JOB PORTAL APPLICATION (MERN STACK)

1. High-Level System Architecture

The system is designed using the MERN Stack with modular backend components handling Users, Recruiters, Jobs, Job Applications, and File Uploads. Authentication is split between Clerk (users) and JWT (recruiters). MongoDB serves as the primary data store, and Cloudinary is used for media storage.

Core Components:

- Frontend (React + Clerk): User authentication, job search, filters, job detail page, applied jobs page.
- Backend (Node.js + Express): Handles user APIs, company/recruiter APIs, job creation & search, job applications, resume uploads, and webhooks.
- Database (MongoDB): Stores users, recruiters, jobs, applications.
- File Storage (Cloudinary): Resume and logo storage.

2. Database Design & Scalability

Collections:

- User: Clerk user data, resume URL.
- Company: Recruiter details, hashed password, logo URL.
- Job: Title, salary, location, skills, visibility.
- JobApplication: Mapping between users and jobs, application status.

Indexing:

- Jobs indexed by title and location for rapid filtering.
- Applications indexed by userId and companyId.
- Compound unique index prevents duplicate applications.

Scaling:

- MongoDB Replication for high availability.
- Sharding strategies:
- Jobs: shard by location.
- JobApplications: shard by userId for even write distribution.

3. Concurrency Control

- Prevent duplicate applications using compound unique indexes.
- Atomic updates ensure strong consistency when recruiters update statuses.
- Multi-document transactions (when needed) guarantee ACID guarantees.
- Optional message queue for high-load email or notification tasks.

4. Caching Strategy

- Cache frequently accessed job searches.
- Use Redis or in-memory caching (LRU cache).
- Cache invalidation when recruiter updates or creates jobs.

5. Optional Message Queue Usage

Event-driven architecture can be used for:

- Sending confirmation emails after applications.
- Notifying recruiters of new applicants.
- Processing resumes asynchronously.

Summary

This architecture ensures a scalable, maintainable, and efficient system with clean separation of user and recruiter responsibilities, optimized database access, strong consistency controls, and readiness for future scaling with caching and message queues.