Frontend Developer Intern - Assignment

📋 Overview

This assignment evaluates your frontend development skills, attention to detail, and ability to implement designs accurately. You'll be building a desktop web interface based on the provided Figma design.

Figma Design

Figma Prototype

🎯 Task Requirements

What to Build

Implement the complete design from the Figma file with pixel-perfect accuracy for **desktop screens only**. The design should be:

- **Pixel-perfect** matching the Figma specifications
- **Interactive** with smooth transitions and hover states
- **Accessible** following WCAG 2.1 guidelines

Technical Stack

Required:

- React (with TypeScript preferred)
- Tailwind CSS
- Modern ES6+ JavaScript

Optional (Bonus Points):

- Next.js for framework
- Framer Motion for animations

✅ Evaluation Criteria

Your submission will be evaluated on:

- Design Accuracy
- Code Quality
- Performance
- Best Practices

📤 Submission Guidelines

What to Submit

1. **GitHub Repository**
    - Public repository with complete source code
    - Clear folder structure
    - All assets included
2. **Live Demo**
    - Deploy on Vercel, Netlify, or similar platform
    - Provide working URL
3. **Documentation**
    - README.md with:
        - Setup instructions
        - Tech stack used
        - Key features implemented
        - Any assumptions made
        - Time spent on the assignment

🚀 How to Submit

1. **Create GitHub Repository**
    - Initialize with proper .gitignore
    - Make regular commits showing your progress
2. **Deploy Your Application**
    - Deploy to Vercel (recommended) or Netlify
    - Ensure all features work in production

3. **Submit Your Work**
    - Email the following to: [YOUR_EMAIL]
        - GitHub repository URL
        - Live demo URL
        - Your resume (PDF)
        - Brief cover letter (optional)

Subject Line: Frontend Intern Application - [Your Name]

⏰ Timeline

- **Assignment Duration:** 2 days from receipt
- **Interview Notification:** Within 3-5 business days

Good luck! We're excited to see what you build! 🚀

Backend Intern Assignment - Organization Management Service

Objective

Build a backend service using any backend framework of your choice preferably a Python framework (Django/FastAPI) that supports creating and managing organizations in a multi-tenant style architecture. It is preferred to use MongoDB as a database.
The system should maintain a **Master Database** for global metadata and create **dynamic collections** for each organization.

Your task is to design and implement the required REST APIs and authentication flow.

Functional Requirements

1. Create Organization

Endpoint: `POST /org/create`
Input:

- `organization_name`
- `email` (admin email)
- `password` (admin password)

Expected Behavior:

- Validate that the organization name does not already exist.
- Dynamically create a **new Mongo collection** specifically for the organization.
  Example collection name pattern: `org_<organization_name>`.
- Create an **admin user** associated with that organization.
- Store the following in the **Master Database**:
  - Organization name
  - Organization collection name
  - Connection details (if required)
  - Admin user reference
- Return a success response with basic organization metadata.

---

2. Get Organization by Name

Endpoint: `GET /org/get`
Input:

- `organization_name`

Expected Behavior:

- Fetch and return the organization details stored in the Master Database.
- If the organization does not exist, return an appropriate error.

---

3. Update Organization

Endpoint: `PUT /org/update`
Input:

- `organization_name`
- `email` (admin email)
- `password` (admin password)

Expected Behavior:

- Validate that the organization name does not already exist.
- Dynamically handle the **new collection creation** specifically for the organization and sync the existing data to the new Table/Collection.

---

4. Delete Organization

Endpoint: `DELETE /org/delete`

Input:

- `organization_name`

Expected Behavior:

- Allow deletion for respective authenticated user only
- Handle deletion of the relevant collections of this organization

---

5. Admin Login

Endpoint: `POST /admin/login`
Input:

- `email`
- `password`

Expected Behavior:

- Validate the admin credentials.
- On success, return a **JWT token** containing:
    - Admin identification
    - Organization identifier/ID
- On failure, return an unauthorized error.

Technical Requirements

A. Master Database

Should store:

- Organization metadata
- Connection details for each dynamic database
- Admin user credentials (securely hashed)

B. Dynamic Collection Creation

When an organization is created:

- Programmatically create a new Mongo collection for that organization.
- The collection can be empty or initialized with a basic schema (optional but good to have).

C. Authentication

- Implement admin login using JWT.
- Passwords must be hashed (e.g., bcrypt).

Additional questions

Do you think this is a good architecture with a scalable design? What can be the trade-offs with the tech stack and design choices? Please feel free to explain briefly if you can design something better. We would love to hear that.

Submission Guidelines

Candidates should submit:

- GitHub repository link
- Modular and clean design - Preferably Class based
- Instructions to run the application in README.md
- A high level diagram of the project
- Brief notes explaining the design choices (optional, but appreciated)
