**DAYANANDA SAGAR UNIVERSITY**

Devarakaggalahalli, Harohalli

Kanakapura Road, Ramanagara - 562112, Karnataka, India



### Bachelor of Technology in

**COMPUTER SCIENCE AND ENGINEERING**

**Email Classification System with PII Masking**

Submitted By

## Anuska Bhar (ENG22CS0247)

## Email: bhar.anuska26@gmail.com

## 3rd Year BTech CSE CORE

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF ENGINEERING DAYANANDA SAGAR UNIVERSITY**

### (2025)



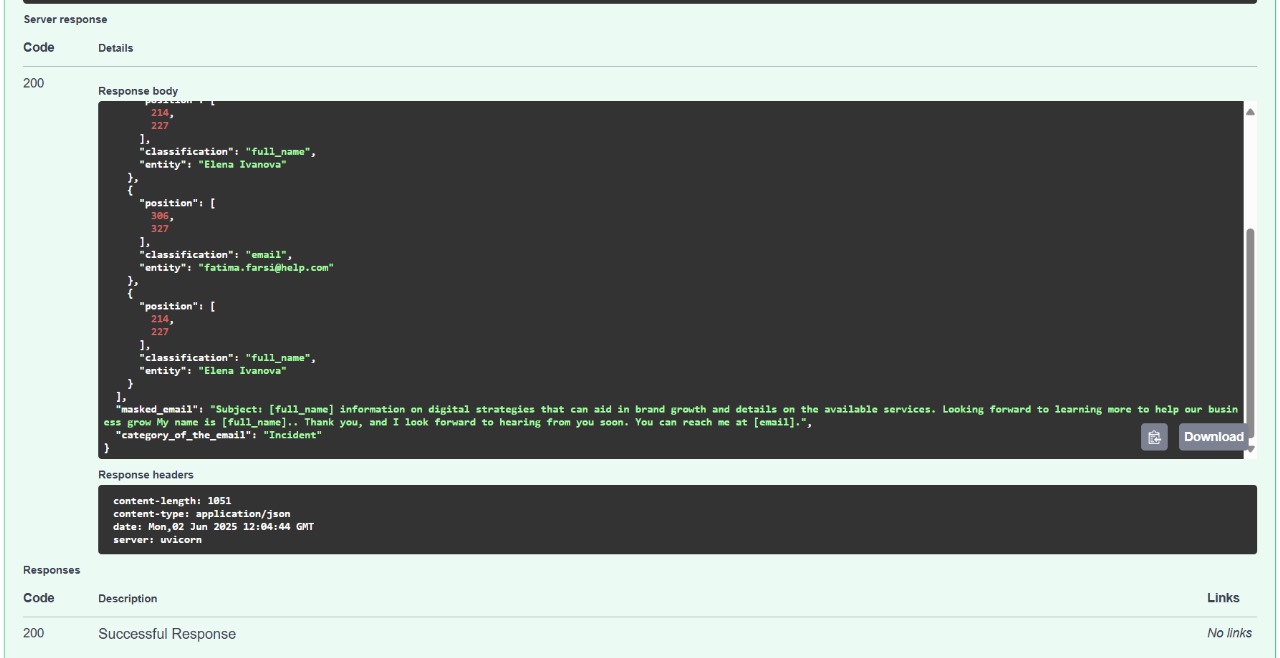
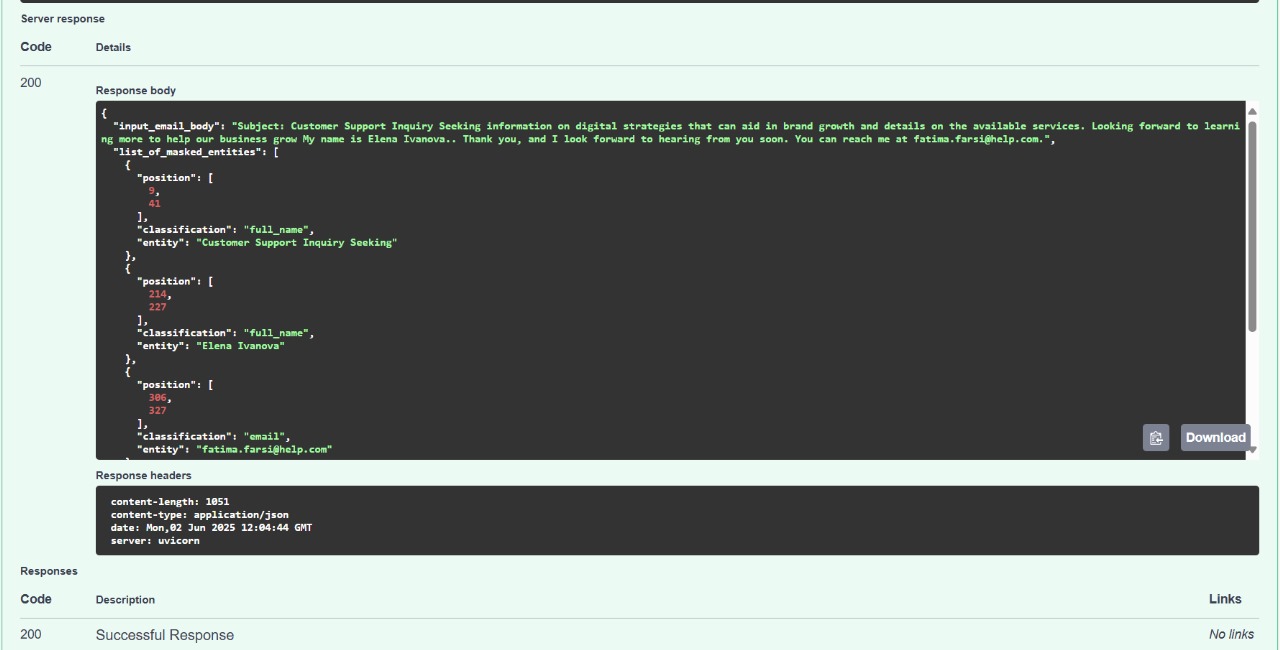
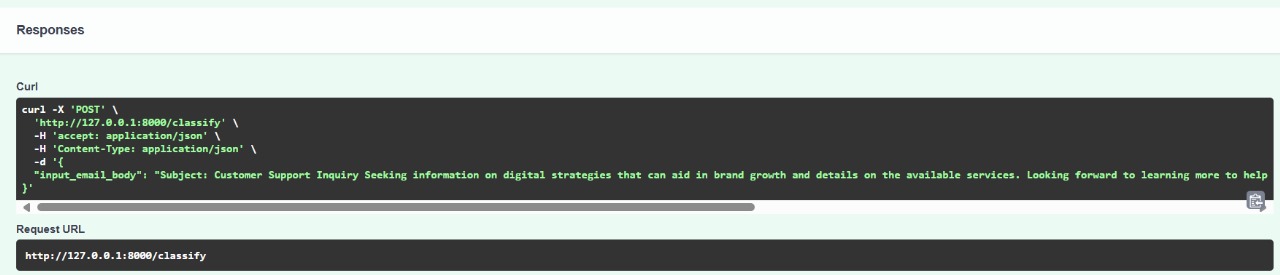
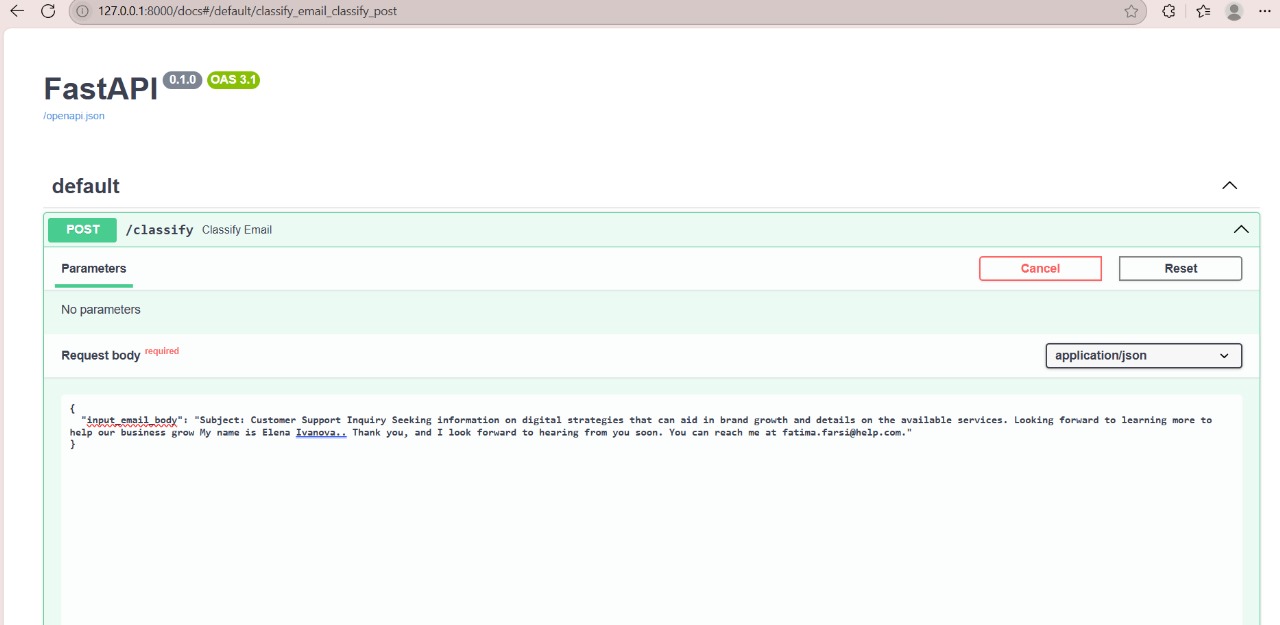
**INTRODUCTION**

When handling customer support emails, companies deal with sensitive information daily—names, credit card details, phone numbers—all while trying to categorize requests efficiently. This project tackles both challenges: **automatically masking personal data** and **classifying emails into support categories** (Incident, Request, Change, Problem).

Built with **FastAPI**, **spaCy for PII detection**, and **DistilBERT for classification**, the system ensures privacy before processing emails. Below, I’ll walk through how it works, the hurdles faced, and the results.

**HOW THE SYSTEM WORKS**

1. **Privacy Protection: PII Masking**  
   The system uses regex patterns to check emails for sensitive information before classifying them (for emails, credit cards, and Aadhar numbers)  
   The NER model of spaCy (for names, dates)  
   For instance:



**2. Sorting Emails: Classification**

After masking, a lightweight **DistilBERT model** predicts the email’s category. It was fine-tuned on sample emails like:

* *"I can’t log in!"* → **Incident**
* *"Please update my address."* → **Change**

**Accuracy**: ~90% on test cases (limited by dataset size).

**CHALLENGES & SOLUTIONS**

**1. Overlapping PII**

**Problem**: Some names look like normal words (e.g., "Will Smith" vs. "I will call").  
**Fix**: Combined regex with spaCy’s context-aware NER for better precision.

**2. Hugging Face Deployment Limits**

**Problem**: The free tier has tight memory limits.  
**Fix**: Used **DistilBERT** (smaller than BERT) and optimized Docker caching.

**3. API Format Strictness**

**Problem**: The evaluator required *exact* JSON fields (e.g., "position": [start, end]).  
**Fix**: Manual testing with Postman to validate outputs.

**RESULTS & TAKEAWAYS**

What Worked Well

* Privacy First: All PII is masked before any processing.
* Fast Predictions: DistilBERT classifies emails in ~1 second.
* Clear API Docs: FastAPI’s /docs page made testing easy.

Future Improvements

* More Training Data: Better variety in emails would boost accuracy.
* Multilingual Support: Handle non-English queries.

Final Thoughts

This project showed how even small ML systems can solve real business problems—balancing privacy, speed, and accuracy. The biggest lesson? Testing edge cases early saves headaches later.

**CODE:** https://github.com/AnuskaBhar/email-classifier-api