**Project Assignment: AI-Powered Spam Filter for A2P SMS**

**Objective**  
Design and implement a lightweight AI-based spam filtering system tailored for A2P (Application-to-Person) SMS messages. The system should intelligently classify SMS content as spam, transactional, or promotional, while also incorporating a whitelist mechanism to allow safe exceptions (e.g., trusted domains like trip.com or specific OTP templates).

**Why This Matters**  
In the telecom messaging industry, blocking suspicious content (phishing links, scam messages) is critical. However, existing systems based solely on keyword or domain blocklists often produce false positives. For example, blocking all “.com” domains may mistakenly block messages containing legitimate domains like trip.com.

This project aims to create a smarter filtering system that applies both rule-based filtering and machine learning classification to avoid such issues.

**Requirements**

### 1. Spam Classification & Dataset Preparation

* You will be provided with a dataset of raw A2P SMS messages from our company.
* Your first task is to preprocess this dataset by:
  + Cleaning the messages (remove duplicates, normalize casing, strip noise like special characters, excessive whitespace, etc.).
  + Performing manual or semi-automated data labeling to classify each message into one of three categories:
    1. Transactional — e.g., OTPs, purchase confirmations, alerts
    2. Promotional — e.g., sales, offers, product promotions
    3. Spam — e.g., phishing links, scam messages, policy violations
* Prepare a final labeled dataset in CSV format with the following columns: | message | category | cleaned\_message |
* Use this labeled dataset to train a lightweight message classifier using open-source tools only (e.g., scikit-learn, fastText, or similar).
* The trained model should support real-time inference (under 100ms per message on CPU).
* Include safeguards against false positives by incorporating contextual awareness (e.g., trust score of domains, message structure, OTP templates).
* Evaluate your model using standard classification metrics:
  + Accuracy
  + Precision, Recall, F1-Score
  + Confusion Matrix (optional visualization)
  + AUC-ROC (if applicable for your chosen model)

### 2. Whitelisting Layer

* Implement a whitelist mechanism to bypass the spam filter for trusted:
  + Domains (e.g., trip.com, icicibank.com)
  + Phrases (e.g., “Your OTP is…”, “Thank you for shopping with…”)
  + Sender IDs (optional but encouraged)
* Ensure this runs before or alongside the AI model.

### 3. REST API

* Expose the filtering logic as an API (using FastAPI or Flask).
* Endpoint: POST /check\_sms
* Input: { "message": "..." }
* Output: { "verdict": "allowed" | "blocked", "reason": "whitelisted" | "ai" | "rule\_match" }

### 4. Logging & Config

* Log every processed message and its classification result.
* Include basic config files (YAML/JSON) to manage whitelist entries and model thresholds.

### 5. Deployment

* Containerize the project using Docker.
* The image should expose the API service and be easy to run with docker run.

### 6. Documentation

* Provide a short README with:
  + Project overview
  + Instructions to train and run the model
  + How to add whitelist entries
  + Example API usage

**Evaluation Criteria**

- Functionality: Does the system accurately classify messages and support whitelisting?

- Simplicity: Is the system easy to understand, run, and maintain?

- Code Quality: Clean, readable, modular code (with comments and reasonable structure).

- Performance: Should handle 1000+ messages per second (on a single containerized instance).

- Model Evaluation: Clear demonstration of model performance using accuracy, precision, recall, F1-score, and (optionally) confusion matrix or AUC-ROC.

- Innovation: Bonus points for intelligent approaches to contextual filtering or confidence thresholds.

**Bonus (Optional)**

- Support multiple message types (e.g., OTP, promo) with different filtering rules

- Confidence scoring from the ML model - Auto-updating whitelist based on user feedback or delivery outcomes

**Submission Instructions**

- Create a GitHub repository and share the link with your submission.

- The repository should include all source code, training data (or download instructions), and documentation.

- Also, create a Loom video of explaining your project walkthrough. Share us the loom link in the readme file.

**Deadline**  
Please submit your completed project within 7 days of receiving this assignment.

**Questions?**  
If you have any questions or face blockers, feel free to reach out to the hiring manager. We’re looking to evaluate your problem-solving ability, not just the final output.