CSI3007 - ADVANCED PYTHON PROGRAMMING

LAB ACTIVITY – 21

Web Deployment (Render)

NAME : SONALI R

REG NO : 22MID0061

Goal: To establish a Continuous Deployment (CD) pipeline by pushing the containerized image to a registry (Docker Hub) and deploying it to a public cloud platform (Render).

Project Overview

Title – SERVICE MIND AI

Aim: ServiceMind-AI is an intelligent customer chatbot application designed to assist users with various services by handling multiple types of customer inquiries (Service Inquiry, Cancel Request, Reschedule Request, Working Hours Inquiry, Pricing Inquiry, Greeting Response) and providing relevant information in real time.

Flask Framework Suitability:

- Lightweight and Minimalist. Flask avoids forcing large libraries or complex structures, allowing you to choose exactly what you need (like just one file, app.py).
- Dynamic HTML Generation. It integrates easily with Jinja2, allowing you to seamlessly inject Python data (like quiz questions or table rows) into your HTML files.

Project Structure:

```
CHATBOT/
├── templates/
   │-- chatbot.html     # HTML file for the chatbot interface
│-- chatbot.py          # Main Python script for chatbot logic
│-- config.py           # Configuration file for chatbot settings
│-- main.py             # Entry point for running the chatbot application
│-- test_main.py        # Unit tests for chatbot functionalities
│-- validators.py       # Python script for input validation
│-- requirements.txt    # List of dependencies for the project
```

```
│
├── CustomerChatbot/      # Virtual environment and dependencies
│
├── static/
   ├── Images/            # Directory containing image assets
   │-- chatbotScript.js   # JavaScript file for chatbot interactions
   │-- chatbotStyles.css  # CSS file for chatbot styling
│
└── user_data.xlsx        # Excel file containing user data
```

Github Link for Project: https://github.com/SonaliRajaram/ServiceMind-AI

Build Process:

1. Deployment Overview: To take the locally verified Docker image and deploy it publicly using an external container hosting service (Render).

2. Image Registry (Docker Hub):

- **docker login (Mentioning PAT for security).**

```
PS C:\Users\sonal\OneDrive\Desktop\Chatbot> docker login
Authenticating with existing credentials... [Username: sonali018]

i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
```

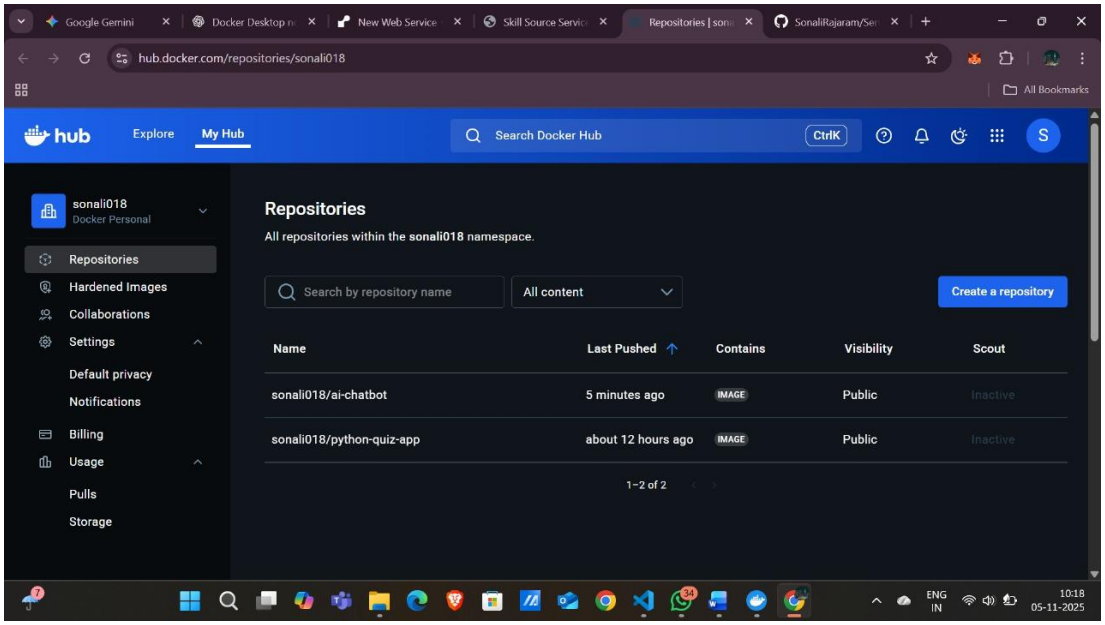
- **Pushing the image in the Docker Hub:**
  - docker tag ai-chatbot sonali018/ai-chatbot:latest

```
PS C:\Users\sonal\OneDrive\Desktop\Chatbot> docker tag ai-chatbot sonali018/ai-chatbot:latest
```

  - docker push sonali018/ai-chatbot:latest

```
PS C:\Users\sonal\OneDrive\Desktop\Chatbot> docker push sonali018/ai-chatbot:latest
The push refers to repository [docker.io/sonali018/ai-chatbot]
65f442233519: Pushed
cfba6d6670cc: Layer already exists
a09dc670095e: Layer already exists
d7ecded7702a: Layer already exists
3874ccec91a1: Pushed
0ddb35ad4c77: Layer already exists
03b7f5f69b04: Layer already exists
7ae18f681ca1: Layer already exists
129d1061b738: Layer already exists
d7796e8184d1: Already exists
latest: digest: sha256:9b84a800874ccf33b2c3836bc5ef5e11d08ddc854547111e24a0d9456f3eba5d size: 856
PS C:\Users\sonal\OneDrive\Desktop\Chatbot>
```

Docker Hub Snapshot with Loaded Image:



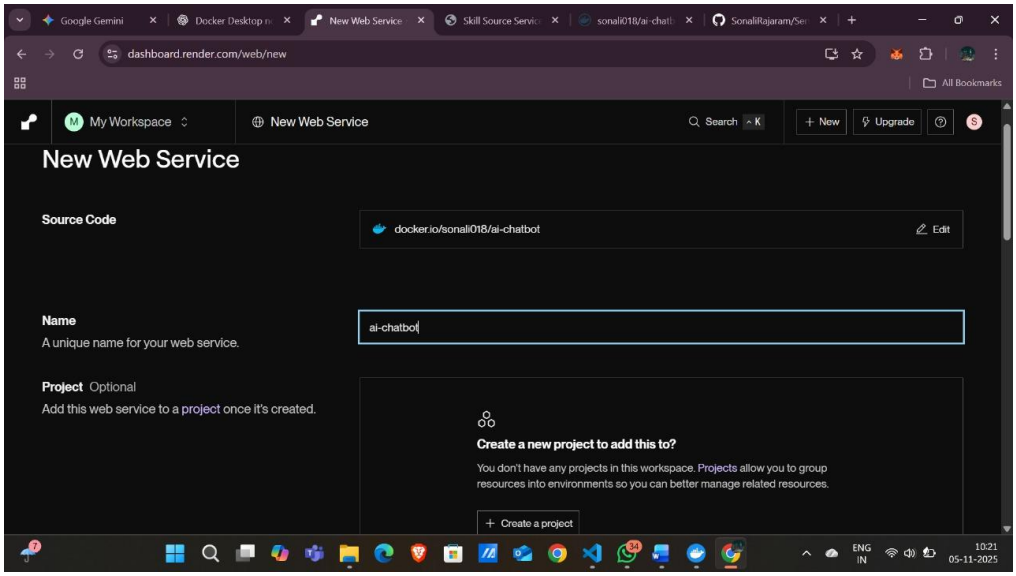
3. Deployment Platform: Render

1. Container Native: Render is built to pull my Docker Hub image and run it as a continuous Web Service, just like you ran it locally on Docker Desktop.
2. Statefulness: Because the Render container is running 24/7, the Flask application maintains the session data, allowing users to complete the multi-step quiz seamlessly.
3. Simplicity: The deployment process is a simple, direct connection using the exact image URL you pushed to Docker Hub, requiring minimal configuration beyond setting the port to 8000.

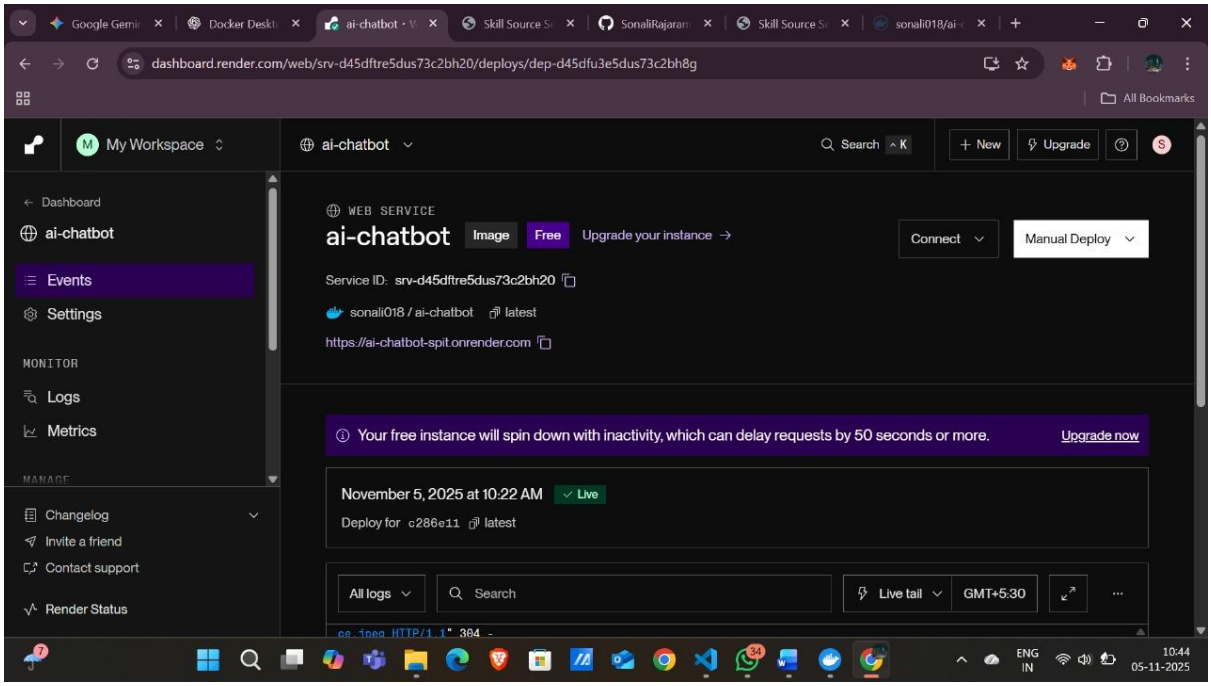
Render is a hosting environment for containers, making it the native and simplest choice for your Dockerized Flask application.

4. Procedure: Render Configuration

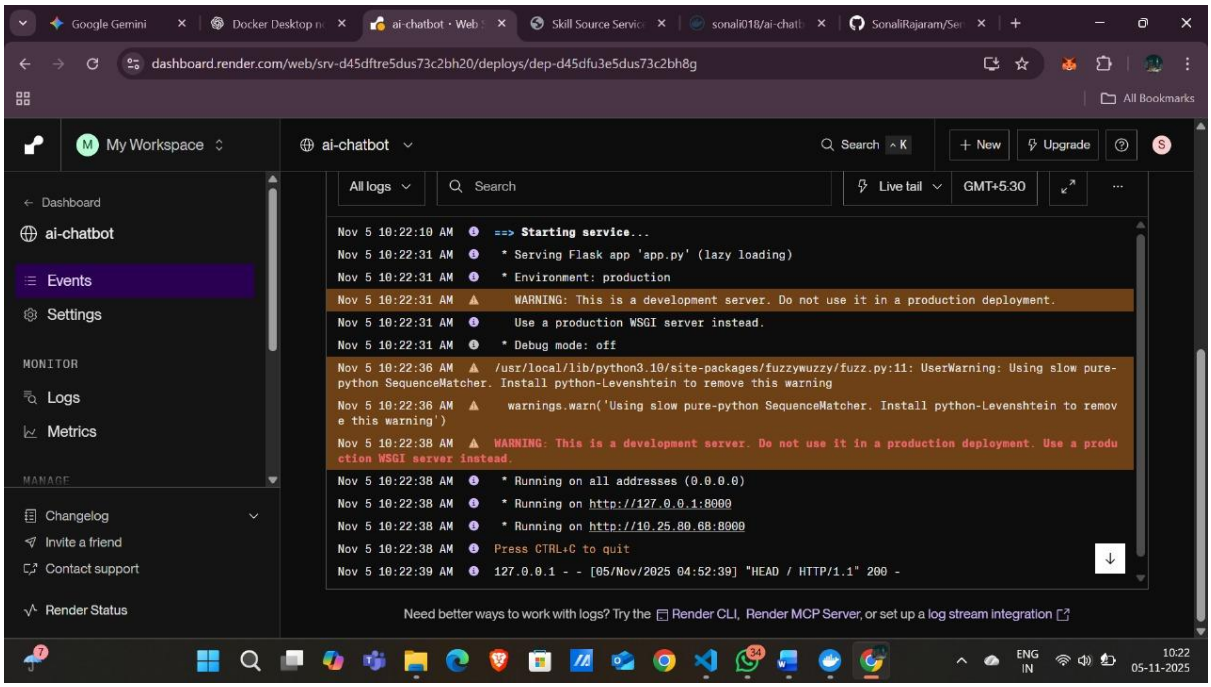
- Creating New Web Services:



- Specifying the image URL: docker.io/sonali018/ai-chatbot:latest.



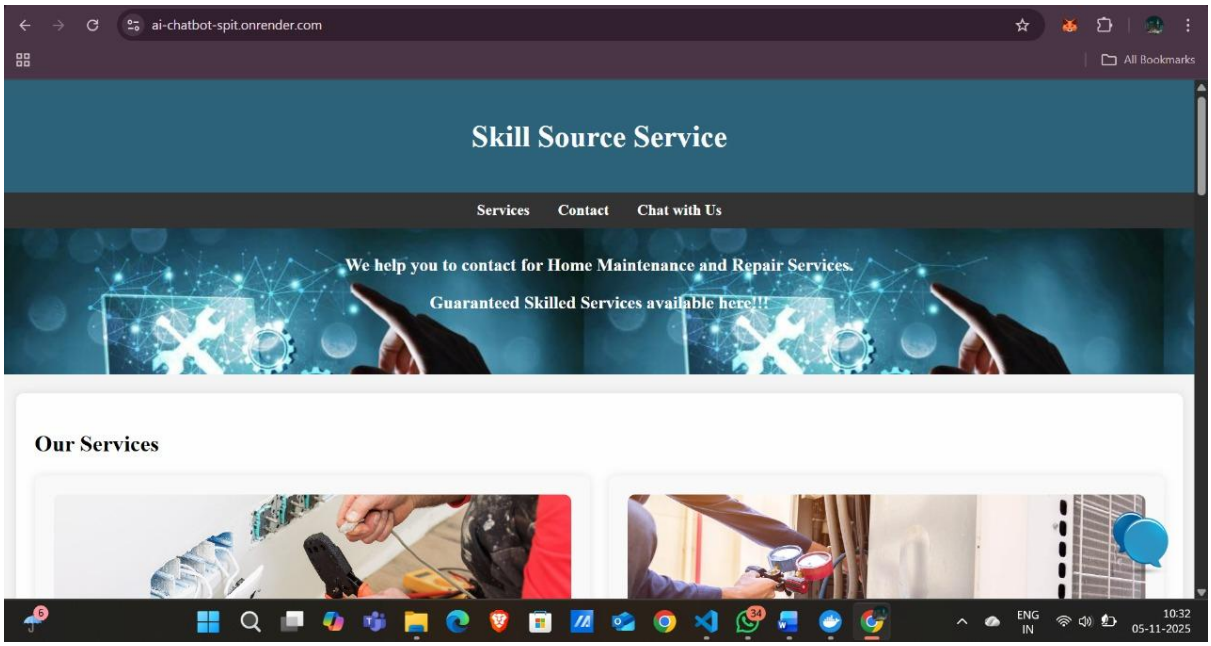
- Event Running Verification:

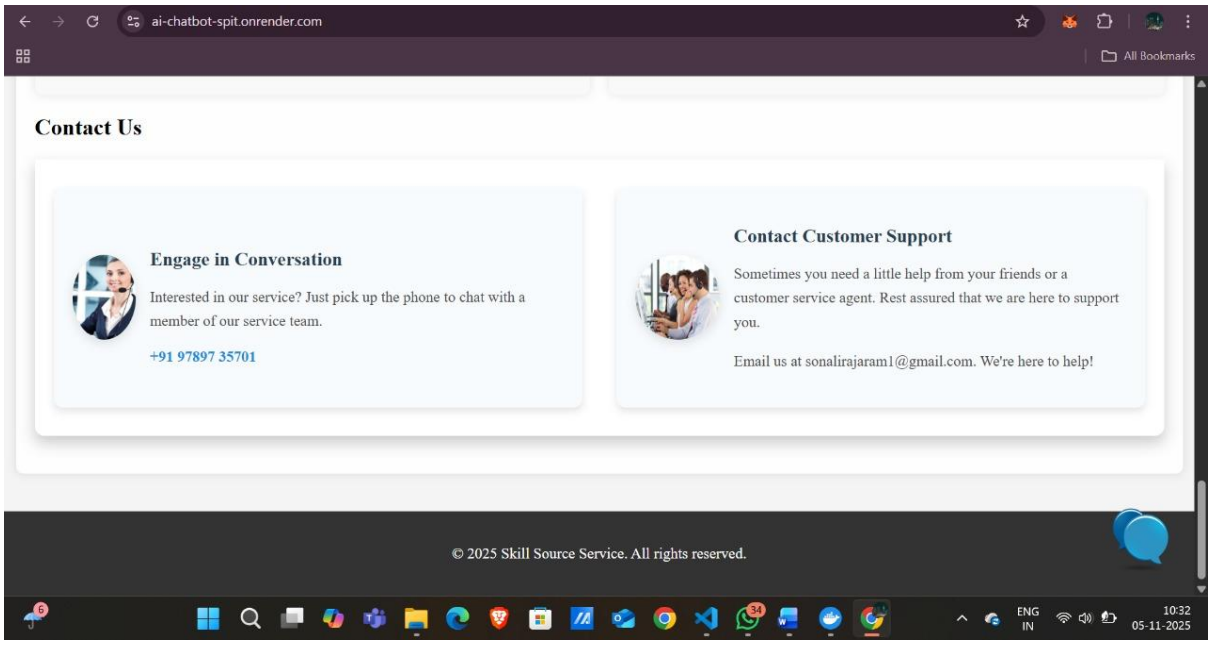


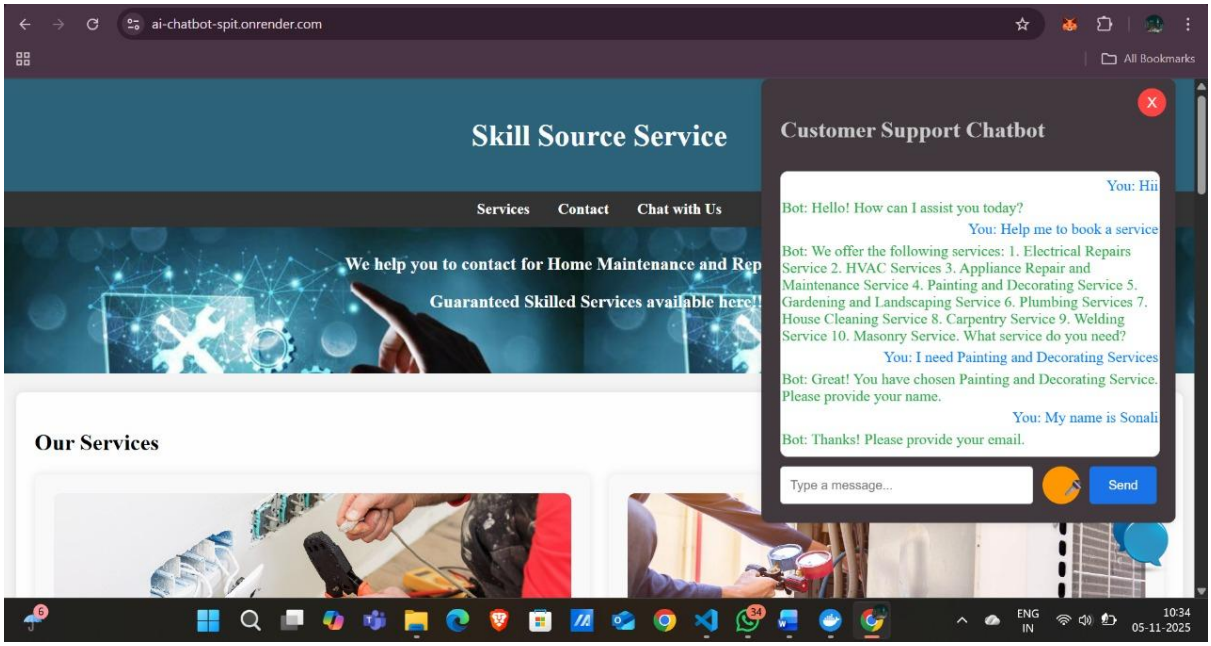
5. Usage & Final Outcome

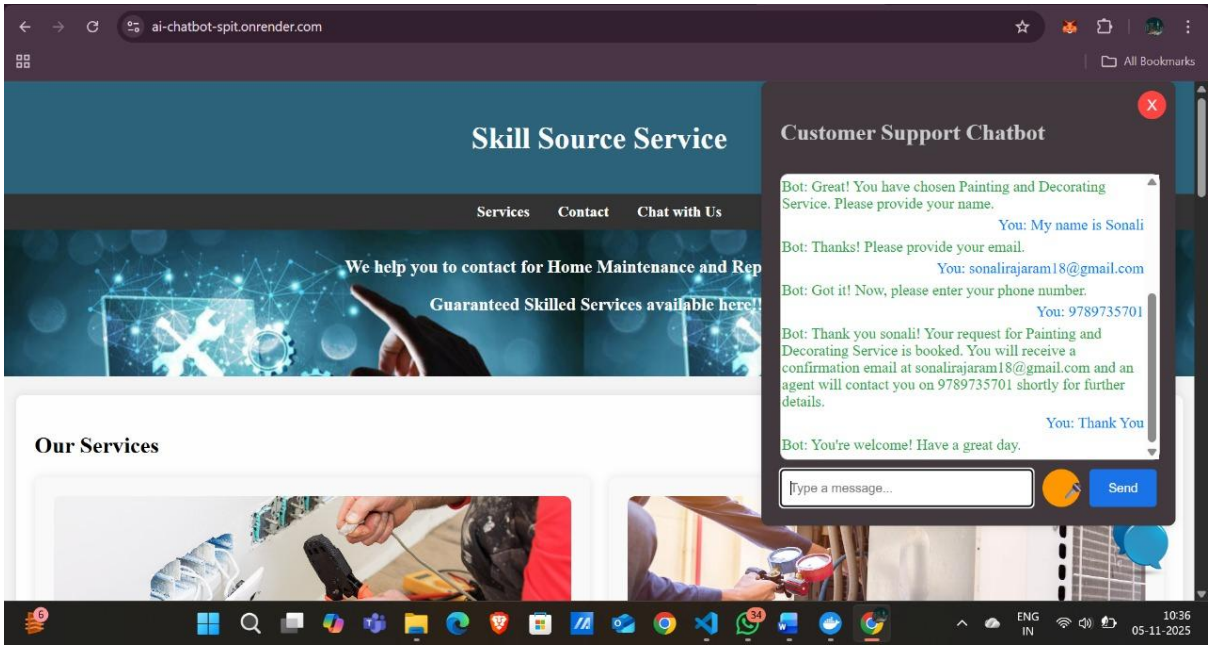
public URL: https://ai-chatbot-spit.onrender.com/

Skill Source Service website and chatbot conversation snapshots:

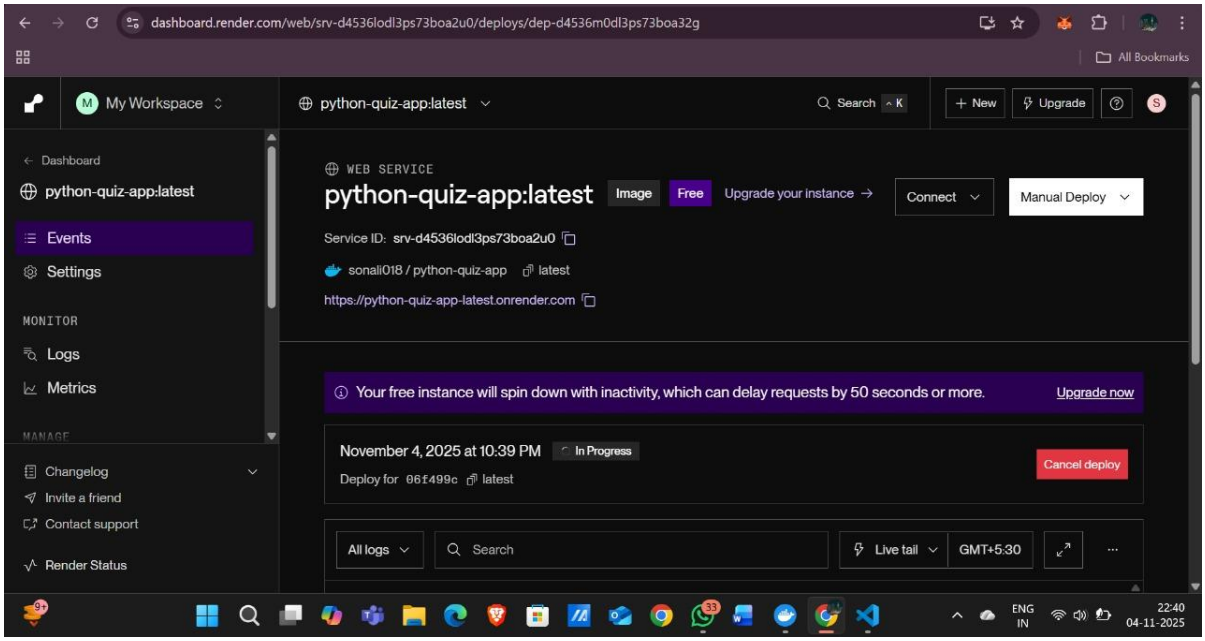




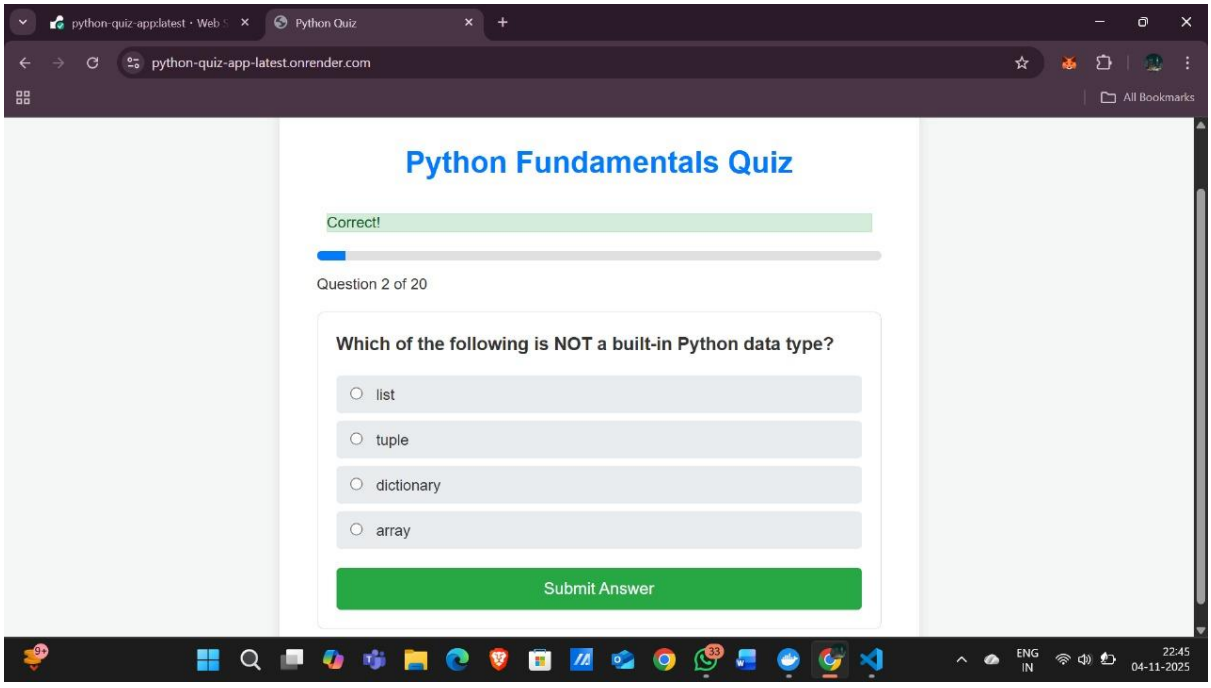

ai-chatbot-spit.onrender.com
All Bookmarks
Skill Source Service
Services
Contact
Chat with Us
We help you to contact for Home Maintenance and Rep
Guaranteed Skilled Services available here!!
Our Services
Customer Support Chatbot
You: Hii
Bot: Hello! How can I assist you today?
You: Help me to book a service
Bot: We offer the following services: 1. Electrical Repairs Service 2. HVAC Services 3. Appliance Repair and Maintenance Service 4. Painting and Decorating Service 5. Gardening and Landscaping Service 6. Plumbing Services 7. House Cleaning Service 8. Carpentry Service 9. Welding Service 10. Masonry Service. What service do you need?
You: I need Painting and Decorating Services
Bot: Great! You have chosen Painting and Decorating Service. Please provide your name.
You: My name is Sonali
Bot: Thanks! Please provide your email.
Type a message...
Send
ENG
IN
10:34
05-11-2025

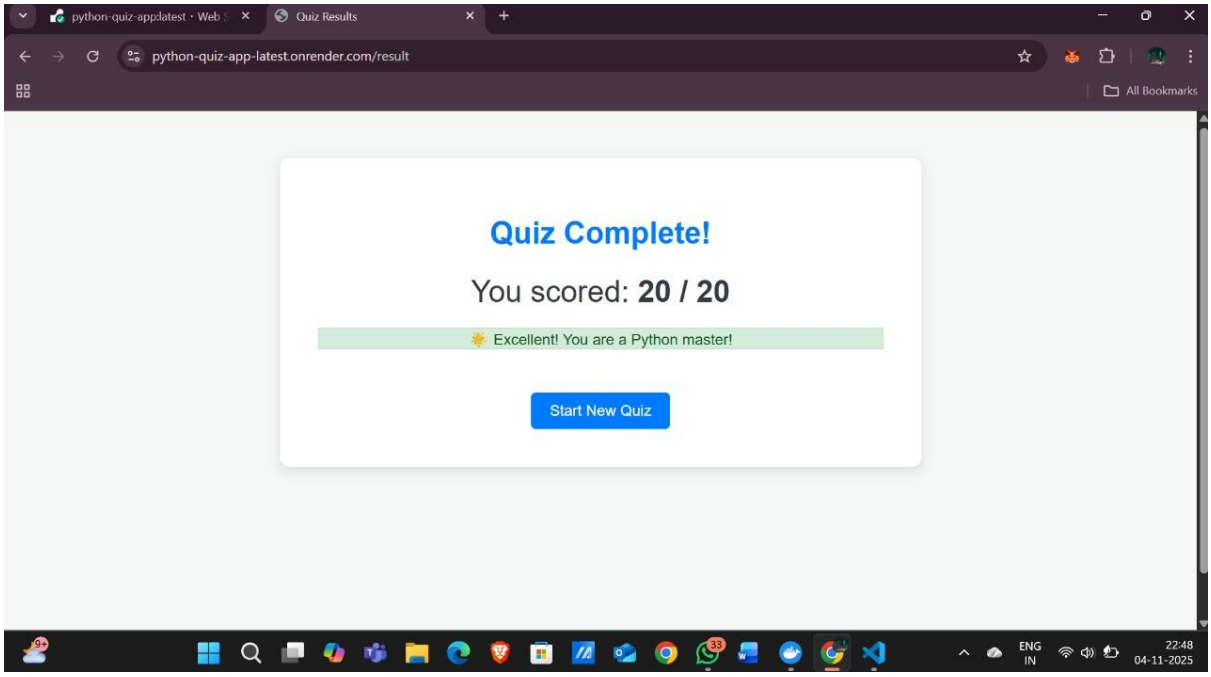

ai-chatbot-spit.onrender.com
All Bookmarks
Skill Source Service
Services
Contact
Chat with Us
We help you to contact for Home Maintenance and Rep
Guaranteed Skilled Services available here!!
Our Services
Customer Support Chatbot
Bot: Great! You have chosen Painting and Decorating Service. Please provide your name.
You: My name is Sonali
Bot: Thanks! Please provide your email.
You: sonalirajaram18@gmail.com
Bot: Got it! Now, please enter your phone number.
You: 9789735701
Bot: Thank you sonali! Your request for Painting and Decorating Service is booked. You will receive a confirmation email at sonalirajaram18@gmail.com and an agent will contact you on 9789735701 shortly for further details.
You: Thank You
Bot: You're welcome! Have a great day.
Type a message...
Send
ENG
IN
10:36
05-11-2025

Python Fundamentals Quiz App Web Deployment:



Public URL: https://python-quiz-app-latest.onrender.com/





Conclusion: The Containerized DevOps Cycle

The deployment of the AI-Chatbot App to the Render cloud platform successfully demonstrated a complete, modern, container-centric DevOps cycle:

The Complete Pipeline

The project transitioned through four distinct and critical stages:

1. Code (Local Development): The Flask application code and necessary configurations (session logic, requirements.txt) were finalized.
2. Docker (Containerization): The Dockerfile packaged the application and its environment (python:3.10-slim, Flask) into a single, portable image, standardizing the runtime environment.
3. Docker Hub (Registry): The image was pushed to the registry, making the finalized, tested artifact available for external cloud services.
4. Render (Execution/Deployment): Render pulled the image from Docker Hub and ran it as a continuous Web Service, exposing the application to the public internet via a dedicated URL.

Advantages of Containerized Web Deployment

This workflow offers significant advantages over traditional bare-metal or virtual machine deployments:

- Speed and Efficiency: Deployment is fast because Render only needs to pull the pre-built, ready-to-run image, avoiding the slow process of installing dependencies and setting up the Python environment on the server itself.

- Zero Dependency Issues: By relying on the Docker image, the application is guaranteed to work in the cloud, eliminating the classic "It worked on my machine!" problem, as the entire local environment is replicated exactly in the cloud environment.
- Portability: The application is no longer tied to any single vendor. The same image that runs on Docker Desktop can now run easily on Render, Google Cloud Run, AWS, or any other container platform, offering maximum flexibility.
- Isolation and Security: The application runs in isolation from other services on the Render platform, reducing conflicts and increasing security.

In conclusion, leveraging Docker and Docker Hub streamlined the transition from local development to scalable, robust cloud hosting on Render, proving the efficiency and reliability of a container-based deployment strategy.

****