Documentation: FastAPI Web Application for Model Predictions

1. Introduction

This document outlines the functionality of a FastAPI web application designed to simulate machine learning model predictions. The application provides endpoints for both synchronous and asynchronous prediction requests, as well as an endpoint to retrieve asynchronous prediction results.

2. Application Overview

The application is built using the FastAPI framework and simulates model predictions using the `mock_model_predict` function. This function takes a string as input, simulates a processing delay, and returns a dictionary containing the input and a randomized result.

3. Endpoints

3.1 /predict (POST)

- **Description:** This endpoint handles both synchronous and asynchronous prediction requests.
- **Input:** A JSON object with a single key, "input," containing the input string for the mock model.
- **Headers:**
    - `Content-Type: application/json` is required.
    - `Async-Mode: true` (optional) to enable asynchronous processing.
    -
- **Output:**
    - **Synchronous (200 OK):** A JSON object containing the "input" string and the "result" from the `mock_model_predict` function.
    -
    - **Asynchronous (202 Accepted):** A JSON object with a "message" indicating that the request is being processed asynchronously and a "prediction_id" to retrieve the result later.
    -
- **Error Handling:**
    - Returns a 400 Bad Request if the "input" is missing in the request body.
- **Example Input (Synchronous):**

```
JSON
{"input": "This is a synchronous test input"}
```

- **Expected Output (Synchronous):**

```
JSON
{"input": "This is a synchronous test input", "result": "12345"}  # "12345" is a random example
```

- **Example Input (Asynchronous):**

```
JSON
{"input": "This is an asynchronous test input"}
```

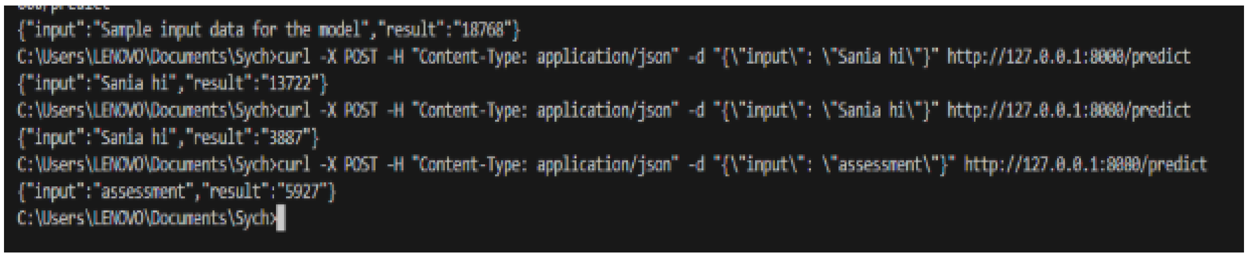
- **Headers (Asynchronous):**

```
Async-Mode: true
Content-Type: application/json
```

- **Expected Output (Asynchronous):**

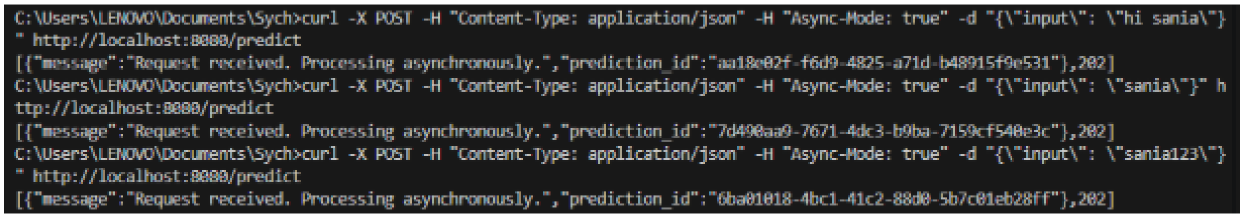
```
JSON
{"message": "Request received. Processing asynchronously.", "prediction_id":
"some_unique_id"}  # "some_unique_id" will be a UUID
```

- **Synchronous Screenshot**



```
{"input":"Sample input data for the model","result":"18768"}
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -d "{\"input\": \"Sania hi\"}" http://127.0.0.1:8080/predict
{"input":"Sania hi","result":"13722"}
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -d "{\"input\": \"Sania hi\"}" http://127.0.0.1:8080/predict
{"input":"Sania hi","result":"3887"}
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -d "{\"input\": \"assessment\"}" http://127.0.0.1:8080/predict
{"input":"assessment","result":"5927"}
C:\Users\LENOVO\Documents\Sych>
```

- **Asynchronous Screenshot**



```
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -H "Async-Mode: true" -d "{\"input\": \"hi sania\"}
" http://localhost:8080/predict
[{"message":"Request received. Processing asynchronously.","prediction_id":"aa18e02f-f6d9-4825-a71d-b48915f9e531"},202]
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -H "Async-Mode: true" -d "{\"input\": \"sania\"}" h
ttp://localhost:8080/predict
[{"message":"Request received. Processing asynchronously.","prediction_id":"7d490aa9-7671-4dc3-b9ba-7159cf540e3c"},202]
C:\Users\LENOVO\Documents\Sych>curl -X POST -H "Content-Type: application/json" -H "Async-Mode: true" -d "{\"input\": \"sania123\"}
" http://localhost:8080/predict
[{"message":"Request received. Processing asynchronously.","prediction_id":"6ba01018-4bc1-41c2-88d0-5b7c01eb28ff"},202]
```

3.2 /predict/{prediction_id} (GET)

- **Description:** This endpoint retrieves the result of an asynchronous prediction.
- **Input:** The `prediction_id` from a previous asynchronous request is passed as part of the URL.

- **Output:**

    - **200 OK:** A JSON object containing the "prediction_id" and the "output" (which includes the original "input" and the "result").
    -
- **Error Handling:**

    - Returns a 404 Not Found if the `prediction_id` is not found.
    -
    - Returns a 400 Bad Request if the prediction is still being processed.
    -
- **Example Input:**

```
/predict/abc123  #  Where "abc123" is a previously obtained prediction_id
```

- **Expected Output (Success):**

```json
{"prediction_id": "abc123", "output": {"input": "This is an asynchronous test input", "result":
"20000"}}
```

- **Expected Output (Still Processing):**

```json
{"error": "Prediction is still being processed."}
```

- **Expected Output (ID Not Found):**

```json
{"error": "Prediction ID not found."}
```

4. Code Explanation

- The application uses FastAPI to define the API endpoints.
- The `mock_model_predict` function simulates a machine learning model prediction.

- The `/predict` endpoint handles both synchronous and asynchronous requests. Asynchronous requests are processed in a separate thread.
- `prediction_results` dictionary stores the results of asynchronous predictions, using the `prediction_id` as the key.
- The `/predict/{prediction_id}` endpoint retrieves the prediction result from the `prediction_results` dictionary.
- Error handling is implemented using FastAPI's `HTTPException` to return appropriate status codes and error messages.
-

5. Dockerization

- A `Dockerfile` is included to build a Docker image of the application, ensuring consistent deployment.

This documentation provides a clear overview of the application's functionality, input/output formats, and error handling, as requested in the assessment.