**📄 SvaraAI – AI/ML Engineer Internship Assignment**

**Duration: 22-09-2025 to 24-09-2025 (No responses will be entertained after the deadline).**

**Submission: Must include a GitHub repository (or Google Colab link), a short README, and a brief video explaining your task.**

**Languages Allowed: Python only.**

**🚀 Problem Statement**

At SvaraAI, we build ML-powered features for outbound sales. One core task is to classify replies from prospects into categories (e.g., interested, not interested, neutral) so that sales reps know where to focus their efforts.

Your assignment is to design, implement, and deploy a simple reply classification pipeline.

**🧩 Part A – ML/NLP Pipeline (Core Task)**

We provide you with a small dataset of email replies in csv. Each reply is labeled as one of :

* positive (interested in meeting/demo)
* negative (not interested / rejection)
* neutral (non-committal or irrelevant)

Tasks :

1. Load and preprocess the dataset (clean text, handle missing values).
2. Train a baseline model (e.g., Logistic Regression or LightGBM).
3. Fine-tune a small transformer model (e.g., distilbert-base-uncased) using Hugging Face.
4. Evaluate models using accuracy + F1 score.
5. Compare results and explain which model you’d use in production and why.

**🧩 Part B – Deployment Task (API)**

1. Wrap your best model in a FastAPI (or Flask) service with a /predict endpoint.
   1. Input: JSON with a text string, e.g. { "text": "Looking forward to the demo!" }
   2. Output: JSON with prediction and probability, e.g. { "label": "positive", "confidence": 0.87 }
2. Add a small README with instructions to run your API locally.
   1. Bonus: Add a Dockerfile or requirements.txt.

**🧩 Part C – Short Answer (Reasoning)**

Answer briefly (2–3 sentences each) in a separate markdown file ([answers.md](http://answers.md)):

* If you only had 200 labeled replies, how would you improve the model without collecting thousands more?
* How would you ensure your reply classifier doesn’t produce biased or unsafe outputs in production?
* Suppose you want to generate personalized cold email openers using an LLM. What prompt design strategies would you use to keep outputs relevant and non-generic?

**📦 Deliverables**

* notebook.ipynb (or Python scripts) with code for Part A.
* app.py (FastAPI/Flask code) for Part B.
* answers.md with reasoning answers.
* README.md with setup instructions.

**✅ Evaluation Criteria**

* ML Pipeline (40%), Deployment (25%), Reasoning (20%), Code Style (15%)