Full Stack Developer – Technical Assignments[#]

📍 **Location:** Mumbai, India
🧪 **Assignment Type:** Practical Coding Test
🕒 **Expected Duration:** 4–7 hours

Assignment 1: Employee Feedback Portal

📌 Project Brief

Build a simple **"Employee Feedback Portal"** where employees can submit anonymous feedback, and an admin can view and categorize the feedback.

📦 Tech Stack Requirements

- **Frontend:** React.js (with functional components and hooks), Redux (optional)
- **Backend:** Node.js with Express.js
- **Database:** MongoDB or MySQL (candidate's choice)
- **API:** RESTful API
- **Bonus:** Use of GitHub repo, basic styling with CSS or any UI library (e.g., Bootstrap)

🎯 Functional Requirements

🧑‍💼 Employee Side

- A form where an employee can:
  - Enter feedback (text area – required)
  - Choose category: [Work Environment, Leadership, Growth, Others]
  - Submit anonymously (no login required)

👩‍💼 Admin Side

- View all submitted feedback in a table with:
  - Feedback text
  - Category
  - Submission time
- Ability to:
  - Filter feedback by category
  - Mark feedback as **reviewed**

[#] Assignment for Full Stack Development Role @ Catallyst

    - Delete feedback (optional)

🔧 Backend API Requirements

Design the following APIs:

- `POST /feedback` – Submit feedback
- `GET /feedback` – Get all feedback
- `GET /feedback?category=xyz` – Filter by category
- `PATCH /feedback/:id/reviewed` – Mark as reviewed
- `DELETE /feedback/:id` – Delete feedback (optional)

📊 Evaluation Criteria

Criteria	Weightage
Code Quality (clean, modular, DRY)	20%
React Implementation (form, state handling, UI)	20%
Node.js API Design (REST principles)	20%
MongoDB/MySQL Design and Integration	15%
Functional Completeness	15%
Optional Features (UI polish, filters, GitHub, CI/CD)	10%

📁 Deliverables

- GitHub or Zip of full project code
- Brief `README.md` with:
    - How to run the app
    - API structure
    - Assumptions made
    - What is complete and what's not

Assignment 2: Learning Portal – Reading Synopsis + Gen AI Grading

📌 Project Brief

Build a **Learning Portal** where:

1. Participants can submit a reading synopsis through a web form.
2. Submissions are saved to a database.

3. The system simulates sending the synopsis to a **GenAI service** and returns a **score** + **feedback**. Checks if the submission is done with Gen AI content in a scale of 1-10 (10 being highest) and gives 0 score if content is more than 8 in scale.
4. Admin can view all submissions and review AI responses.

💡 Use Case

Participants in a corporate learning program submit summaries of assigned readings (e.g., articles or case studies), and a GenAI engine provides a score and written feedback.

🎯 Core Features

🧍 Participant Interface:

- Form fields:
  - Name
  - Email
  - Synopsis (textarea – min 100 words)
  - Topic/Reading Title
- On submission:
  - Data is saved
  - System fetches AI score + feedback (simulate via mock API)
  - Display response on screen

👩‍💼 Admin Interface:

- Table listing:
  - Name, Title, Score, Feedback, Submission Time
- Filter by title or score
- Optional: mark as reviewed

⚙️ Tech Requirements

- **Frontend:** React.js (functional components, hooks)
- **Backend:** Node.js with Express.js
- **Database:** MongoDB or MySQL
- **Mock GenAI API:** Accepts synopsis text, returns mock:

```
{
  "score": 8.5,
  "feedback": "Well-structured and insightful summary."
}
```

📊 Evaluation Criteria

Criteria	Weightage
Frontend Experience & Form Validation	20%
API Integration & Async Handling	20%
Backend & DB Design	20%
Realism of GenAI Integration Simulation	15%
UX Flow and Completion	15%
Bonus (UI polish, Admin filters, etc.)	10%

📁 Deliverables

- GitHub repo or zip file with code
- README including:
    - Setup steps
    - How GenAI is simulated
    - What's complete/incomplete

⏱ Estimated Time

- 5–7 hours for a solid prototype

Note to Candidates: Choose **either one** of the assignments to showcase your skillset. A well-finished, thoughtful solution with clean code, documentation, and good UI/UX will be prioritized.

Good luck! 🚀