HTTP APIs for Short URL System with Redirection

Objective:

Design and implement two RESTful HTTP APIs for creating and retrieving short URLs. The “short_url” should be a link that redirects users to the original URL. The solution should accept a JSON payload as input and return a JSON payload as output, with appropriate error handling and input validation. The implementation should be in Python and must follow RESTful style principles, including the use of HTTP methods (GET, POST), resource naming, and HTTP status codes. The solution must also be packaged in a Docker container, pushed to Docker Hub, and hosted in a GitHub repository. Additionally, please provide an API document and a user guide on how to run the container with Docker.

API 1: Create Short URL

Inputs:

A JSON payload containing the following field:

- "original_url": A string representing the URL to be shortened. The URL should be a valid format (validated by the API), and its length should not exceed 2048 characters.

Output:

Returns a JSON payload containing the following fields:

- "short_url": A string representing the generated short URL. This URL should be accessible and should redirect users to the “original_url” when accessed.
- "expiration_date": A timestamp indicating when the short URL will expire, defaulting to 30 days from creation.
- "success": A boolean field indicating whether the URL was successfully shortened.
- "reason": A string field indicating the reason for a failed shortening attempt, such as "Invalid URL" or "URL too long".

API 2: Redirect Using Short URL

Description:

This API allows users to access the “short_url”, which will automatically redirect them to the corresponding “original_url”.

Behavior:

- When users navigate to the “short_url” in their browser, the browser should automatically redirect them to the “original_url”.
- If the “short_url” is expired or invalid, return an appropriate error message or HTTP status code.

Solution Requirements:

1. Use **Python** to implement the solution.
2. Any Python framework (such as Flask, Django, or FastAPI) is acceptable for this implementation.
3. Design and implement the APIs following RESTful principles, utilizing appropriate HTTP methods (GET, POST), status codes, and resource naming conventions.
4. Validate input data such as URL format, length, and presence of required fields.
5. Implement robust error handling to ensure clear, actionable error messages and the use of appropriate status codes.
6. Ensure the “short_url” redirects to the original URL when accessed.
7. Use a suitable data storage solution to persist short URL mappings, including information about expiration dates.
8. Implement rate limiting to prevent abuse of the URL shortening service.
9. Each short URL should have an expiration period (default of 30 days), after which it should no longer be accessible. Ensure this policy is enforced.
10. Package the solution in a Docker container and push it to Docker Hub.
11. Host the solution in a GitHub repository with the source code.
12. Provide a comprehensive API document detailing how to use the APIs, including sample request and response payloads, HTTP methods, status codes, and error messages.
13. Provide a detailed user guide on how to run the container with Docker, including necessary commands and configurations. It is essential to provide clear and detailed instructions to ensure the container can be successfully run using Docker.

Please note that the assessment will not be considered complete until the container is successfully running, so failure to provide a detailed user guide may result in a failed assessment.