# Itamar Home Assignment - RTB Creative Relevance (≈ 2 to 3 hours completion time)

## Scenario

You receive 150 page snippets and a creative brief for a Crocs campaign. Your task is to decide bid or no-bid and an appropriate CPM for each page using a lightweight, reproducible approach. In addition, outline a minimal production architecture that can serve these decisions with tight latency and cost constraints.

## Goals

1. Show clear reasoning and a small working relevance model/or LLM approach that beats a trivial similarity baseline.
2. Propose a pragmatic end-to-end system that can go from request to decision under realistic startup constraints.

## Materials

* [pages.csv](https://docs.google.com/spreadsheets/d/e/2PACX-1vSysNXA8HDae7aQzCVr8jE9bgBnZEjHKdVAGhj6IlGR1BV8QHDOvnSSQ2LEWopzY463PuwiBKFx8MfX/pub?gid=69240309&single=true&output=csv) – 150 rows: url, snippet (use for bidder app)
* [labeled\_examples.csv](https://docs.google.com/spreadsheets/d/e/2PACX-1vR4Fb4vWw39JOViNdy3qNtq2VC-2qklndzMup6BUHJ-eDQfVcLbFU0o60Qkaxj0qRwhxrvwYdTqZTgo/pub?gid=1010487581&single=true&output=csv) – 85 rows with labels 0 or 1 (use for training or as LLM context examples)
* [brief.txt](https://docs.google.com/document/d/e/2PACX-1vSDn1hQYTlkg1Ad6THuXxodl8kFPsysU9ufz6qOjdFmjonkFZ20zEI4B0V1cEB2y8D_pPgroItFo_3R/pub) – creative brief for a Crocs campaign (calculate embeddings)
* [test\_set\_1.csv](https://docs.google.com/spreadsheets/d/e/2PACX-1vRbGbWr_3jjggOIuoflK6D7hFILBZVRVFn32Oo2m45kviWnf19WDmjaesezr_tXrd1cRBDeeLD73Bl5/pub?gid=1325369954&single=true&output=csv) – 50 rows with labels for validation (use to evaluate accuracy of your chosen approach)

## Tasks

### A) Hands-on relevance and pricing

**Baseline**

* Encode snippets and the brief with any public embedding model.
* Compute cosine similarity, pick a threshold for bid or no-bid, map similarity to CPM.

**Learnable model**

* Use the 85 labeled\_examples.csv labeled examples with your chosen approach.
* **Choose one approach:**
  + **Traditional ML**: Fit a compact model (logistic regression, linear SVM, small tree, MLP) that outputs a relevance score
  + **LLM-based**: Use a fast-inference LLM as a classifier, providing labeled examples in the context window (no training needed)
* Compare to baseline on at least one metric such as ROC AUC or PR AUC.
* Evaluate your model on test\_set\_1.csv and report performance metrics.
* Make sure to include accuracy metrics as percentage
* Produce results with fields: url, bid, price, score.

**Approachability guidance**  
Treat this as a page-to-brief relevance problem. Higher relevance suggests a higher CPM within a small range. No RTB knowledge is required.

**Optional add-on**

* Add one brand-safety rule (e.g. block a toy deny list).
* Or add a single feature ablation (e.g. brief-only vs brief + keyword anchors).

### B) Production architecture slice

Design a small system that can make the above decision online.

**Constraints**

* Response budget: 100 ms inclusive of network to the exchange.
* Scale: peak 20k QPS, average 5k QPS.
* Model: note differences and tradeoffs here
* Reliability: graceful degradation if the model or store is unavailable.

**Describe and diagram**

* Data and features: where embeddings are computed, refreshed, and stored. Versioning approach.
* Online path: request arrives with url and snippet, retrieval or on-the-fly embedding, scoring, price mapping, decision.
* Caching: what you cache, suggested TTLs, and hit-rate targets.
* Latency plan: expected p50 and p99 per hop, timeouts, retries, circuit breakers.
* Observability: key metrics and alerts (latency, cache hit rate, bid rate, CPM distribution, model drift).
* Safety and policy: block-lists or sensitive category filters.
* Experimentation and roll-out: baseline vs learned switch, canary, rollback.
* Cost notes: main cost drivers and one cost-control lever.

Provide one diagram and a short write-up that calls out the most important trade-offs.

## Deliverables

* repo/
* report.pdf or notebook.ipynb (concise results & notes). Please include relevant performance charts (training/test loss, accuracy comparisons for all approaches)
* model.py implements your chosen approach (trains/loads model or sets up LLM pipeline) and predicts relevance
* serve.py tiny HTTP endpoint: POST {url,snippet} -> {bid,price}
* design.md ≤ 1 page system design with diagram
* diagram.png architecture sketch if not embedded in PDF
* results.json [{url, bid, price, score}]
* requirements.txt small dependency set required to run the project
* **PUBLIC\_URL.txt** **REQUIRED**: operational public URL for deployed bidder endpoint (deploy to Railway, Render, Fly.io, Vercel, or Hugging Face Spaces)

## Notes

* For traditional ML: keep models small, no heavy GPU training. For LLM: use efficient inference APIs.
* Endpoints must be deployed and accessible via public URL for testing.
* If you skip something due to time, note what you would do next and why.