cloud environment
- Achieved portability between local and cloud setups

- Improved reliability and environment consistency
- Enhanced understanding of real-world DevOps and containerization practices

Conclusion:

The Dockerized Healthcare Python Flask Service successfully demonstrates a modern container-based deployment approach using Docker and Docker Compose. The project ensures portability, consistency, and reliability by eliminating environment-specific dependency issues.

By deploying the application on AWS EC2 and implementing multi-stage Docker builds, the solution achieves optimized performance, improved security, and efficient resource utilization.

Overall, the project reflects practical DevOps fundamentals, including containerization, cloud deployment, and infrastructure management, making it a scalable and maintainable solution for healthcare application deployment.

Future Scope & Enhancement:

- Migration from SQLite to MySQL or PostgreSQL for better scalability
- Implementation of CI/CD pipeline using Jenkins or GitHub Actions
- Deployment using Kubernetes for container orchestration at scale
- Integration of Redis for caching and performance improvement
- Implementation of authentication and role-based access control
- Enabling HTTPS using SSL/TLS certificates
- Monitoring and logging using tools like Prometheus and Grafana
- Auto-scaling and load balancing for high availability
- Backup and disaster recovery mechanisms
- UI enhancement with modern frontend frameworks (React/Angular)

Problem Statement & Objectives

1. Problem Statement
2. Project Objectives
3. Scope of the Project

Proposed Solutionures *(Just mention key features here no need to go into details)*

1. Overall Architecture / Workflow
2. Tools & Technologies Used (*If applicable)*

Results & Output

Add the below details here:

1. *Screenshots / outputs*
2. *Reports / dashboards / models*
3. *Key outcomes*

Conclusion

Mention a brief conclusion about your project summing up everything you have worked on and the key learning.

Future Scope & Enhancements-

* Integration of *AI* for smart diagnosis and decision support (like systems used by IBM Watson Health).

* Addition of *telemedicine* and online video consultation.

* Use of *cloud platforms* such as Amazon Web Services for secure data storage and backup.

* Development of a *mobile app* for easy appointment booking and report access.

* Integration with *wearable devices* for real-time health monitoring.

* Implementation of *blockchain* for better data security.

* Integration with government schemes like Ayushman Bharat for easy insurance processing.