**Issues Identified in Python**

1. **Inconsistent API Endpoint:**
   * **The initial application used the root endpoint (/) for returning user information, which is not a standard practice for RESTful APIs.**
2. **Non-JSON Response Format:**
   * **The response was returned as an HTML document, making it difficult for frontend applications to parse and utilize the data effectively.**
3. **Potential KeyError in MAC Address Retrieval:**
   * **The implementation of the get\_mac\_address() function did not handle cases where network interfaces might not return the expected format, potentially leading to KeyError exceptions.**
4. **Lack of Clear Data Structure:**
   * **The data returned to the user was displayed directly in HTML format rather than structured as a JSON object, reducing usability for API clients.**
5. **Improper Use of Environment for Username Retrieval:**
   * **The application retrieved the username from request headers without clear handling for the absence of the expected header.**
6. **Errors in Docker File**

**Resolution Steps**

1. **Updated API Endpoint:**
   * **Changed the endpoint to @app.route('/user-info') to clearly define the purpose of the route and align with RESTful API conventions.**
2. **Implemented JSON Response Format:**
   * **Modified the user\_info() function to return a JSON object using jsonify(). This allows frontend applications to easily parse and utilize the returned data.**
3. **Enhanced MAC Address Retrieval Logic:**
   * **Revised the get\_mac\_address() function to use the .get() method when accessing dictionary elements, providing default values to prevent KeyError exceptions and ensure graceful failure.**
4. **Structured Data Return:**
   * **Organized the output data into a structured JSON format within the user\_info() function, enhancing the clarity and usability of the API response.**
5. **Improved Username Handling:**
   * **Clarified the retrieval of the username from request headers, specifying a default value of 'Guest' when the header is not present. This ensures the application behaves predictably even when the header is absent.**

6. **Corrected Directory Name:**

* **Old:** WORKDIR /appp
* **New:** WORKDIR /app
* **Resolution:** Fixed the typo in the working directory.

7 . **Corrected Filename:**

* **Old:** COPY appy.py /app
* **New:** COPY app.py /app
* **Resolution:** Changed the filename to match the actual file.

8. **Corrected Package Name:**

* **Old:** RUN pip install flask netiface
* **New:** RUN pip install flask netifaces
* **Resolution:** Fixed the package name to ensure the correct package is installed.

9. **Fixed EXPOSE Directive:**

* **Old:** EXPOSE "eight thousand"
* **New:** EXPOSE 8000
* **Resolution:** Changed the EXPOSE directive to use the correct numeric format.

10.**Corrected Command Syntax:**

* **Old:** CMD ["pythn", "app.py"]
* **New:** CMD ["python", "app.py"]
* **Resolution:** Corrected the command to ensure proper execution of the application.

**Issues Identified in NGINX**

1. **Nginx Configuration Errors:**
   * **Typographical Errors:** The original nginx.conf file contained multiple syntax errors, such as misspellings in directives (worker\_process, worker\_connection, mime.typess, default\_typ), which could lead to Nginx not starting correctly or not handling requests as intended.
   * **Improper Proxy Configuration:** The proxy\_pass directive was missing a trailing slash, which caused 404 Not Found errors when accessing the application through Nginx.
2. **Application 404 Errors:**
   * The Flask application returned 404 errors for requests made to the root endpoint. This indicated that the application was either not correctly routed by Nginx or that the endpoint did not exist as expected.
3. **Obsolete Docker Compose Version Warning:**
   * A warning was issued indicating that the version attribute in the docker-compose.yaml file is obsolete.
4. **Initial HTML File Missing:**
   * The application lacked a frontend to display user information fetched from the backend, which was necessary for user interaction.

5. **Spelling Error in Base Image Tag:** Used latests instead of latest.

6. **Typo in Configuration File Name:** Used nginix.conf instead of nginx.conf.

7.**HTML Directory Path Error:** Used htmll instead of html.

8. **Incorrect EXPOSE Directive:** Specified the port as a string ("eighty").

9. **Misspelled CMD Command:** The command CMD had a typo (daemon of).

**Resolution Steps**

1. **Fixed Nginx Configuration:**
   * Corrected typographical errors in the nginx.conf file to ensure proper syntax. Changes included:
     + worker\_process auto to worker\_processes auto;
     + worker\_connection 1024; to worker\_connections 1024;
     + include /etc/nginx/mime.typess; to include /etc/nginx/mime.types;
     + default\_typ application/octet-stream; to default\_type application/octet-stream;
   * Added a trailing slash in the proxy\_pass directive to ensure correct routing: proxy\_pass http://python\_app:8000/;.
2. **Created HTML File:**
   * Developed an HTML file to serve as the frontend interface. The file uses JavaScript to fetch user data from the Flask backend and dynamically displays the user's IP address, MAC address, username, and timestamp.

3. **Corrected Base Image Tag:** Changed to latest.

**4. Fixed Configuration File Name:** Corrected to nginx.conf.

1. **Corrected HTML Directory Path:** Fixed to html.
2. **Updated EXPOSE Syntax:** Changed to 80 without quotes.
3. **Corrected CMD Command:** Fixed to daemon off; to ensure NGINX runs properly.

**Issues Identified in the Old Docker Compose File**

1. **Port Configuration:**
   * The port for the nginx service was incorrectly specified as "eighty:80", which is not a valid format.
   * The python-app service used expose instead of ports, and the value was incorrectly stated as "eight thousand".
2. **Volume Mapping:**
   * The volume mapping for the Nginx configuration file had a typo in the destination path (/etc/nginx/nginx.confi should be /etc/nginx/nginx.conf).
3. **Network Driver Typo:**
   * The network driver was incorrectly typed as bridg instead of bridge.
4. **Complex Options in Network Configuration:**
   * The options section included an invalid key compelex\_option, which does not exist in Docker Compose specifications.
5. **Build Context Path:**
   * The python-app service referenced an image instead of building from a Dockerfile. The context path was also incorrectly provided.

**Resolution Steps in the New Docker Compose File**

1. **Corrected Port Configuration:**
   * Changed ports mapping for the nginx service to "80:80".
   * Used ports in the python-app service to specify "8000:8000" correctly.
2. **Fixed Volume Mapping:**
   * Corrected the volume mapping for the Nginx configuration file to ensure it maps to the correct destination path: /etc/nginx/nginx.conf.
3. **Corrected Network Driver:**
   * Fixed the network driver name from bridg to bridge.
4. **Removed Invalid Options:**
   * Removed the options section under nginx-network, simplifying the network configuration.
5. **Build Configuration for Services:**
   * Updated both services to use the build directive, specifying the correct context for the nginx service, and added the build context for the python-app to ensure it correctly builds from its respective Dockerfile.