# Project Documentation

## Comment Categorization & Reply Assistant Tool

1. Project Overview  
  
This tool is developed to automatically analyze and categorize user comments based on their tone, sentiment, and intent. It helps brands and teams respond more effectively by identifying the nature of comments such as praise, support, criticism, spam, or questions.  
  
2. Objective  
  
The goal is to build a lightweight, modular, and intelligent assistant that:  
- Classifies comments into meaningful categories.  
- Suggests appropriate replies.  
- Includes fallback handling using BERT for ambiguous or low-confidence inputs.  
- Provides a user-friendly interface.  
  
3. Categories Covered  
  
1. Praise  
2. Support  
3. Constructive Criticism  
4. Hate/Abuse  
5. Threat  
6. Emotional  
7. Spam  
8. Question/Suggestion  
  
4. Final Folder Structure  
  
categorize\_comments/  
├── app.py # Main Streamlit application  
├── preprocess.py # Text cleaning and lemmatization  
├── reply\_templates.py # Predefined response generation  
├── bert\_classifier.py # Fallback classifier using BERT  
├── requirements.txt # Required Python packages  
├── models/ # Saved model and vectorizer  
│ ├── model.pkl  
│ └── vectorizer.pkl  
├── data/ # Input sample dataset  
│ └── sample\_comments.csv

5. Technologies Used  
  
- Language: Python 3.9+  
- Libraries: scikit-learn, pandas, nltk, transformers, torch, streamlit, plotly  
- Model: Logistic Regression with TF-IDF + BERT fallback (DistilBERT)  
- UI Framework: Streamlit  
  
6. Key Features  
  
- Efficient hybrid classification  
- Fallback to BERT for ambiguous cases  
- Auto-reply suggestion per category  
- Interactive UI (web-based)  
- Visualizations of comment category distribution  
  
7. How to Run the Project  
  
 Using Python (Streamlit)  
  
1. Install dependencies:  
 pip install -r requirements.txt  
  
2. Download NLTK resources (one-time setup):  
 import nltk  
 nltk.download('stopwords')  
 nltk.download('wordnet')  
  
3. Run the application:  
 streamlit run app.py  
  
9. Conclusion  
  
This project successfully delivers a clean, professional comment analysis tool with automation, flexibility, and extensibility in mind. With its user-friendly interface and intelligent architecture, it can be readily adopted into real-world content moderation workflows.  
  
Prepared by: Don Joshy