**1. High-Level Architecture & Stack**

**Backend:**

* **Node.js + TypeScript + NestJS**: For structured, maintainable code with dependency injection.
* **MongoDB**: Flexible schema for property listings, brokers, and user data (supports nested documents like amenities, images).
* **Elasticsearch**: Powering "Smart Search" with:
  + **Full-text search** (property descriptions, titles)  
    **Geo-queries** ("near Burj Khalifa")  
    **Vector search** (for NLP-based semantic matching).  
    *(I previously implemented Elasticsearch at scale for* [*booksmm.com*](https://booksmm.com)*, optimizing 4 millions queries for international marketing providers to return results in <30ms under heavy load ,Elasticsearch also supports smart search.)*

**Frontend:**

* **Next.js (React) + TypeScript**: SSR for SEO, type safety.
* **Tailwind CSS**: Rapid UI development.
* **tRPC or REST**: Type-safe API calls.

**Why This Stack?**

* **Scalability**: Elasticsearch and MongoDB sharding handle 20k/mo listings + 50k concurrent users.
* **Performance**: Elasticsearch delivers sub-100ms responses for complex queries (proven at booksmm.com).
* **DevEx**: TypeScript end-to-end + NestJS modularity.

**2. Scalability & Performance**

**a) Handling 20k Listings/Month & 50k Concurrent Users**

* **Elasticsearch Cluster**: Deployed with 3+ nodes (hot/warm architecture) for "Smart Search."
* **MongoDB**: Read replicas + sharding by region (geo\_lat/geo\_lng).
* **Redis**: Caches frequent searches (e.g., "3-bedroom Dubai Marina").

**b) Large CSV Uploads**

* **Background Jobs**: Brokers upload CSVs → S3 → BullMQ queues → **MongoDB bulk inserts** (no blocking).
* **Validation**: Stream rows with fast-csv, reject malformed data (e.g., negative prices).

**c) Real-Time Notifications**

* **MongoDB Change Streams**: Trigger alerts when new properties match saved searches.
* **WebSockets**: Push notifications via Socket.io (scaled using Redis adapter).

**3. AI Integration & Innovation**

**a) "Smart Search" Technical Approach**

* **NLP Pipeline**:
  1. **Query Parsing**: Use OpenAI’s text-embedding-3-small to convert "3-bedroom near Burj Khalifa with pool" → vector.
  2. **Hybrid Search**:
     + **Elasticsearch**: Filters by price/bedrooms (structured data).
     + **Pinecone/Weaviate**: Vector similarity for semantic matches (e.g., "sea view" ≈ "oceanfront").
  3. **Ranking**: Combine vector similarity + business rules (e.g., boost premium listings).

**b) Bonus AI Feature: Automated Image Tagging**

* **Tech**: AWS Rekognition (or Google Vision) to tag property photos ("pool", "gym").
* **Value**: Improves search faceting + surfaces high-quality images first.