# Take-Home Assignment: Senior ML / GenAI Engineer - LLM Agent for Ad Campaign Creation

**Estimated time:** 4–6 hours

## Objective

You will build an **LLM-driven agent** that converts a short campaign brief into a machine-readable ad campaign plan plus ad creative variants.

The agent should:

- Interpret a brief (goal, product summary, budget, audience hints).
- Produce a structured plan that downstream systems could ingest (campaign metadata, budget split, audience segments, ad groups).
- Generate multiple ad creative variants per ad group (headline, body copy, CTA) with short justifications.
- Run basic automated consistency checks and surface any issues.
- Provide documentation about prompt choices, failure modes, and how you validated outputs.

## What to submit (must-have)

Produce a repository (GitHub link or zip) with the following **minimum** contents:

1. **Agent implementation (code)**
    - CLI or small script that accepts a JSON brief and returns JSON plan. (Python preferred but any language OK.)
    - If you call a remote LLM, use env vars for keys; do not commit keys.
2. **Prompt files + commentary**
    - Exact system + user prompts used.
    - Short notes (200–400 words) explaining prompt design and known failure cases.
3. **README** (how to run)
    - Setup, dependencies, run commands, how to run with a mock LLM, examples of input/output.
4. **Examples**
    - At least 2 example input briefs and the agent's outputs saved as JSON.
5. **Automated checks & tests**
    - At least one unit/integration test that validates output structure and a consistency check (e.g., budget sums to total, required fields present).
6. **1-page evaluation report**
    - How you validated outputs, key limitations, one short plan for improving hallucination and grounding in production.

## Good-to-have (bonus, do if time permits)

These are optional but valuable:

- Retrieval grounding: the agent consults a small KB (CSV/JSON) of product facts or past ad metrics to ground claims.
- A simple scoring heuristic (toy model or rules) that ranks creatives by expected relative performance.
- Dockerfile to run the agent.
- Small UI (Streamlit) to enter a brief and view results.
- Logging/metrics (token counts, hallucination flags).

## Input & expected output (format examples)

### Example input brief (use this JSON format)

```
{
  "campaign_id": "cmp_2025_09_01",
  "goal": "increase trial signups by 20% over next 30 days",
  "product": {
    "name": "FocusFlow",
    "category": "productivity app",
    "key_features": [
      "AI-assisted task prioritization",
      "calendar integration",
      "offline mode"
    ],
    "price": "free trial (14d) then $9.99/mo"
  },
  "budget": 5000,
  "channels": ["search", "social"],
  "audience_hints": ["young professionals", "remote workers"],
  "tone": "confident and concise"
}
```

### Minimal expected output (JSON excerpt)

```
{
  "campaign_id": "cmp_2025_09_01",
  "campaign_name": "FocusFlow Trial Push Sep2025",
  "objective": "trial_signups",
  "total_budget": 5000,
  "budget_breakdown": {
    "search": 3000,
    "social": 2000
  },
  "ad_groups": [
    {
      "id": "ag_1",
      "target": {"age": "25-40", "behaviors": ["remote work", "productivity apps"]},
      "creatives": [
        {
          "id": "c_1a",
          "headline": "Get More Done — Free 14-Day Trial",
          "body": "AI task prioritization that fits your calendar. Try FocusFlow free for 14 days.",
          "cta": "Start Free Trial",
          "justification": "Highlights trial + AI feature; concise for search."
        },
        ...
      ]
    },
    ...
  ],
  "checks": {
    "budget_sum_ok": true,
    "required_fields_present": true
  }
}
```

Make sure your output matches the schema you document.

## Constraints & practical guidance

- **Timebox:** 4–6 hours. Prioritize must-haves. If you run out of time, submit a working core plus clear notes about unfinished items and how you would complete them.
- **LLM choice:** any model is acceptable (hosted or local). If you don't have API access, mock responses but include the real prompts and explain what the live run would return.
- **Keep it small and reproducible:** a single script or small CLI is preferred over a large multi-service repo.
- **Be explicit:** document assumptions (e.g., budget granularity, channel definitions).

- **Avoid copyrighted content:** outputs should be original.

## What we're looking for

- **Structured thinking:** can you design an output schema and stick to it?
- **Prompt engineering discipline:** system prompt, output constraints, and detectable failure cases.
- **Grounding & verification:** practical steps to reduce hallucination and detect nonsense.
- **Reproducibility:** clear README and tests so a reviewer can run your agent quickly.
- **Judgement:** reasonable assumptions, trade-offs, and a short plan for production hardening.

## Submission checklist

- `README.md` — run instructions and high-level notes.
- `agent/` — code. One command to run (e.g., `python run_agent.py examples/brief1.json`)
- `prompts.txt` (or `.md`) — system & user prompts + design notes.
- `examples/` — at least 2 briefs and corresponding outputs.
- `tests/` — one or two tests that validate schema and consistency checks.
- `report.pdf` or `report.md` — one-page validation/limitations document.
- Optional: `Dockerfile`, `ui/`, or `kb/` (if you used a KB).

## Final notes

- This is a short exercise. We value clarity and reproducibility over bells and whistles.
- If you mock LLM responses, be explicit about why and include the intended live prompts.

Good luck.