**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 27th JUNE 2025 |
| Team ID | LTVIP2025TMID58557 |
| Project Name | Freelance Finder – Freelance Hiring Platform |

**Project Title**: Freelance Finder – Freelance Hiring Platform

**Date**: 27th JUNE 2025

**Objective**

The objective of this report is to outline the database design and implementation details for the Freelance Finder project. This includes defining key entities, schema structure, and integration of MongoDB using Mongoose for effective data management and secure operations.

**Technologies Used**

* **Database Management System (DBMS):** MongoDB
* **Object-Document Mapper (ODM):** Mongoose

**Design the Database Schema**

The database schema includes key collections to support user management, job listings, applications, and chat functionality.

**1. Users**

* Attributes:
  + \_id: ObjectId
  + name: String
  + email: String (unique)
  + password: String (hashed)
  + role: String ("freelancer" or "client")
  + phoneNumber: String
  + skills: [String]
  + location: String
  + experience: Number
  + profileImage: String (Cloudinary URL)
  + resume: String (Cloudinary URL)
  + createdAt: Date
  + updatedAt: Date

**2. Jobs**

* Attributes:
  + \_id: ObjectId
  + title: String
  + description: String
  + category: String
  + location: String
  + skillsRequired: [String]
  + postedBy: ObjectId (references User)
  + createdAt: Date
  + updatedAt: Date

**3. Applications**

* Attributes:
  + \_id: ObjectId
  + jobId: ObjectId (references Job)
  + applicantId: ObjectId (references User)
  + resumeLink: String
  + appliedAt: Date

**4. Messages (Chat System)**

* Attributes:
  + \_id: ObjectId
  + senderId: ObjectId (references User)
  + receiverId: ObjectId (references User)
  + messageText: String
  + timestamp: Date

**Implement the Database using MongoDB**

**Database Name**: freelance\_finder

**Collections and Example Schemas**:

// users

{

\_id: ObjectId,

name: "John Doe",

email: "john@example.com",

role: "freelancer",

skills: ["React", "Node.js"],

location: "Bangalore",

profileImage: "https://cloudinary.com/...",

resume: "https://cloudinary.com/...",

createdAt: ISODate,

updatedAt: ISODate

}

// jobs

{

\_id: ObjectId,

title: "Frontend Developer Needed",

description: "Build a responsive UI...",

category: "Web Development",

postedBy: ObjectId,

createdAt: ISODate

}

// applications

{

\_id: ObjectId,

jobId: ObjectId,

applicantId: ObjectId,

resumeLink: "https://cloudinary.com/...",

appliedAt: ISODate

}

// messages

{

\_id: ObjectId,

senderId: ObjectId,

receiverId: ObjectId,

messageText: "Hello, I’m interested in your project!",

timestamp: ISODate

}

**Integration with Backend**

* Database connection:



The **Express.js backend** interacts with MongoDB via Mongoose, allowing secure and structured operations:

* **User APIs**: Create, read, update user profiles
* **Job APIs**: CRUD operations for job posts, with filtering options
* **Application APIs**: Save job applications and view applicants
* **Chat APIs**: Send and retrieve user-to-user messages
* **Upload APIs**: Resume and image files are stored using Cloudinary and URLs are saved in MongoDB