AI Engineer
Take-Home Assignment

Overview

This assignment evaluates your ability to build production-grade AI systems from first principles. You will choose **ONE** challenge from the options below and complete it within 2 working days.

We are not looking for state-of-the-art results. We want to see how you think about building AI systems, make engineering trade-offs, and communicate your decisions.

Time and Constraints

- **Time Limit:** 2 working days
- **Hardware:** Must train on a standard laptop (no cloud GPUs required)
- **Allowed:** PyTorch/JAX, NumPy, standard tokenizer libraries, pandas, matplotlib
- **Not Allowed:** Pre-trained model weights, HuggingFace model classes (Trainer, AutoModel, etc.)

Data Sourcing

You must create your own training dataset. This is part of the evaluation.

- **Allowed:** Public datasets (Kaggle, HuggingFace, academic), LLM-generated synthetic data, self-recorded data
- **Not Allowed:** Web scraping, API abuse, copying proprietary datasets

Deliverables

1. **Code:** Clean, modular implementation with clear documentation
2. **DATA.md:** Document data sources, generation prompts, and preprocessing steps
3. **README.md:** Setup instructions, approach explanation, results, and 5 qualitative examples
4. **Model:** Trained weights and inference script

Evaluation Criteria

Criteria	Weight	What We Look For
Technical Depth	30%	Understanding of architecture, training dynamics
Code Quality	25%	Clean, modular, well-documented code
Data Strategy	20%	Creative sourcing, realistic distribution, quality validation
Practicality	15%	Runs on laptop, reproducible, sensible design choices
Communication	10%	Clear documentation and technical writing

Challenge 1: Offline Partner Assistant

Context

Delivery partners frequently lose connectivity in elevators, basements, and tier-3 cities. They need an assistant that works 100% offline on low-end Android phones (2-3GB RAM). Cloud APIs are not an option.

The Challenge

Build an on-device language model (<15M parameters, <50MB) that can perform ONE of the following tasks:

Task	Description	Example
A	Hinglish Command Parser: Parse Hinglish text commands into structured intents	"Bhai next order ka address batao" → {intent: "get_address", order: "next"}
B	Smart Reply Generator: Generate quick reply options for partner-customer chat	Customer: "Kitna time lagega?" → ["5 min mein", "Traffic hai, 10 min", "Pahunch gaya hoon"]
C	Indian Address Parser: Parse messy Indian addresses into structured format	"opp more megastore, 2nd flr, nr shiv mandir, HSR" → {landmark: "More Megastore", floor: 2, area: "HSR"}

Constraints

- Model size: <50MB (quantized)
- Inference: <100ms on mid-range mobile CPU
- Memory: <500MB RAM during inference
- Export to ONNX or TFLite

Data Generation Prompt

Use the following prompt with an LLM to generate training data. Modify as needed for your chosen task.

```
Generate 50 realistic Hinglish sentences that an Indian food delivery partner
might say or type while working. Mix Hindi and English naturally. Cover these
intents equally: get_address, call_customer, mark_delivered, mark_picked_up,
report_delay, navigation_help, order_issue, customer_unavailable. Include
variations in formality, regional accents, typos, and sentence length. Format
as JSON: {"text": "...", "intent": "...", "slots": {...}}
```

Challenge 2: Real-Time Search Ranking

Context

When a user searches "butter chicken near me", we have <20ms to score 500+ restaurants. At Swiggy scale, LLM APIs are too slow and expensive. We need a tiny, fast model that can run at scale.

The Challenge

Build a text relevance model (<10M parameters) that can perform ONE of the following tasks:

Task	Description	Example
A	Query-Restaurant Matcher: Score relevance between search query and restaurant	Query: "healthy salad lunch" + Restaurant: "Salad Days - Fresh & Healthy" → 0.92
B	Query-Dish Matcher: Score relevance between search query and menu item	Query: "spicy paneer" + Dish: "Paneer Tikka Masala - Extra Hot" → 0.88
C	Typo-Tolerant Fuzzy Matcher: Handle misspelled and transliterated queries	Query: "chiken biryni" → Match to "Chicken Biryani" with score 0.95

Constraints

- Batch inference: 500 items scored in <20ms (GPU) or <100ms (CPU)
- Model size: <20MB
- Must handle Hindi, English, and Hinglish queries

Suggested Public Datasets

- Kaggle: Swiggy Bangalore Restaurants
- Kaggle: Indian Food 101
- Yelp Open Dataset

Data Generation Prompt

```
Generate 100 realistic food delivery app search queries that Indian users would type. Cover these categories equally: exact dish names, misspelled dish names, cuisine types, dietary preferences (veg, keto, jain), occasion-based (party, quick lunch), attribute-based (spicy, less oily), Hinglish queries, vague/exploratory queries. Format as JSON: [{"query": "...", "category": "...", "expected_keywords": [...]}]
```

Challenge 3: Streaming Voice Understanding

Context

Partners need hands-free operation while riding. The model must run continuously on-device, processing audio in real-time. Cloud streaming would drain the battery and require constant connectivity which is often unavailable.

The Challenge

Build a streaming speech understanding model (<20M parameters) that can perform ONE of the following tasks:

Task	Description	Example
A	Wake Word + Command: Detect trigger phrase then parse following command	[Audio] → {wake: true, command: "order complete kar do"}
B	Keyword Spotting + Intent: Detect predefined keywords and map to intents	[Audio with "delivered"] → {keyword: "delivered", intent: "mark_delivered"}
C	Streaming Hinglish ASR: Real-time transcription optimized for delivery vocabulary	[Audio] → "customer bol raha hai gate pe aao"

Constraints

- Latency: <300ms from speech end to output
- Must work with background noise (traffic, wind) at SNR 10-15dB
- Model size: <30MB

Suggested Public Datasets

- Mozilla Common Voice (Hindi)
- Google Speech Commands v2
- ESC-50 / UrbanSound8K (for background noise)
- OpenSLR Hindi datasets

Data Generation Prompt

Generate text commands, then convert to audio using TTS (gTTS, Coqui TTS, or similar). Add noise augmentation.

```
Generate 100 realistic Hinglish voice commands that a delivery partner would
speak while riding. Use natural spoken style (not written). Keep commands short
(2-8 words). Cover intents: navigation, order status, customer contact, issue
reporting. Include filler words and natural speech patterns. Use romanized
Hindi only (no Devanagari). Format as JSON: [{"text": "...", "intent": "..."}]
```

Submission

1. Share a private GitHub repository OR a zip file
2. Include a Loom video (5-10 mins) walking through your approach and demonstrating the model
3. Email submission with subject: "AI Assignment - [Your Name]"

Good luck! We look forward to seeing your work.