Email Classification System with PII Masking

# Introduction

The goal of this project is to develop an email classification system tailored for a company's support team. Incoming emails must be categorized into predefined categories such as "Change", "Request", "Complaint", etc. To comply with data privacy requirements, all personally identifiable information (PII) should be masked before classification and demasked afterward in the response.

# Approach

## 1. PII Masking Strategy

A modular masking function was built using regular expressions (regex) to detect and replace the following PII types:

- Emails (e.g., abc@domain.com)

- Phone numbers (Indian and general formats)

- Credit/Debit card numbers (13–16 digit patterns)

- CVV numbers (3 digits following keyword)

- Expiry dates (MMYY or MM/YYYY)

- Aadhar numbers (12-digit format with optional spaces/hyphens)

- Full names (identified by phrases like “my name is...”)

- Dates of birth (numerical patterns with or without separators)

## Challenges & Solutions

Several challenges were encountered during development:

1. Overlapping Entities: Some values like '0827' could be interpreted as credit card expiry or Aadhaar depending on context.

- Solution: Improved regex and masked in a priority order to reduce ambiguity.

2. Incomplete or Poorly Formatted Data: Emails often contain partial information or unusual formatting.

- Solution: Regex was made more flexible, covering optional separators, varied lengths, and different cases.

3. Name Detection: Full name detection was tricky without LLMs.

- Solution: Structured regex tied to identifying phrases (e.g., “my name is”).

4. Reversible Demasking: Ensuring the mapping stayed consistent after multiple substitutions was important for traceability.

- Solution: We used first-match masking and a dictionary of original values per tag.

5. Aadhar Detection Error: In some cases, parts of Aadhar were misclassified as other tokens (e.g., DOB or expiry).

- Solution: The issue was solved by ordering regex patterns carefully so Aadhar is matched before DOB or expiry patterns.

6. CVV Not Masked: Initial regex missed CVV if not followed by spaces or “is”.

- Solution: Regex was improved to be case-insensitive and allow flexible spacing.

# Model Selection and Training

## Data Preparation

- Lowercasing: All emails were lowercased.

- Cleaning: Removed special characters, headers like `Subject:`, and numbers.

- Stopword Removal: English stopwords were eliminated using NLTK.

- Vectorization: TF-IDF vectorizer was used to convert text into numeric features.

## Model: Logistic Regression

Logistic Regression was selected for its simplicity and strong performance on text classification with TF-IDF features.

- Accuracy: 75.58% on validation set.

- Training time: Fast and efficient on small-medium datasets.

- Model was serialized using `joblib`.

# Pipeline Overview

1. Input Email → Passed to FastAPI endpoint.

2. Masking Module → Extracts and replaces PII.

3. Classification Module → Vectorizes the masked text and classifies using Logistic Regression.

4. Demasking Module → Restores original PII in final output.

# API Deployment & Usage

Deployed API (Hugging Face Spaces):

<https://harshr16-email-classifier-api.hf.space/classify>

Source Code (GitHub Repository):

## <https://github.com/Harshr16/Email_classifier>

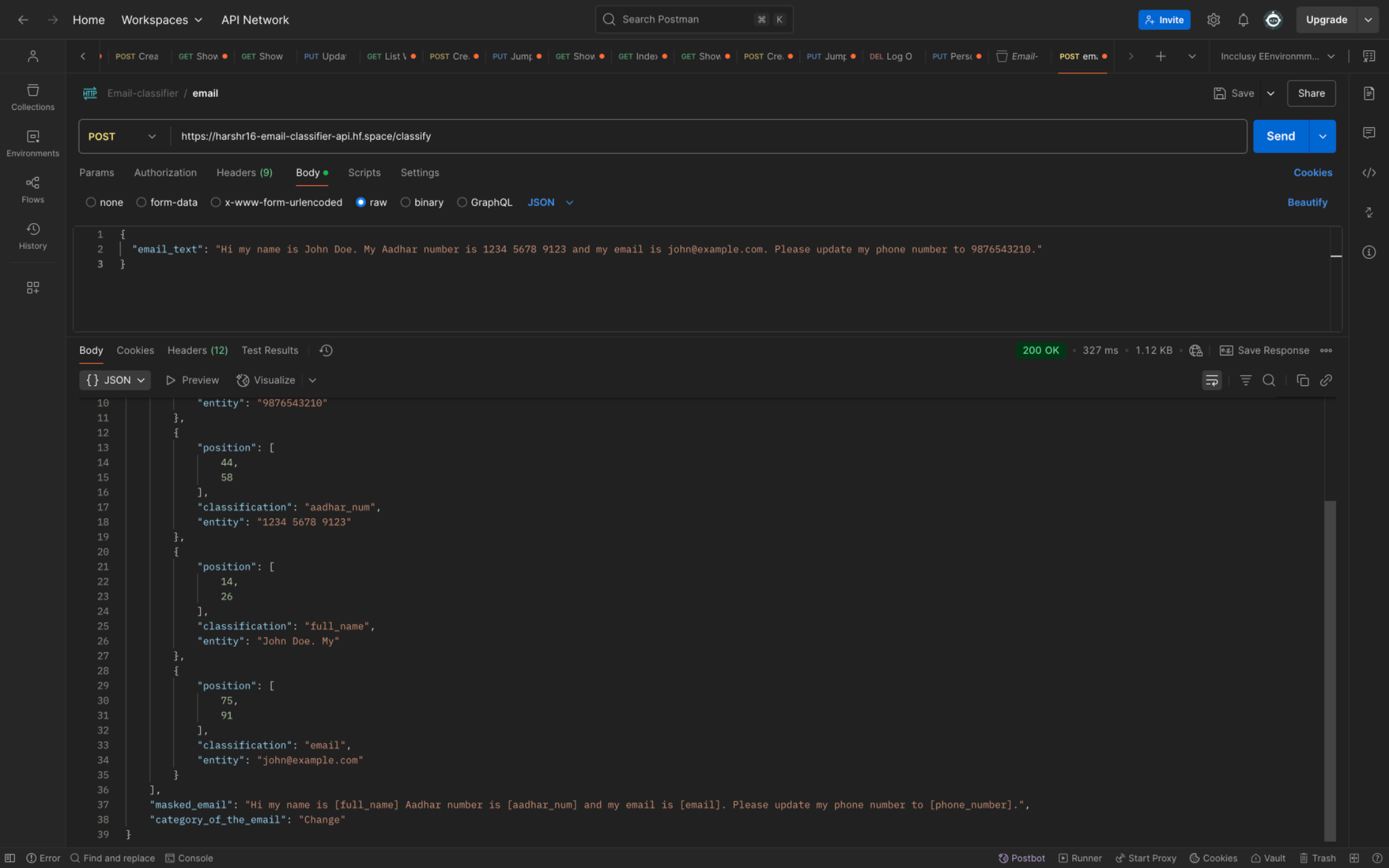
## 

## How to Use the API via POST Request (e.g., Postman):

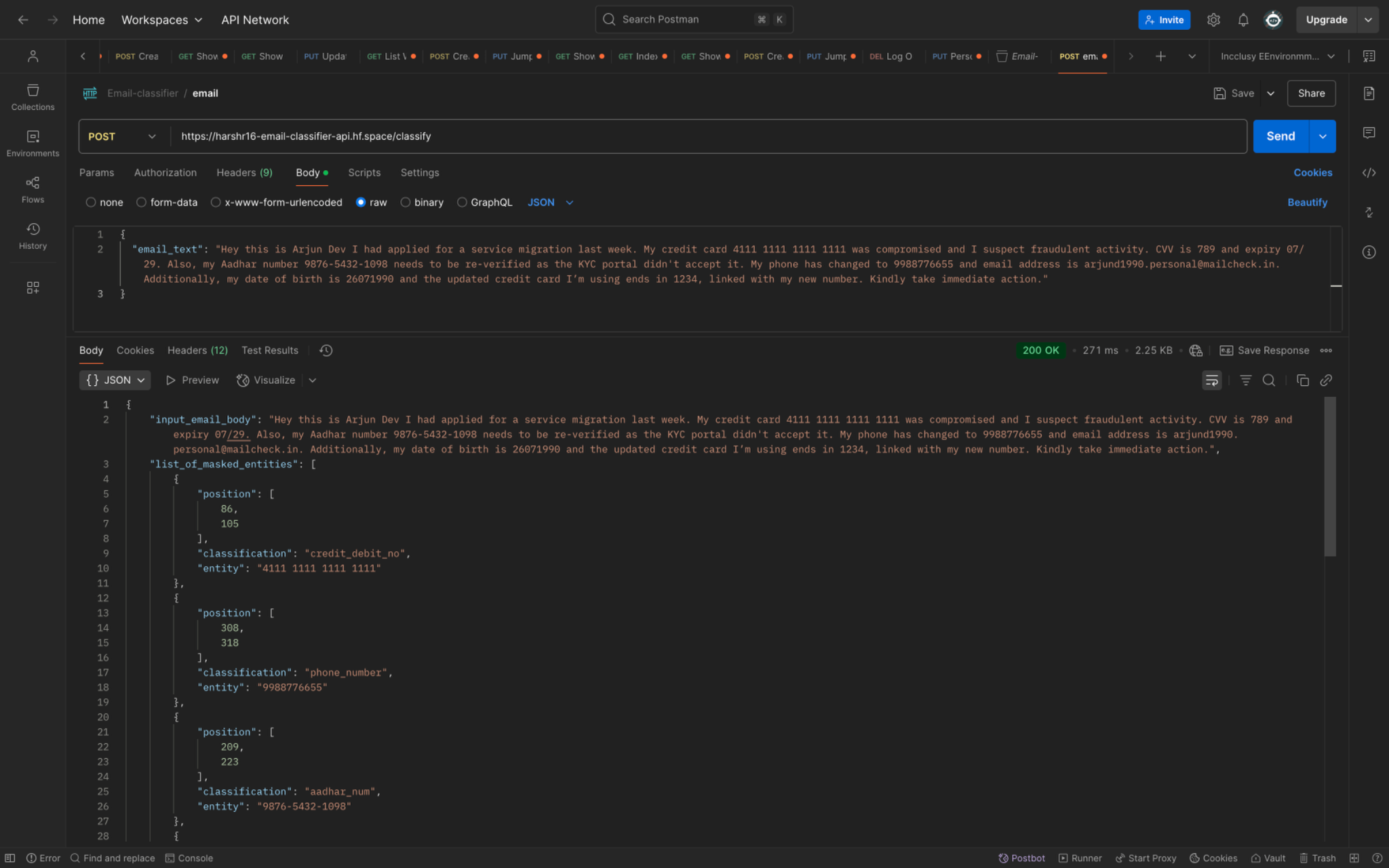
To use the API, send a POST request to:  
https://harshr16-email-classifier-api.hf.space/classify  
  
Request Body (JSON):  
{  
 "email\_text": "Hi my name is Harsh Rudani. My phone is 9876543210 and CVV is 123."  
}  
  
Response (JSON):  
{  
 "input\_email\_body": "...",  
 "list\_of\_masked\_entities": [...],  
 "masked\_email": "...",  
 "category\_of\_the\_email": "..."  
}

## Postman Screenshots:

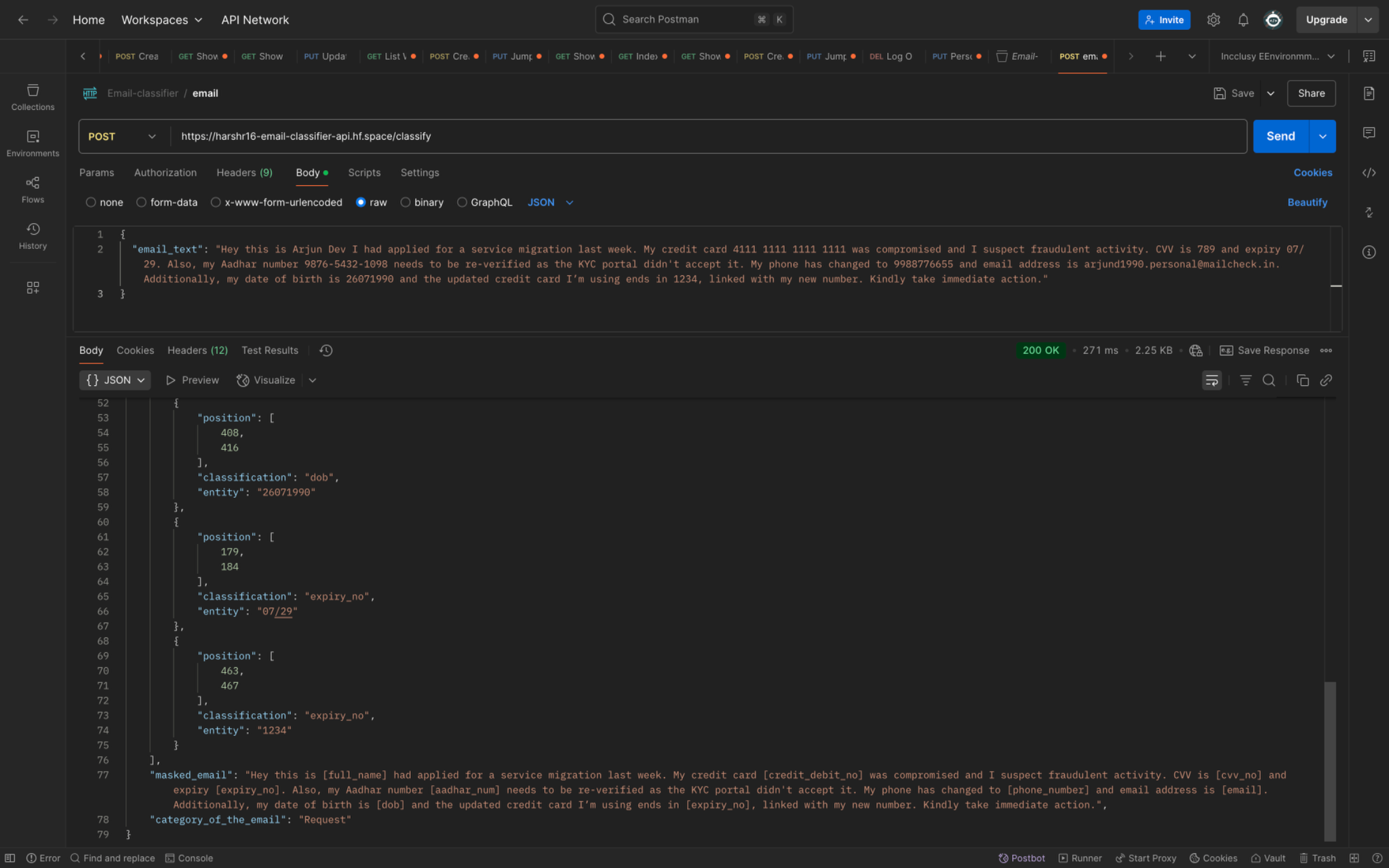
### POST Request and Response Screenshot



### POST Request Screenshot



### Masked Result Screenshot



# Conclusion

# This solution effectively balances privacy protection and automated email classification. Through robust PII masking using regular expressions and a FastAPI-powered deployment, the system ensures sensitive data is securely handled before classification. The interactive API documentation makes it developer-friendly and easy to test.

The modular architecture of the PII detection logic allows for future extensibility i.e whether that’s supporting international formats for PII (like UK NI numbers or EU phone codes) or integrating advanced NLP-based classification models for greater accuracy and contextual understanding.

This project sets a strong foundation for secure, scalable, and intelligent email handling in real-world applications.

­­­­­­­­­­­