Round 2 – Team Technical Task

QOTD Module Integration & Deployment

Overview

Congratulations on being shortlisted for **Round 2 (Team Technical Task)**.

This round is a **collaborative engineering challenge** focused on **integration, scalability, cost awareness, and production readiness**.

Please read this document carefully. All expectations are clearly defined. **Any deviation or shortcut may lead to disqualification.**

Context (Important)

In **Round 1**:

- Frontend Developers built the **QOTD UI**
- Backend Developers implemented **QOTD APIs, submission logic, and evaluation**

In **Round 2**, with the help of **Full-Stack Developers**, teams are **NOT expected to start from scratch**.

Your goal is to:

Merge, integrate, improve, and deploy a production-ready QOTD module.

This round evaluates your ability to:

- Understand existing code
- Collaborate across roles

- Debug and integrate systems
- Ship a real, working feature under time constraints

Team Composition

Each team will consist of:

- Minimum **2 Frontend Developers**
- Minimum **2 Backend Developers**
- Minimum **1 Full-Stack Developer**

All members must actively contribute.

Core Goal

Build a **globally accessible, scalable QOTD (Question of the Day) module** that:

- Works end-to-end
- Is production-ready
- Can scale to **1000s of users per day**
- Is cost-optimised
- Is deployed live

Functional Requirements (Mandatory)

---

1. QOTD Question Management

- QOTD can be fetched from a **dummy admin source**
    - Hardcoded admin input or mock admin API is allowed
- One question is displayed globally for the day
- Each question must include:
    - Title
    - Difficulty (Beginner / Intermediate / Advanced)
    - Problem statement
    - Sample input & output

---

2. Code Execution (Online Compiler)

Users must be able to:

- Write code
- Click **Run**
- View execution output

Mandatory Supported Languages:

- Python
- Java

Coding Environment:

- Dark mode allowed
- Must strictly follow **TechLearn Blue Theme**
- Any other UI theme → **entire team shall be disqualified**

---

3. Submission Rules

- Users can submit **only once per day**
- **Free users:**
    - Maximum **2 code runs**
    - Maximum **1 submission**
- **Paid users:**
    - Maximum **4 code runs**
    - Maximum **1 submission**
- Evaluation logic:
    - Can be basic or mock
    - Must clearly return **correct / incorrect**
    - Submission data must be stored

---

4. Daily Leaderboard

- Only **logged-in users** appear on leaderboards
- Separate leaderboard for each difficulty:
    - Beginner
    - Intermediate
    - Advanced
- Leaderboard data:

    - Exists only for the current day
    - Automatically resets the next day

Access Rules:

- Free users:
    - Can see score
    - Cannot appear on leaderboard
- Paid users:
    - Appear on leaderboard
    - Can view solutions and personal stats

---

5. Paid User Features (Design-Level)

Paid users should be able to:

- View official QOTD solution
- View personal stats or streaks (mock data allowed)

Payment gateway implementation is **NOT required**.
Design-level handling is sufficient.

Technical Expectations

Frontend

- Clean UI
- Clear UX

- Responsive design
- Basic accessibility practices

Backend

- RESTful APIs
- Proper HTTP status codes
- Basic rate-limiting logic (implementation or design explanation)
- Scalable architecture

Database

- Any option allowed:
    - MongoDB
    - In-memory
    - JSON storage

AI Tools

- AI tools are allowed to improve productivity
- **Direct deployment from AI site builders (Lovable, Replit, Claude, etc.) is strictly prohibited**

Deployment (Mandatory)

- Frontend and Backend must be deployed
- Live URLs must be accessible

- Any hosting platform is allowed:
    - Render
    - Vercel
    - Railway
    - Fly.io
    - Similar platforms

Submission Requirements

Each team must submit:

1. **GitHub Repository Link**
    - Integrated frontend + backend
2. **Live Deployed URLs**
3. **README.md** including:
    - System architecture overview
    - Integration approach
    - Scalability strategy
    - Cost optimisation reasoning
    - Submission storage & evaluation logic
    - Improvements planned with more time

Timeline

- **Duration:** 48 hours
- Start once team details are shared
- Late or incomplete submissions **will not be evaluated**

Disqualification Criteria (Strict)

- UI theme not matching TechLearn blue
- Direct deployment from AI platforms
- No live deployment
- Broken or incomplete end-to-end flow
- Ignoring run/submission limits
- Hardcoding without explanation

Evaluation Criteria

Teams will be evaluated on:

- Collaboration and teamwork
- Code integration skills
- Debugging capability
- System design thinking
- Scalability and cost awareness

- Production-readiness mindset

Final Note

This round closely simulates **real-world engineering teamwork**.
Focus on clarity, ownership, and delivery—not just features.

Good luck, and happy building.

TechLearn Solutions