Progress Report on AI-Driven Proactive Maintenance API Deployment

Introduction

This report's goal is to provide an overview of the developments made in the Flask deployment of the AI-Driven Proactive Maintenance API. This project aims to improve proactive maintenance techniques by offering a dependable way to forecast equipment failure based on IoT data.

Deployment Summary

I successfully deployed the Flask application on PythonAnywhere. Below are the key steps taken during the deployment process:

Account Creation: Created a free account on PythonAnywhere.

Web App Setup:

- Navigated to the "Web" tab and added a new web app.
- Selected the Flask framework and specified the Python version.

Code Upload:

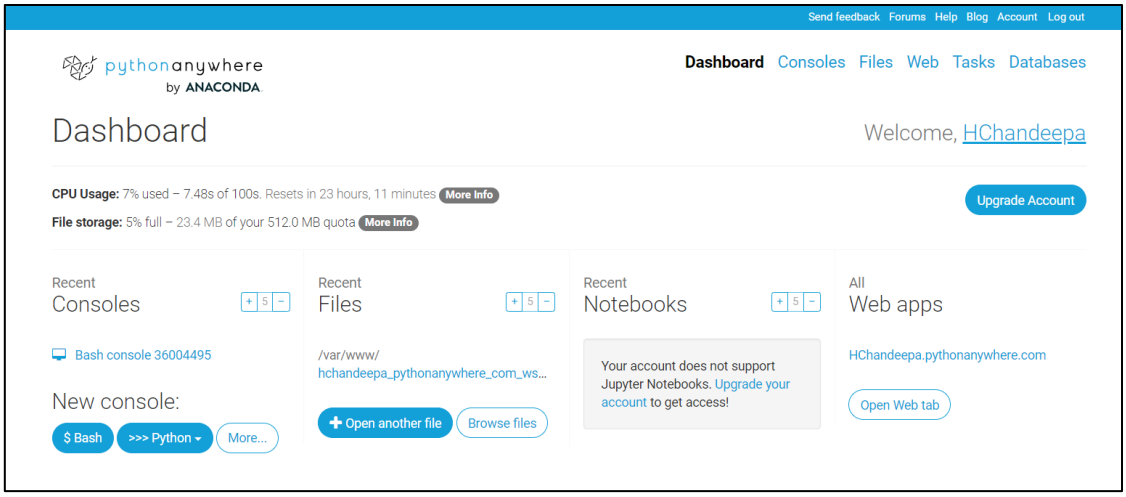
- Uploaded the app.py file and the random_forest_model.pkl model file.

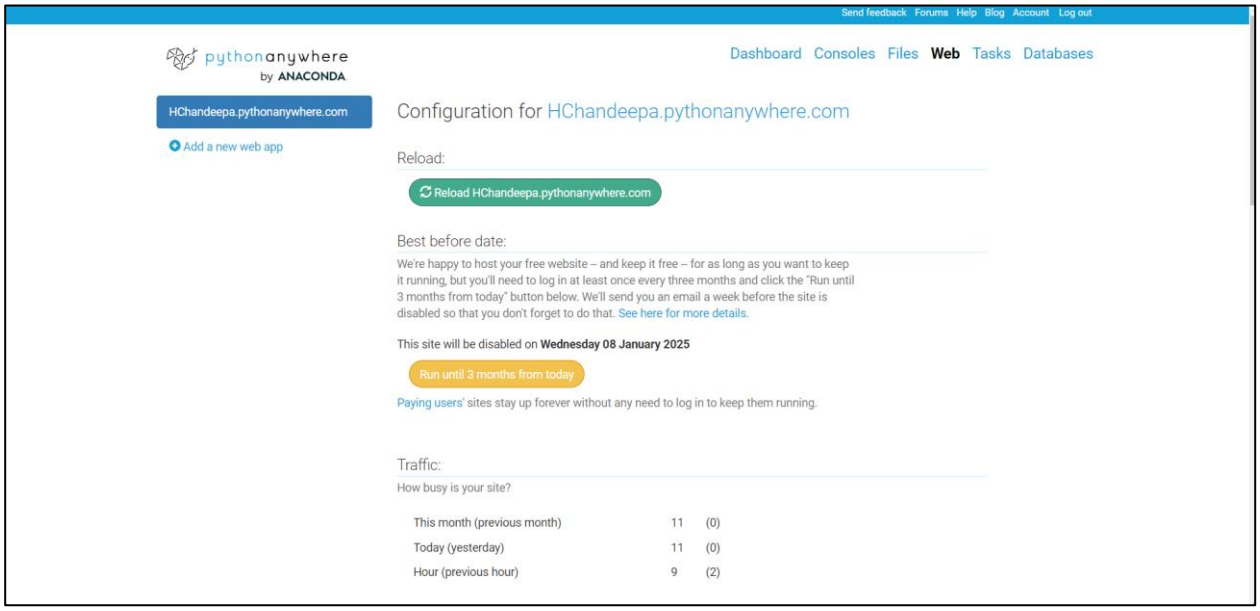
Configuration:

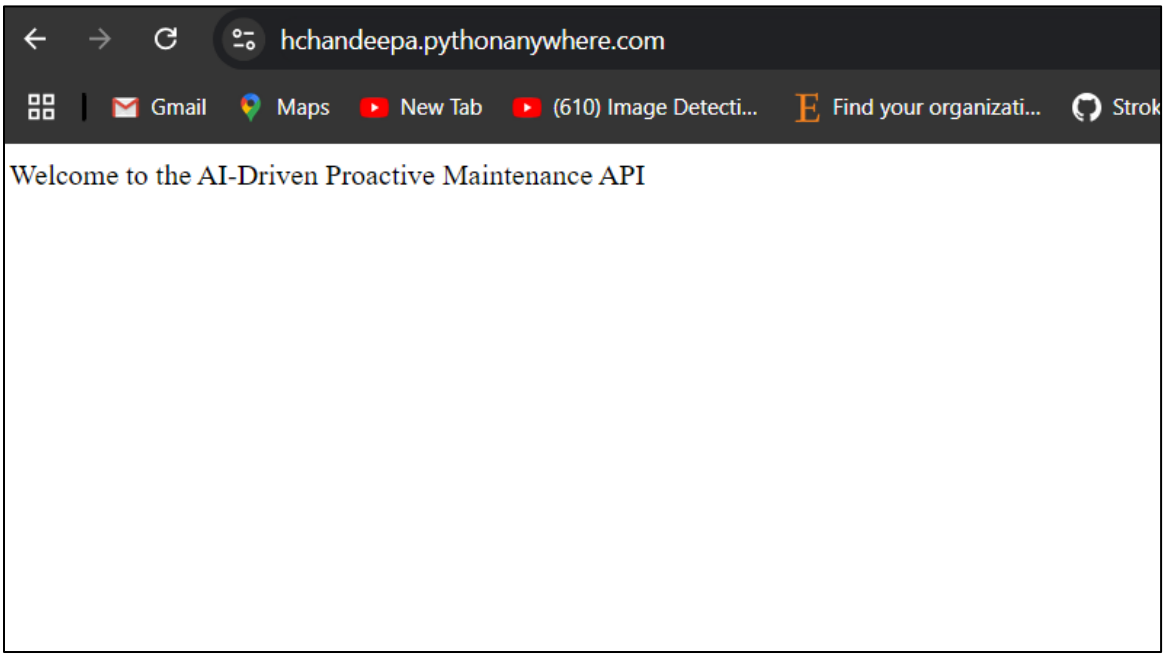
- Ensured that the application was configured to run on the correct port and settings.

Testing:

- Verified the application was running by accessing the root URL, which displayed a welcome message.







API Functionality

The API provides an endpoint for making predictions regarding equipment failure based on the following features: Temperature, Humidity, and HVAC_Status.

Endpoint Details

- **Root Route**: GET /
    - Returns a welcome message.
- **Prediction Route**: POST /predict

- **Input**: JSON format containing the features:

{

"Temperature": 72,

"Humidity": 50,

"HVAC_Status": 1

}

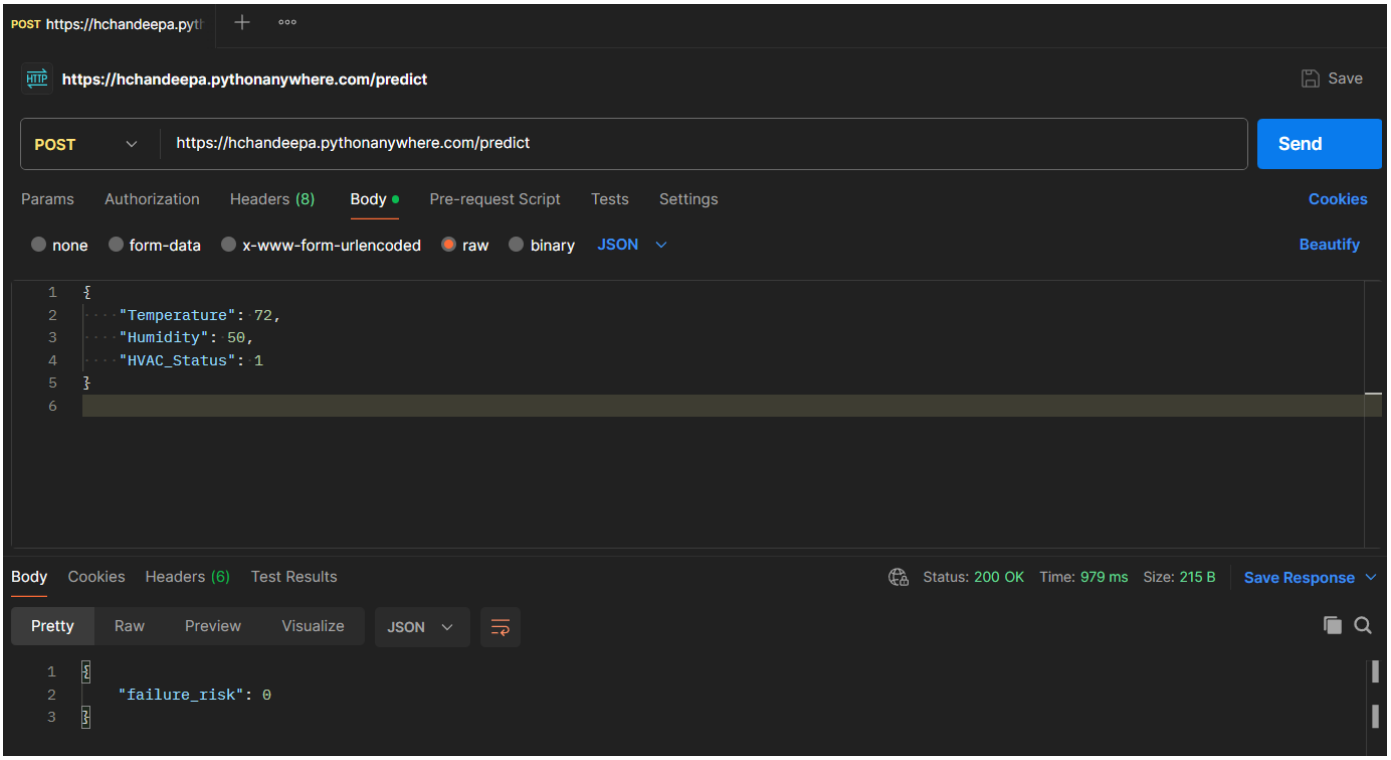
- **Response**: JSON indicating the risk of failure:

{

"failure_risk": 0 }

Testing

The API has been tested successfully using Postman, which confirmed that it responds correctly to valid input.



Connecting to IoT

The Flask API can be integrated with IoT devices to receive real-time data for making predictions. For example, a sample code snippet for sending data from an IoT device might look like this:

```
import requests


# Sample IoT data
data = {
   "Temperature": 72,
   "Humidity": 50,
   "HVAC_Status": 1
}


response = requests.post('https://hchandeepa.pythonanywhere.com/predict', json=data)
print(response.json())
```

This connection allows for immediate prediction based on live data, facilitating proactive maintenance actions.

Next

- Test the API with actual IoT data to ensure robustness and accuracy.
- Implement additional features such as data logging and user authentication for enhanced security.
- Monitor the API's performance and make necessary adjustments to improve efficiency.