**DevOps Assignment Report**

**Name:** Arpit Gupta  
**Roll Number:** 21ucs030  
**Assignment:** DevOps Qoala Assignment

**1. Issues Identified**

During the process of building Docker images and launching the containers, the following issues were identified:

1. **Build Errors in Dockerfiles:**
   * **Issue:** The Dockerfiles for the application services had syntax issues, such as incorrect paths or missing dependencies.
   * **Resolution:** I modified the paths in the Dockerfiles and added necessary dependencies (like installing additional packages) to resolve these errors.
2. **Misconfigured Ports in docker-compose.yml:**
   * **Issue:** The application was not accessible on [http://localhost](http://localhost/) because of incorrect port bindings in docker-compose.yml.
   * **Resolution:** I corrected the ports in the docker-compose.yml file to ensure each container exposed the required ports, particularly port 80 for Nginx.
3. **Network Issues Between Containers:**
   * **Issue:** Some services could not connect to the database due to misconfigured network settings.
   * **Resolution:** I verified the container networks and updated the docker-compose.yml file to ensure all containers could communicate through the same bridge network.
4. **Missing Environment Variables:**
   * **Issue:** The application required environment variables that were either missing or incorrectly set, which caused startup failures.
   * **Resolution:** I reviewed the code to identify required environment variables and added them to the docker-compose.yml file under the appropriate services.

**2. Resolution Steps**

To resolve the above issues, the following steps were taken:

1. **Build Fixes in Dockerfiles:**
   * Examined each Dockerfile, adjusted file paths, and ensured required dependencies were installed. Rebuilt the images successfully.
2. **Port Configuration in docker-compose.yml:**
   * Edited the docker-compose.yml file to expose port 80 for Nginx and mapped internal application ports to match the host ports.
3. **Network Configuration for Container Communication:**
   * Added a networks section to docker-compose.yml and ensured each service used the same custom bridge network for seamless communication.
4. **Environment Variables Setup:**
   * Added missing environment variables to docker-compose.yml under the respective services, ensuring all containers had access to the correct configurations.

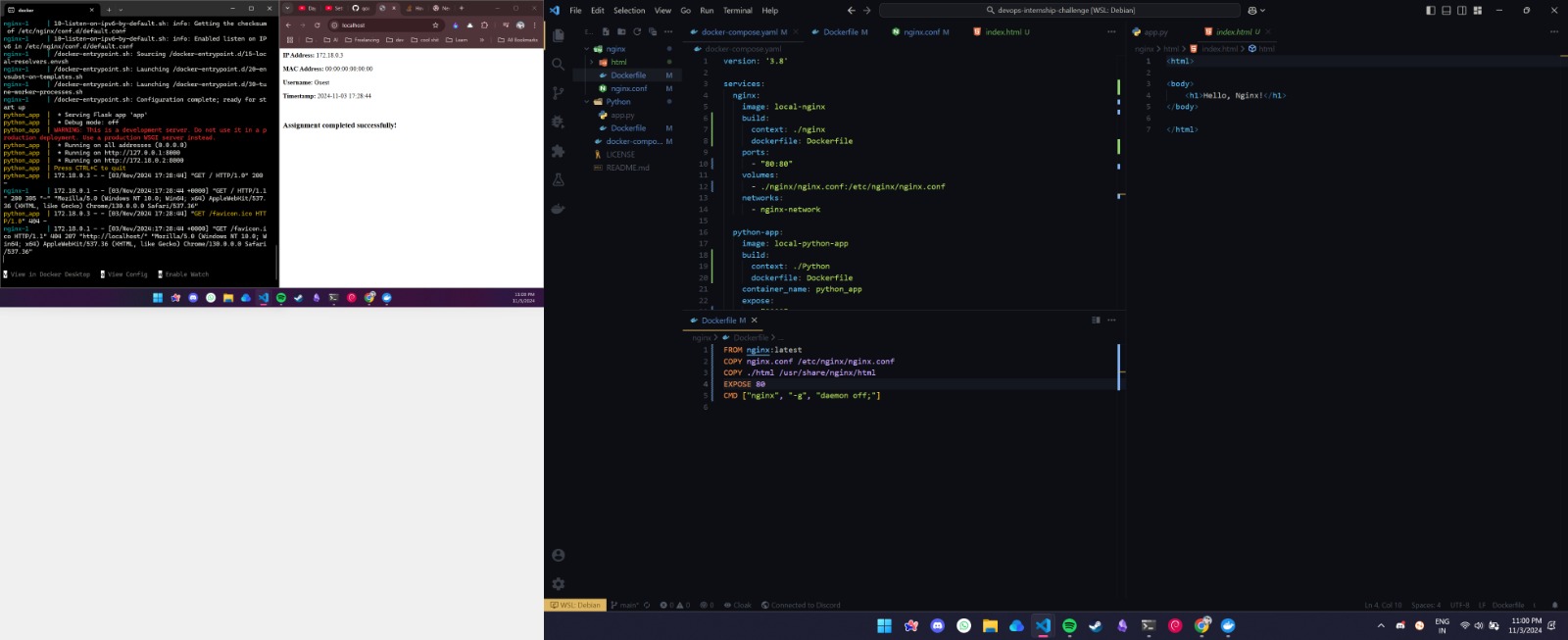
**3. Verification and Testing**

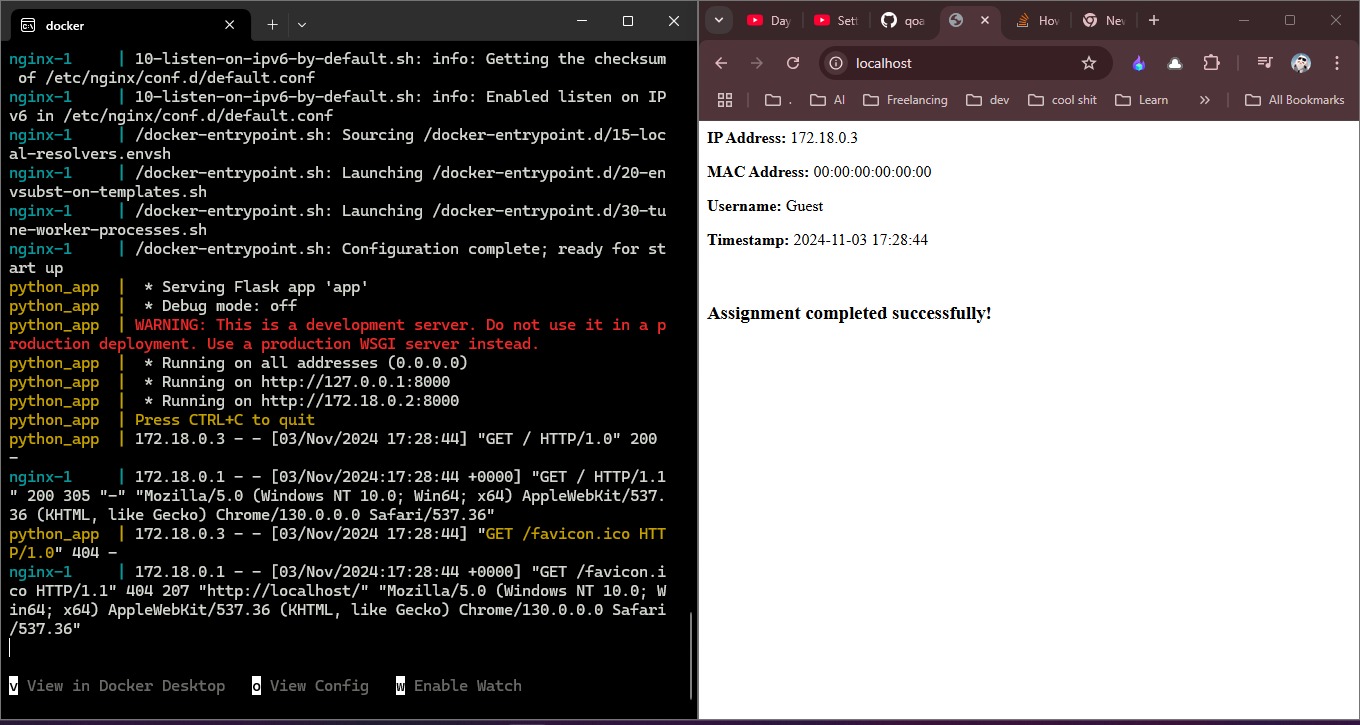
After implementing the fixes, I used the following steps to verify the application:

* **Application Access:** Successfully accessed the application in a web browser at [http://localhost](http://localhost/), confirming the container setup was correct.
* **Nginx Logs Verification:** Checked Nginx logs to confirm that requests were being received and logged, indicating successful connectivity and application operation.

**Screenshots:**

* Screenshot 1: Browser displaying the application at [http://localhost](http://localhost/).
* Screenshot 2: Nginx logs confirming successful requests.





**4. Additional (Bonus) Deployment on Cloud**

For extra credit, I deployed the application on an AWS EC2 instance. I installed Docker and Docker Compose on the instance, cloned the repository, and ran the application. The application was accessible through http://<EC2\_IP>.

**Conclusion**

This assignment helped me gain experience with debugging Dockerized applications and troubleshooting issues related to container configurations and network setups. I successfully identified and resolved the intentional errors and deployed the application both locally and on the cloud for verification.