**Persistence Model**

The database we use is a NoSQL database management system. NoSQL (Not Only SQL) is designed to handle large volumes of unstructured and semi-structured data. [1]

The NoSQL definition refers to databases that support multiple models—including document, graph, key-value, wide-column, and vector stores.[ 1]

**Why NoSQL?**

NoSQL databases offer scalability, flexibility, and high performance for modern applications. They handle unstructured data, support horizontal scaling, and eliminate complex ORM layers. Ideal for AI, big data, and real-time analytics, NoSQL enables agility and faster development.[3]

Our system handles multiple user types jobseekers, employers, admins with different data structures. When using NoSQL we don’t need to stick to a rigid schema, which is ideal when attributes differ between users or evolve over time. NoSQL databases are designed for **horizontal scaling** As our platform grows we can add more servers easily to handle the load. It also Supports real-time updates and AI integration.

Use a NoSQL database when [1]:

* Data is unstructured or semi-structured and needs a flexible schema.
* We require high scalability and must handle large datasets.
* Performance is critical for real-time applications.
* The application needs to handle distributed workloads efficiently.

**MongoDB overview**

MongoDB is an open source [NoSQL](https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL) database management program. MongoDB is a tool that can manage document-oriented information, store or retrieve information. MongoDB is used for high-volume data storage, helping organizations store large amounts of data while still performing rapidly. Organizations also use MongoDB for its ad-hoc queries, indexing, [load balancing](https://www.techtarget.com/searchnetworking/definition/load-balancing), aggregation, server-side JavaScript execution and other features. MongoDB environments provide users with a server to create databases with MongoDB. MongoDB stores data as records that are made up of collections and documents.[2]

Even though mapping class diagram into relation and normalization is not needed when using NoSQL we normalized some tables to make it simpler and embedded as necessary.

Our original tables and tables were as follows:

|  |  |
| --- | --- |
| Tables | Attributes |
| Users | User\_ID, Name, Email, Password, Role, Created\_at, Updated\_at, Seeker\_ID, Employer\_ID, Admin\_ID |
| JobSeekers | Seeker\_ID, Resume, Skills, Experience, Education, Address, Profile\_Status, Profile\_Picture |
| Employers | Employer\_ID, Company\_Name, Industry, Address, Company\_Description, Contact\_info, Company\_logo |
| Jobs | Job\_ID, Employer\_ID, Title, Description, Category, Job\_type, Salary\_Range, Posted\_at, Status, Location |
| Applications | App\_ID, Job\_ID, Seeker\_ID, Status, Applied\_at, Updated\_at |
| Interviews | Interview\_ID, App\_ID, Scheduled\_at ,Status, feedback |
| Notification | Not\_ID, User\_ID, Message, Status, Sent\_at |
| SavedJobs | Saved\_ID, Seeker\_ID, Job\_ID, Saved\_at |
| Shortlisted Candidates | Shortlisted\_ID, Employer\_ID, Seeker\_ID, Job\_ID, Status |
| Admin\_logs | Log\_ID, Admin\_ID, Action, Timestamp |
| Recommendation History | Recommendation\_ID, Seeker\_ID, Job\_ID, Score, Recommended\_at |

Table : Database Tables

Our final tables with embedding and normalizing:

|  |  |  |  |
| --- | --- | --- | --- |
| Collection Name | Type | Attributes | Description |
| User | Root table (with Embeddings) | User\_ID, fullName, email, Phone\_Number, Profile\_Picture, role, Jobseeker\_profile (embedded), notifications (embedded), SavedJobs (embedded), administrator (embedded if admin) | Main user document. Embeds Jobseeker profile, notifications, and saved jobs based on user role. |
| JobSeeker (embedded in User) | Embedded | resume, jobPreferences, savedJobs | Jobseeker-specific info stored inside the User document if role = jobseeker. |
| Notifications (embedded in User) | Embedded | Notification\_ID, message, status, sent\_at | Notifications for a specific user. Embedded if count is low or mostly user-specific. |
| SavedJobs (embedded in JobSeeker) | Embedded | job\_id, saved\_at | Saved job references inside a jobseeker profile. |
| Administrator (embedded in User) | Embedded | adminID, registeredDate | Administrator role and metadata, embedded in the user if role = admin. |
| Employers | Normalized | employerID, companyName, industry, address, company\_description, contact\_info, company\_logo | Standalone collection for employer details. |
| Jobs | Normalized | Job\_ID, title, description, category, location, job\_Type, salary, posted\_By (employer\_ID), shortlisted\_candidates (optional embedded) | Job listings posted by employers. May embed shortlist if small. |
| Shortlisted\_Candidates (embedded in Jobs) | Embedded | seeker\_id, status | Embedded inside each job document if the number is small. |
| Applications | Normalized | Application\_ID, job\_ID, jobSeeker\_ID, status, applied\_at, updated\_at | Links jobseekers to jobs. Stored independently to support queries and updates. |
| Interviews | Normalized | interviewID, jobID, jobSeekerID, employerID, date, status | Each interview links multiple entities. Kept separate for tracking and scheduling. |
| Recommendation\_History | Normalized | recommendationID, userID, recommendedJobs (jobID list), score, recommended\_at | AI-generated job recommendations. Stored separately for analysis and AI training. |
| Admin\_Log | Normalized | Log\_ID, admin\_ID, action, timestamp | Tracks all admin activities system-wide for auditing purposes. |

Table: Normalized Database Tables

References

1. <https://www.geeksforgeeks.org/introduction-to-nosql/>
2. <https://www.techtarget.com/searchdatamanagement/definition/MongoDB>
3. <https://www.couchbase.com/resources/why-nosql/>