AI Developer Technical Assessment

Objective

Greeklish usually refers to writing in greek but using latin characters, like "ti kaneis" instead of "τι κάνεις", "ekeino to tragoudi einai teleio" instead of "εκείνο το τραγούδι είναι τέλειο".

Your task is to develop a text classifier capable of accurately determining whether a sentence written using the Latin alphabet is "**Greeklish**" (Greek transliterated into Latin characters) or **standard English**.

Task Requirements

Data Collection

- Scrape at least 3 distinct online sources containing authentic Greeklish text.
- Scrape standard English sentences from at least 2 different sources.
- Minimum data requirement: 300 unique sentences per class (600 total minimum).

Data Preparation

- Each sentence should be labeled as either “Greeklish” or “English” (do **NOT** manually label the dataset)
- Clean and preprocess the collected data for NLP classification.
- Format data into a structured CSV file clearly showing labeled data.

Model Development

- Train an NLP classifier to distinguish Greeklish text from standard English.
- Evaluate your model's performance, providing accuracy, precision, recall, and F1-score.

Technical Requirements

- You have freedom in choosing NLP methods and classifier algorithms. Clearly justify these choices in your documentation.
- Code must be clearly structured and readable.
- Do NOT use pre-existing datasets; all data must be independently collected.
- Python is mandatory.

Submission Requirements

You must submit your project as a GitHub repository, including:

- Source Code, all Python scripts (for scraping, preprocessing, training, evaluation).
- Your **trained model** file under a directory /model.
- The labeled dataset (Greeklish and English).
- A Documentation PDF clearly explaining:
  - Choice of data sources.
  - Data scraping methods and preprocessing steps.
  - Rationale for model selection, training process, and evaluation.
  - Challenges faced and solutions provided.
- README.md with clear instructions on:
  - Setting up the development environment (dependencies).
  - Running scraping, preprocessing, training, and evaluation scripts.
  - How to test the trained classifier.

Evaluation Criteria

Data Quality	40%
Model Performance (Accuracy, F1-score, Precision, Recall)	25%
Code Quality and Documentation	20%
Creativity and technical choices	15%

Clarifications

No Manual Use of LLMs:

- You should **not copy-paste** data into ChatGPT or similar tools for labeling or extraction. This won't scale in real-world scenarios.
- However, if you choose to use an **LLM via code** (like an API or a local model), that's acceptable **as long as it's fully automated** (i.e., your code loops through rows, sends input, parses output).
- You are also encouraged to explore **non-LLM approaches** for labeling — we want to see your thought process and creativity.

Clarifying 'Sources':

- When asked for 3 Greeklish sources, we mean **distinct platforms** (e.g., Reddit, forums, Twitter), **NOT** different articles from the same site.

No Need to Pay for Anything:

- You are **not expected to pay** for APIs, cloud services, or datasets.
- Everything can be done with **free tools**, open-source libraries, or local models.

We Want You to Succeed:

- This is not a "trap" test. If you show a good approach to scraping, labeling, and training — even with basic models — you'll do well.