Swipe Internship Assignment — AI-Powered Interview Assistant (Crisp)

Goal: Build a React app that works as an AI-powered interview assistant. It must:
Provide **two tabs**: Interviewee (chat) and Interviewer (dashboard). Both stay synced.

- **Interviewee (chat)**
  - Let a candidate upload a resume (**PDF required, DOCX optional**).
  - Extract **Name, Email, Phone** from the resume; if anything is missing, the chatbot collects it **before starting the interview**.
  - Run a timed interview where AI generates questions and judges answers.
- **Interviewer (dashboard)**
  - List of candidates ordered by score.
  - Ability to view each candidate's chat history, profile, and final AI summary.
- Persist all data locally so closing/reopening restores progress.
- Support pause/resume with a Welcome Back modal.

Core Requirements

- **Resume Upload:** Accept PDF/DOCX. Extract following fields - Name, Email, Phone.
- **Missing Fields:** Chatbot prompts candidates to fill gaps before the interview starts. Like if the phone number is missing, it should 1st ask for it before starting the interview.
- **Interview Flow**
  - AI will dynamically generate questions for full stack (React/Node) role 1 by 1
  - 6 questions total: 2 Easy → 2 Medium → 2 Hard.
  - Questions are shown one at a time in the chat.
  - Timers per question: Easy 20 s, Medium 60 s, Hard 120 s.
  - When time runs out, the system automatically submits the answer and moves on.
  - After the 6th question, the AI calculates a final score and creates a short summary of the candidate.
- **App contains Two Tabs**
  - Interviewee Tab
    - Candidate chat flow — questions, answers, timers, and progress.
  - Interviewer Tab (Dashboard)
    - Shows a list of all candidates with their final score and summary.
    - Clicking on a candidate opens a detailed view showing all questions, answers, and AI scores for that candidate.
    - Search and sort functionality included.

- **Persistence (Data Storage)**
  - Use any state management lib to save **all timers, answers, and progress** locally.
  - If a candidate refreshes or closes the page, everything is restored on reopening.
  - Show a **"Welcome Back"** modal for unfinished sessions.

Tech

- You are free to use any libraries or tools you prefer.
- Suggested stack: **React + Redux (state & persistence, e.g., redux-persist / IndexedDB)**.
- Suggested UI: **Ant Design** or **shadcn** (or any other modern UI library).
- Build a **clean, responsive UI** (basic chat + tables).
- Implement **friendly error handling** (invalid files, missing fields)

Deliverables

1. Public GitHub repo + README.
2. Live demo (Vercel/Netlify)
3. 2–5 minute demo video
4. Submit form: https://forms.gle/Yx5HGCQzHFmHF1wM6

Notes

- We kept a few things ambiguous intentionally. Choose the best possible path to complete the assignment.
- You are encouraged to use AI APIs wherever possible, with no restrictions.

Who Should Not Apply

- Anyone **unwilling to relocate to Hyderabad** for the internship role.
- Anyone **unable to commit to a full 6-month internship**.
- Anyone **who can't start at least by December**.

We look forward to seeing your approach to this real-world project!