Alcovia - Full Stack Engineering Intern Assignment

Context

In the current scope of Alcovia, we are moving beyond simple data tracking. We are building an **"Intervention Engine"** - a system that detects when a student is falling behind in real-time and automatically triggers a mentorship loop.

We are looking for a **Product-First Engineer** - someone who understands that code is just a tool to solve human problems. You won't just be pushing pixels; you will be architecting a system where **Human (Mentor)**, **Machine (App)**, and **Database (Logic)** talk to each other seamlessly.

Task at Hand

1. **Build a "Closed-Loop" System:** Create a working prototype that connects a Student App, a Backend Server, and an Automation Workflow.
2. **Deploy for Usage:** We do not want to run your code locally. You must deploy the app to the web so we can interact with it immediately.
3. **Timeline:** You have **48 hours** to architect, build, and ship.

Tech Stack Constraints

- **Backend:** Node.js or Python.
  - *Constraint:* Must use a SQL Database (Postgres, MySQL, or Supabase). No NoSQL/Firebase.
- **Automation:** n8n
  - *Note:* You can use the cloud version or self-hosted.

- **Frontend:** React Native or Flutter.
    - *Constraint:* Must be deployed as a Web App (Flutter Web or Expo Web).

Problem Statements at Hand

1. The "State of the Student" Backend

Context: A student's progress isn't static. If they perform poorly on a daily check-in, their status in our system changes. We need a backend that manages this state and enforces "Digital Rigour."

Problem Statement: Design and build a backend that manages the "Intervention State."

Requirements:

1. **Database:** Design a SQL Schema for Students, Daily_Logs (Focus Time, Quiz Score), and Interventions.
2. API Logic: Create a POST endpoint /daily-checkin that accepts:
   { "student_id": "123", "quiz_score": 4, "focus_minutes": 30 }
3. **The Logic Gate:**
    - **Success:** If quiz_score > 7 AND focus_minutes > 60 → Log success. Return { status: "On Track" }.
    - **Failure (The Lock):** If stats are low → Update the student's status in the DB to "Needs Intervention". **Trigger the n8n webhook** (see below). Return { status: "Pending Mentor Review" }.

2. The "Human-in-the-Loop" Automation (n8n)

Context: When a student fails, an algorithm cannot simply solve it. A human mentor must intervene. However, we cannot wait for mentors to constantly refresh a dashboard to see who is failing. The system must notify them and wait for their command.

Problem Statement: Create an n8n workflow acting as the "Mentor Dispatcher."

Requirements:

1. **Trigger:** Receives the webhook from your Backend when a student fails (from Problem 1).

2. **Action:** Sends a notification (Email or Slack) to *you* (acting as the Mentor) with the student's stats.
3. **The Wait (Crucial):** The workflow must **pause execution** and wait for the Mentor to click a link or approve a remedial task (e.g., "Assign: Read Chapter 4").
4. **The Loop Back:** Once the Mentor clicks/approves, n8n hits your Backend endpoint /assign-intervention to update the DB with the new task and "Unlock" the student.

3. The "Focus Mode" App (Frontend)

Context: The student interface must react to their real-time status. It is not just a passive display; it enforces the mentorship rules.

Problem Statement: Build a specific "Focus Mode" interface that reacts to the Backend State.

Requirements:

1. **Normal State:** Student sees "Start Focus Timer" and "Daily Quiz" input.
2. **Locked State:** If the student submits a bad score (from Problem 1), the app must immediately **Lock Down**. All features are disabled.
    - *Display:* "Analysis in progress. Waiting for Mentor..."
3. **Remedial State:** Once n8n updates the DB (from Problem 2), the app unlocks but shows **ONLY** the remedial task (e.g., "Task: Read Chapter 4").
4. **Completion:** Student clicks "Mark Complete" to return to Normal State.

The "Chaos" Component (System Design)

Context: In Problem 2, the automation waits for a human mentor.
Question: What happens if the Mentor doesn't reply for 12 hours? The student is locked out of the app indefinitely.

Task: In your README, briefly describe a "Fail-Safe" mechanism you would architect to handle this latency (e.g., auto-unlocking after X hours or escalating to a Head Mentor).

Bonus Challenges

Want to guarantee an interview? Implement one of these.

1. The "Cheater" Detection: Since this is a Web App, if the student switches tabs or minimizes the browser during the "Focus Timer", the session should automatically fail and log a penalty.
2. Real-Time Magic (WebSockets): In the "Locked State", the app typically needs to poll the server to see if the Mentor has unlocked it. Instead, implement WebSockets (Socket.io) so that the moment the Mentor clicks the email link, the Student's screen unlocks *instantly* without them refreshing the page

Submission Deliverables

We need to see this working. Submit the following via the Google Form:

Deliverable	Requirement
1. Live App Link	Deploy your React Native/Flutter app to the web (e.g., Vercel, Netlify, Flutter Web). We must be able to open the URL and test the "Lockout" flow ourselves.
2. GitHub Repository	Public link containing clear folders for: Server, Client, and n8n_Workflow (JSON export).
3. Loom Video	A short (max 5 min) walkthrough showing the full loop: 1. Submitting a "Bad Score" on the live app. 2. The App entering "Locked State". 3. The n8n execution triggering the email. 4. You (Mentor) assigning a task. 5. The App unlocking and showing the Remedial Task.

This assignment has been designed to understand how you think and how you connect systems. Use AI as an enabler, but ensure the logic holds up when we stress-test it.

SUBMISSION DEADLINE : 9pm, 24th November 2025
SUBMISSION FORM : https://forms.gle/1Qq9bcR7KPE6ZAgUA