# AI-Driven Interview System: 1-Day Prototype

## Goal

Build a minimal prototype of an AI-Driven Interview System that:  
- Dynamically generates a small set of interview questions.  
- Evaluates and scores the candidate’s responses.  
- Produces basic feedback.

## 1. Agents

### Question Agent

Generates three interview questions based on a given job description.

### Response Evaluation Agent

Receives the candidate’s responses and assigns a score to each.

### Validation Agent

Validates the scores from the Response Evaluation Agent.  
Produces final feedback and summary.

## 2. Workflow

### Question Agent

Generates 3 questions dynamically (e.g., “What is your experience with X?”).

### Candidate

Provides responses to these questions.

### Response Evaluation Agent

Analyzes each response, assigns a score (e.g., 1–5), and notes brief comments.

### Validation Agent

Confirms or adjusts the scores.  
Outputs a short summary report with:  
- Questions  
- Responses  
- Scores  
- Feedback

## 3. Shared Context

Store basic context such as:  
- Candidate’s job title/role (e.g., “Backend Developer”).  
- Current question and past responses.  
  
Use any simple method (in-memory dictionary, small DB) for storing and retrieving this context.

## 4. Requirements

### Functional Requirements

- Implement the 3 agents (Question, Response Evaluation, Validation).  
- Orchestrate them in a workflow.  
- Use shared context so agents can access stored data.

### Technical Requirements

- Use any AI framework (e.g., LangChain, AutoGen, CrewAI, OpenAI Assistant API) and justify your choice.  
- Choose a local LLM or an external LLM API (e.g., OpenAI GPT).  
- Store context in memory or a simple database (e.g., Redis, SQLite).

- Build a **FastAPI** application to expose the system as an API. Implement all routes using asynchronous programming for high performance and scalability.

- Log each interview session with the following details

* + Candidate ID.
  + Job title/role.
  + Matching timestamp.
  + S3 URL or local file path for saved data.
  + Use DynamoDB (or SQLite/local JSON-based logs) to store the logs asynchronously.

- Store the generated questions, responses, scores, and feedback as JSON objects in an S3 bucket (or a local equivalent, such as file storage). Use asynchronous methods for uploading files to ensure non-blocking I/O.

- Make the code modular and easy to extend.

## Deliverables

### Code

* A fully functional implementation of the task in your chosen framework.
* Include clear documentation on how to run the solution.

### Short Write-Up

* Explain your approach to solving the task.
* Justify your choice of framework and key design decisions.
* Highlight any challenges you faced and how you overcame them.