Smart Customer Support Chatbot using NLP

1. Introduction

Customer support is a critical part of any service-based company. As ticket volume grows, so do the challenges in maintaining timely and accurate responses. This project uses **Natural Language Processing (NLP)** to build an intelligent chatbot that automates responses, categorizes inquiries, and extracts useful information for efficient issue resolution.

---

2. Problem Statement

A company wants to enhance its customer support by implementing NLP techniques to automate responses and categorize customer inquiries. The aim is to improve **response time**, **efficiency**, and **overall customer satisfaction**.

---

3. Objectives

- Build a smart chatbot powered by NLP.
- Automatically classify customer support messages.
- Extract important details like order ID, product name, etc.
- Provide recommendations and log support cases for future analysis.

---

4. Goals

- Achieve classification accuracy of **85% or higher**.
- Handle major intents like **Refund Request, Technical Issues, Product Inquiry**, etc.
- Reduce average response time by **20%**.
- Maintain structured logs for all ticket conversations.

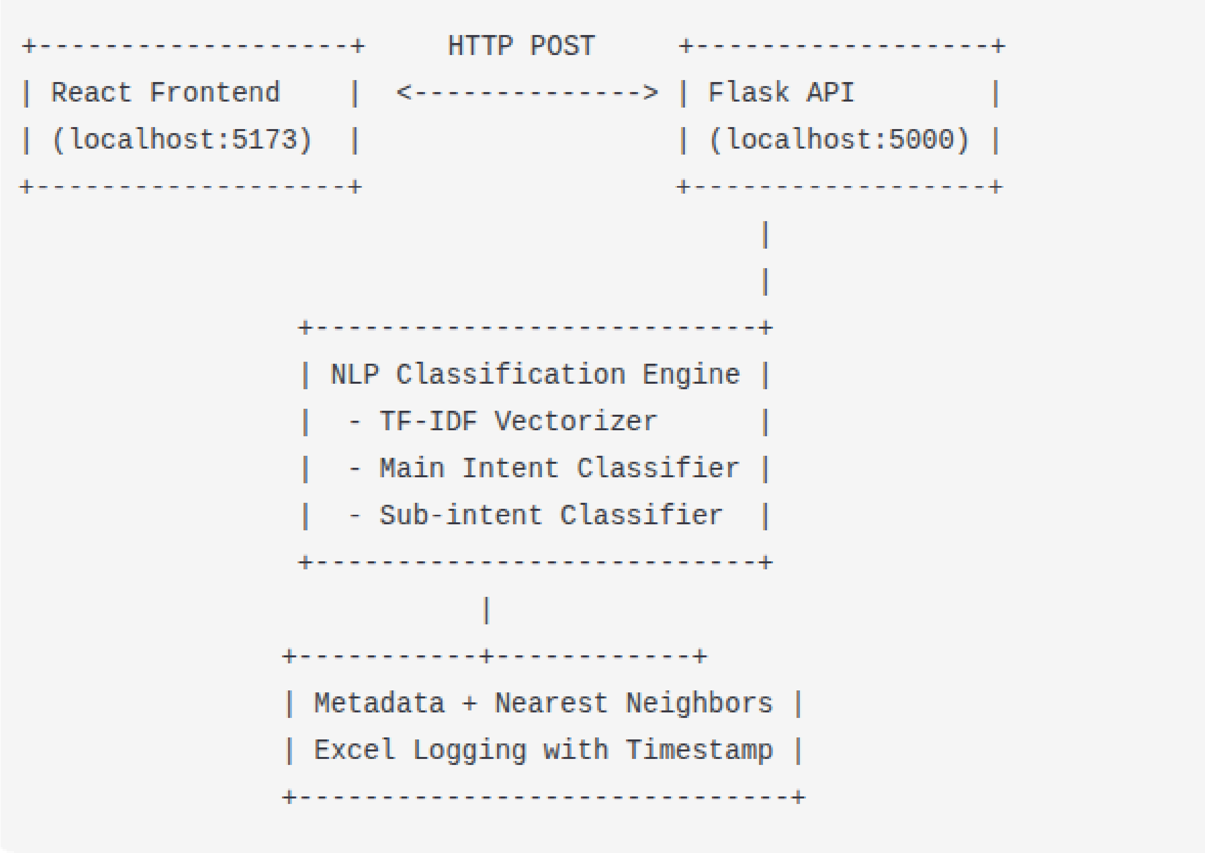
---

5. Scope

- Data collection and cleaning
- Intent classification & sub-intent detection
- Slot filling using rule-based and keyword matching
- Nearest neighbor search for similar past issues
- End-to-end integration of backend (Flask API) and frontend (React)

---

6. System Architecture



```
+--------------------+     HTTP POST     +-------------------+
| React Frontend     |  <--------------> | Flask API         |
| (localhost:5173)   |                   | (localhost:5000)  |
+--------------------+                   +-------------------+
                                                   |
                                                   |
                 +-----------------------------+
                 | NLP Classification Engine |
                 |   - TF-IDF Vectorizer       |
                 |   - Main Intent Classifier |
                 |   - Sub-intent Classifier  |
                 +-----------------------------+
                             |
                +------------+------------+
                | Metadata + Nearest Neighbors |
                | Excel Logging with Timestamp |
                +------------------------------+
```

---

7. Dataset Description

- **Data Source:** Real/synthetic support ticket dataset
- **Fields:**
  - ticket_id
  - text (user message)
  - category

    - timestamp
    - product_name, order_id, etc. (optional metadata)

8. NLP Pipeline & Techniques Used

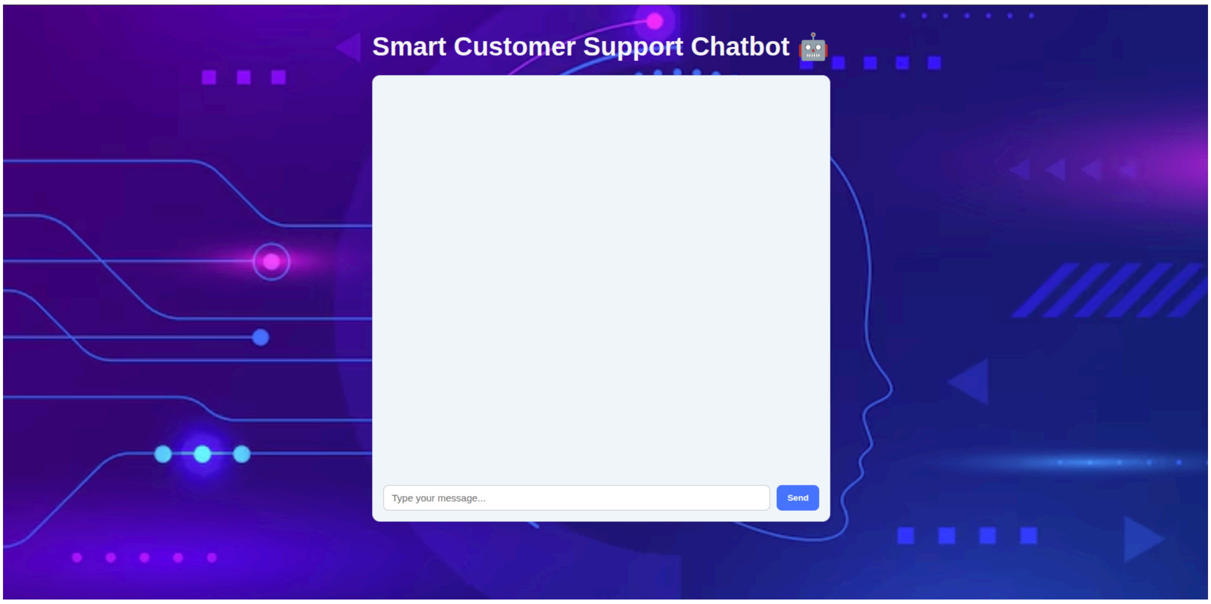
Task	Technique Used
Text Preprocessing	Lowercasing, Stopword removal, Lemmatization
Feature Extraction	TF-IDF for text, One-Hot Encoding for metadata
Classification	Logistic Regression (main), Sub-models (sub-intents)
Slot Filling	Regex + keyword-based extraction
Similar Retrieval	NearestNeighbors from scikit-learn

9. Chatbot Flow (Multi-turn)

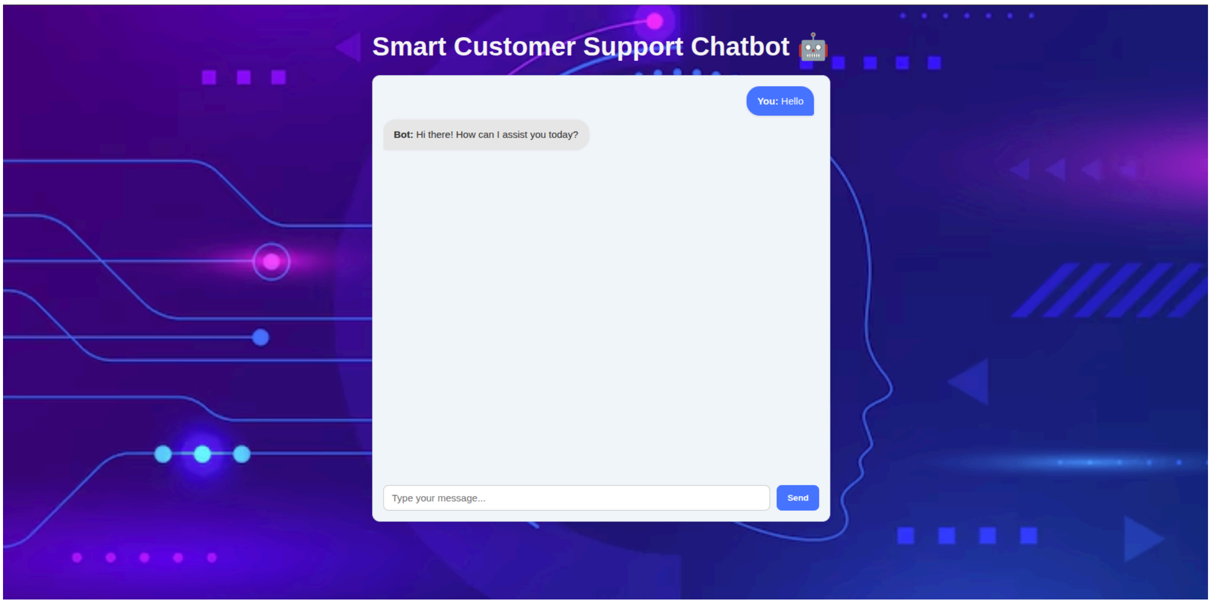
1. User types query (e.g., *"I want a refund on my phone"*).
2. Bot classifies intent: **Refund Request**.
3. Bot checks and asks for required slots: order_id, product_name, reason.
4. Bot extracts info using rules or prompts user.
5. Bot confirms:
   "Your request for 'Refund request' has been submitted with info: {...}"
6. Logs the ticket to Excel (support_logs.xlsx).

10. User Interface Snapshots

- **Initial View**



- **After Sending Message**



You: Hello

Bot: Hi there! How can I assist you today?

11. Deployment Instructions

Backend Setup

```
pip install -r requirements.txt
python run.py
```

Frontend Setup

```
npm install
npm run dev
```

12. Files Included in Submission

- frontend.zip – Frontend source code (React)
- backend.zip – Backend source (Flask, models, SupportChatbot class)
- support_logs.xlsx – Chat log file (generated at runtime)
- use_manual.txt – Steps to run backend & frontend
- report.pdf – This submission report

13. Conclusion

This chatbot offers a scalable solution to streamline customer support using machine learning. With its modular backend, explainable structure, and clear UI, it can be adapted for any customer-driven service portal. Future enhancements can include:

- Integrating LLMs for richer responses
- Supporting more languages
- Adding feedback loops to learn from escalations

Note: Product Name Limitation

Currently, the chatbot is configured to recognize a limited set of product names for slot extraction:
['tv', 'laptop', 'fan', 'shirt', 'book', 'phone', 'headphones', 'router']

This list was chosen for demonstration purposes. To make the chatbot more robust and adaptable to your company's product catalog, you can easily **extend or replace** this list by:

- Updating the `product_keywords` list in the `extract_slots()` function inside `SupportChatbot` class.
- Dynamically loading product names from a database or CSV file for scalability.

This flexibility ensures that the chatbot can be customized for **any domain or industry**.