Full Stack Development Internship Assignment: "GigFlow" Platform

1. Project Overview

GigFlow is a mini-freelance marketplace platform. The goal is to build a system where **Clients** can post jobs (Gigs) and **Freelancers** can apply for them (Bids). This assignment tests your ability to handle complex database relationships, secure authentication, and state management.

- **Estimated Time:** 2 - 3 Days.
- **Submission:** GitHub Repository link + Hosted Link.

2. Technical Stack

- **Frontend:** React.js (Vite preferred) + Tailwind CSS.
- **Backend:** Node.js + Express.js.
- **Database:** MongoDB (via Mongoose).
- **State Management:** Redux Toolkit or Context API.
- **Authentication:** JWT (JSON Web Tokens) with HttpOnly cookies.

3. Core Features to Implement

A. User Authentication

- Secure Sign-up and Login.
- Roles are fluid: Any user can post a job (Client) or bid on a job (Freelancer).

B. Gig Management (CRUD)

- **Browse Gigs:** A public/private feed showing all "Open" jobs.
- **Search/Filter:** Users should be able to search for jobs by title.
- **Job Posting:** A form for logged-in users to post a job with Title, Description, and Budget.

C. The "Hiring" Logic (Crucial)

1. **Bidding:** A freelancer submits a "Bid" (message + price) on a gig.
2. **Review:** The Client who posted the job sees a list of all Bids.
3. **Hiring:** The Client clicks a **"Hire"** button on one specific Bid.
    - **Logic:** The Gig status must change from `open` to `assigned`.
    - **Logic:** The chosen Bid status becomes `hired`.
    - **Logic:** All other Bids for that same Gig should automatically be marked as `rejected`.

4. API Architecture

Category	**Method**	**Endpoint**	**Description**
Auth	POST	/api/auth/register	Register new user.
Auth	POST	/api/auth/login	Login & set HttpOnly Cookie.
Gigs	GET	/api/gigs	Fetch all open gigs (with search query).
Gigs	POST	/api/gigs	Create a new job post.
Bids	POST	/api/bids	Submit a bid for a gig.
Bids	GET	/api/bids/:gigId	Get all bids for a specific gig (Owner only).
Hiring	PATCH	/api/bids/:bidId/hire	The "Hire" logic (Atomic update).

5. Database Schema Hints

- **User:** name, email, password.
- **Gig:** title, description, budget, ownerId, status (open or assigned).
- **Bid:** gigId, freelancerId, message, status (pending, hired, rejected).

6. Bonus Questions (Hard)

Complete these to stand out from other candidates.

Bonus 1: Transactional Integrity (Race Conditions)

Implement the "Hire" logic using **MongoDB Transactions** or a highly secure logic flow. You must ensure that if two people (e.g., two admins of a project) click "Hire" on different freelancers at the exact same time, the system **only** allows one to be hired and prevents the other.

Bonus 2: Real-time Updates

Integrate **Socket.io**. When a Client hires a Freelancer, the Freelancer should receive an instant, real-time notification in their dashboard saying, *"You have been hired for [Project Name]!"* without having to refresh their page.

7. Submission Guidelines

1. **Code: GitHub Repository Link** (with complete source code) `README.md`.
2. **Environment:** Include a `.env.example` file so we know which keys to set up.
3. **Video/Demo:** A 2-minute Loom video walking through the "Hiring" flow.

8. Email Submission Details

To: ritik.yadav@servicehive.tech
CC: hiring@servicehive.tech