upliance.ai - AI Product Engineer (Conversational Agents) assignment details

Task

Build a **minimal AI Game Referee chatbot** that can run a short game of **Rock–Paper–Scissors–Plus**, enforcing rules, tracking state, and responding intelligently to user inputs.

This assignment is designed to evaluate:

- Logical reasoning
- Agent design
- ADK usage
- Product clarity
- Engineering communication

Game Overview

The chatbot acts as a **referee** for a game played between the user and the bot.

Game Rules

- The game is **best of 3 rounds**
- Valid moves:
    - rock
    - paper
    - scissors
    - **bomb** (can be used **once per game**)
- bomb beats all other moves
- bomb vs bomb → draw
- Invalid input wastes the round
- After 3 rounds, the game must end automatically

Functional Requirements

1. Game Flow

The bot must:

- Explain the rules in **≤ 5 lines**
- Prompt the user for a move
- Validate and interpret user input
- Decide and explain the round outcome
- Track round count and score
- End the game automatically with a clear result

2. Logic Constraints

- bomb may be used **only once per player**
- Invalid inputs must be handled gracefully (no crashes, no dead ends)
- The game must not exceed **3 rounds**
- State must persist across turns

3. Technical Constraints (Important)

Languages

- **Python**

Frameworks / SDKs

- Must use **Google ADK**
- You may use:
    - ADK agents
    - ADK tools / functions
    - Structured outputs (schemas, JSON, etc.)

Tooling Rules

- You must define **at least one explicit tool**, for example:
    - `update_game_state`
    - `validate_move`
    - `resolve_round`
- Game state **must not live only in the prompt**
- Tools should be used for **state mutation or validation**

What NOT to use

- No databases
- No external APIs
- No UI frameworks
- No long-running servers

Assume the game runs in a simple conversational loop (CLI or chat-style interface).

4. Architecture Expectations

Your solution should clearly separate:

- **Intent understanding**
  (What did the user try to do?)
- **Game logic**
  (Is it valid? Who won the round?)
- **Response generation**
  (What should the user see next?)

You do **not** need multiple agents, but clean separation is expected.

5. Output Requirements

- Clear, round-by-round feedback
- Explicit indication of:
  - Round number
  - Moves played
  - Round winner
- Final result:
  - User wins / Bot wins / Draw

Deliverables

Submit:

- A **single file** or **small repo**
- A short **README** (½–1 page) explaining:
  - Your state model
  - Your agent/tool design
  - Any tradeoffs you made
  - What you would improve with more time

What We'll Evaluate

We are **not** looking for polish or perfect UX.

We care about:

- Correctness of logic
- Quality of state modeling
- Clarity of agent boundaries
- Use of ADK primitives
- Ability to explain decisions

Deadline: You have **48 hours/2 days** to submit the assignment through the ***google form link*** provided in the mail.