Coding Assessment - Frontend Engineer

Scenario

You are encouraged to implement as much of the scenario requirements as you have time for, based on the submission deadline communicated to you by your TA contact (5 days)

Building CloudHive's Feature Idea Portal

CloudHive, a fictional forward-thinking technology company, is in the process of developing a new product called *Integration Hub*. To foster innovation and encourage company-wide collaboration, CloudHive wants to build an **internal web application** where employees can propose feature ideas for Integration Hub and help shape its roadmap.

You have been tasked with creating a **proof of concept** for this idea submission tool. The goal is to design an intuitive and user-friendly interface that encourages employees to submit, explore, and vote on feature ideas.

Core Features to Implement:

1. **Idea Submission Form:**
   - Employees should be able to submit new feature ideas through a form that captures:
     - **Summary** (required) – A short title for the idea.
     - **Description** (required) – A detailed explanation of the idea.
     - **Employee** (required) – A dropdown of pre-defined employees to select who is submitting the idea.
     - **Priority** (optional - defaults to *Low*) – A selection of either *High*, *Medium*, or *Low*.
2. **Idea List & Voting System:**
   - Display a **stacked list** of submitted ideas, sorted by **upvotes descending** (most upvoted ideas at the top).
   - Each idea in the list should display:
     - Summary of the idea.
     - Name of the submitting employee.
     - Current **upvote** and **downvote** counts.
   - Employees should be able to **upvote** or **downvote** ideas directly from the list.
   - Employees should also be able to delete directly ideas from the list
3. **Idea Exploration:**
   - Allow users to **click on an idea** to view its **full description** and details.
4. **Search Functionality:**
   - Provide a **search bar** to help employees quickly find specific ideas based on keywords in the summary or description.

5. **Pagination:**
    - CloudHive expects **high engagement**, leading to a significant number of idea submissions.
    - To maintain usability and performance, the idea list should be **paginated** to display **20 ideas per page**.
6. **Visual Appeal:**
    - While this is a proof of concept, CloudHive expects a **clean, visually appealing interface** that promotes engagement. If the tool is successful, **formal Figma designs** will be provided to align the UI with CloudHive's brand in the future.

Technical Requirements

To ensure consistency and alignment with CloudHive's tech stack, please follow these guidelines when building the proof of concept:

1. **Framework & Setup:**
    - The project should be built using **Next.js 15** with **React 19**.
    - Initialize the project using the following command to set up **TypeScript**, **TailwindCSS**, **App Router**, and **pnpm** as the package manager:

```
npx create-next-app@latest --typescript --tailwind --app --use-pnpm
```

2. **Data Handling:**
    - **Server Actions** or **REST APIs** should be used for all data interactions.
    - **Employee List:**
        - Retrieve the list of employees from a **JSON file** stored in the project.
        - Use the following mock data as a starting point:

```
[
  {
    "id": "1",
    "name": "Alice Johnson",
    "profileImage": "https://randomuser.me/api/portraits/women/1.jpg"
  },
  {
    "id": "2",
    "name": "Bob Smith",
    "profileImage": "https://randomuser.me/api/portraits/men/2.jpg"
  },
  {
    "id": "3",
    "name": "Charlie Davis",
    "profileImage": "https://randomuser.me/api/portraits/men/3.jpg"
  },
  {
    "id": "4",
    "name": "Danielle Brooks",
    "profileImage": "https://randomuser.me/api/portraits/women/4.jpg"
  },
  {
    "id": "5",
    "name": "Ethan Parker",
    "profileImage": "https://randomuser.me/api/portraits/men/5.jpg"
  },
  {
    "id": "6",
    "name": "Fiona Mitchell",
    "profileImage": "https://randomuser.me/api/portraits/women/6.jpg"
  },
  {
    "id": "7",
    "name": "George Hamilton",
    "profileImage": "https://randomuser.me/api/portraits/men/7.jpg"
  },
  {
    "id": "8",
    "name": "Hannah Lee",
    "profileImage": "https://randomuser.me/api/portraits/women/8.jpg"
  },
  {
    "id": "9",
    "name": "Isaac Thompson",
    "profileImage": "https://randomuser.me/api/portraits/men/9.jpg"
  },
  {
    "id": "10",
    "name": "Jasmine Patel",
    "profileImage": "https://randomuser.me/api/portraits/women/10.jpg"
  }
]

```

- **Ideas List:**
  - Store ideas in an `ideas.json` file at the root of the project.
  - Implement **server actions** or **REST APIs** to:
    - **Fetch** ideas with **pagination** (20 ideas per page).
    - **Submit** new ideas.
    - **Upvote** or **downvote** existing ideas.
  - Database Persistence isn't required— the server actions or rest APIs should modify the `ideas.json` directly.

3. **UI Components:**
   - You are welcome to use the **RadixUI** library for any UI components in your application.
   - We recommend creating your own **custom components** that wrap RadixUI primitives to ensure consistency and flexibility in styling.
4. **Form Management:**
   - Use **react-hook-form** to manage the state and validation of the **idea creation form**.
   - Ensure proper form validation for required fields such as **Summary**, **Description** and **Employee**
5. **Routing:**
   - Implement routing using **Next.js App Router** with the following structure:
     i. **Ideas Dashboard (Default Route):**
       - Displays the list of ideas with voting, searching, and pagination.
     ii. **Create Idea Route:**
       - A dedicated page with the **idea submission form**.
     iii. **Idea Details Route:**
       - A page to view the **full details** of a selected idea, including the full description and voting options.
6. **Client-Side State Management:**
   - Use **TanStack Query** to handle all client-side data fetching and synchronization.
   - Ensure the UI updates seamlessly when new ideas are submitted, or when votes are cast, reflecting the latest data without manual refreshes.
7. **Additional Notes:**
   - **No authentication or authorization** is needed, as this is an internal proof of concept.

    - Focus on clear code structure and maintainability, with an emphasis on **React best practices** and **Next.js conventions**.

Submission

- **Create a Public GitHub Repository:**
    - Push your project code to a **public repository** on GitHub.
    - Ensure the repository includes:
        - A clear and concise **README** with:
            - Instructions on how to run the project.
            - Any **design constraints**, **assumptions**, or **technical explanations**
            - Notes on **potential future enhancements** or improvements to the tool.
- **Share the Repository Link:**
    - Once your code is pushed, email the link to your **TA contact** .