**Espyr Hackathon Challenge Document**

## **🏆 Challenge Overview**

Welcome to the **Espyr Hackathon!** Your mission is to **build a prototype of an AI-powered DSA Preparation Coach** that guides users through solving curated interview problems. Think of it like the Leet Code — but with a smart mentor guiding, challenging, and evaluating you at every step.

### **🎯 Goal**

Build a web-based interface and logic system where:

* Users solve problems through a Leet Code-like UI.
* An AI **Mentor Agent** guides the user during the thought process.
* An AI **Code Agent** helps verify, analyze, and test user code.
* The system **adapts to different skill levels** (beginner → advanced).

💡 Note: You may design your own agent structure if you feel a different configuration better suits the user journey. Just be sure it fulfills the same goals of mentorship, feedback, and adaptive learning.

## **🧠 Default Agent Roles (Recommended)**

You are free to design your own agent structure, but here’s a proven setup that you can take inspiration from:

1. Mentor Agent: Drives thought process, gives hints, adapts tone.

2. Code Agent: Analyzes code, generates test cases, detects bugs.

3. Evaluation Agent: Reviews performance holistically.

4. Persona Manager: Adjusts tone and behavior based on user level.

5. Orchestrator Agent: Manages transitions (approach → code → evaluation).

## **🔧 Core Features to Build**

### **1. 🧠 Mentor Agent**

* Ask the user for their approach first.
* Provides hints when the user is stuck.
* Motivates and adapts tone based on user level.
* Triggers the code editor when the user is ready.

### **2. 💻 Code Agent**

* Activates only after Mentor gives the go-ahead.
* Runs tests, catches bugs, and suggests performance improvements.
* Sends the user back to the mentor if logic is incorrect.

### **3. 🧪 Evaluation Summary**

* After solving, provide a session recap:
  + How clear was the user's thought process?
  + How many hints were used?
  + Was the final code optimal?

### **4. 🌐 Web Interface**

* Left panel: Problem statement and AI chat.
* Right panel: Code editor (Monaco editor preferred).
* Keep it simple, like Leet Code meets ChatGPT.

## **🧩 Tools You Can Use**

👇 Here are some suggestions:

**Frontend**: React, Next.js, Tailwind CSS

**Editor**: Monaco Editor or Code Mirror

**AI Backend** *(Limited credits will be provided)*:

* OpenAI GPT-4 (via API)
* Hugging Face Transformers (local)

**Agent Orchestration**:

* Lang Graph (recommended)
* Crew AI / Auto Gen (if exploring multi-agent logic)
* React pattern manually

**State Tracking**:

* Use finite state machine or context-based logic

## **🧠 Understanding Skill Levels**

The AI Mentor must adapt dynamically to user skill. Here's how:

|  |  |  |
| --- | --- | --- |
| Level | Behavior Pattern | Mentor Style |
| Beginner | Tries brute-force, misses edge cases | Motivational, gives base hints |
| Intermediate | Knows data structures, may miss optimal path | Encouraging, nudges toward patterns |
| Advanced | Recognizes optimal patterns quickly | Acts like an interviewer, tests edge logic |

## **🔄 Sample Scenario Walkthroughs**

### **🧑‍🎓 Beginner**

* **Mentor**: "What do you think the approach could be?"
* **User**: "Maybe try every pair and check the sum."
* **Mentor**: "That's a good start! What happens if the input size is 10^4?
* **Hint**: "Can a HashMap help you reduce time?"

### **🧑‍💻 Intermediate**

* **User**: "I’ll use HashMap to store indexes."
* **Mentor**: "Great! Watch out for duplicates. What is complexity?
* **Mentor**: Triggers code editor with TOOL\_CALL: open editor

### **🧑‍🏫 Advanced**

* **User**: "This looks like a two-pointer pattern."
* **Mentor**: "Nice recognition. Can you prove why it's optimal?"
* **Code Agent**: Gives edge case with sorted array and negative numbers.
* **User**: Adjusts and passes.
* **Mentor**: "Perfect. Final complexity?

## **🗓️ Submission Timeline**

* **Kickoff**: 3rd June 25
* **Deadline**: 10th June 25

## **✅ Submission Requirements**

* GitHub Repo (with README, setup, agent logic)
* Hosted Demo (optional but preferred)
* Video Walkthrough (max 5 mins)

## **🧰 Tips for Success**

* Keep the UI minimal but functional.
* Show agent reasoning like hint level, feedback, retries.
* You don’t need a perfect LLM — you need a smart orchestration.
* Don't hardcode solutions — make the mentor generic for any DSA problem.
* Make sure the system adapts the mentor tone to the user’s proficiency.

Good luck — and let your agents *think* like mentors! 🤖