Coding Assignment

Duration

You will have **2 hours** to complete this assignment in one continuous session.

Objective

We want to assess your ability to design and implement a small but production-ready system, simulating an actual work environment and requiring thoughtful tradeoffs along the way.

Problem Statement

Build a **Quiz Management System** with:

1. **Admin Panel**
    - Ability to create a quiz with:
        - Quiz title
        - A few questions of various types (MCQ, True/False, text, etc.)

2. **Public Page**
    - A page where a quiz can be taken by anyone
    - Display results after completion (e.g., score or correct answers)

Tech Stack/Tools

Frontend	React (Next.js or similar) or React Native, or Flutter with any UI framework (Tailwind or similar) of your choice
Backend	Any language/framework of your choice
Database	Neon DB (Postgres) or MongoDB Atlas
IDE	VSCode or Cursor or similar
AI Coding Agent	Any CLI agent of your choice, you are free to use multiple agents in parallel

Deliverables

- One GitHub repo with all the source code
    - A PLAN.md describing:
        - Assumptions, scope, and approach
        - Any scope changes during implementation

        - A short reflection (~5 min) at the end on what you would do next if you had more time
    - Minimum 4 commits
        - Make at least one commit every 30 minutes
- Recording of your session (screen + audio).
    - Include a short demo of your final work at the end
- Deployment link (good to have)

Notes

- The brief is intentionally light — we want to see how you scope, prioritize, and make trade-offs.
- Whatever you build should feel production-ready. Less features are fine, as long as the system works reliably.
- Plan first: Create a PLAN.md describing your assumptions, scope, and high-level architecture, including schema. Update your PLAN.md if you adjust scope or make significant changes while coding.

Evaluation Criteria

1. Product Thinking & Planning

- Creation of a PLAN.md before coding and updating it if scope or approach changes
- Picking a realistic scope for the timebox
- Clear explanation of trade-offs (what was included vs. skipped)
- High-level architecture and assumptions are well thought out

2. AI Collaboration

- Thoughtful use of AI to assist in coding
- Validation and adaptation of AI output rather than blindly copying
- Awareness of AI limitations and integration in a meaningful way
- AI usage demonstrates understanding, not over-reliance

3. Communication & Reasoning

- Thought process communicated clearly while coding
- Decisions are justified, including assumptions and trade-offs
- Explanation of architecture and implementation approach
- Adaptability shown when scope or approach changes

4. Production Readiness

- Core flow works reliably
- System feels deployable even with limited scope
- Key features are functional and tested
- Implementation reflects a realistic production-ready approach

5. Code Quality & Architecture

- Code is clean, readable, and idiomatic
- Structure is sensible (frontend, backend, database clearly separated)
- Basic errors or invalid inputs are handled gracefully
- Clean commit history and effective use of comments to capture thought process