Full Stack Development with MERN

Database Design and Development Report

Date	11-07-2024
Team ID	SWTID1719923176
Project Name	Freelancing Finder
Maximum Marks	

Project Title: Freelancing Application MERN

Date: 11-07-2024

Prepared by: Aryaman Parashar Behera,SWARNA KAMALAM S,Disha M,P R Adithya

Objective

The objective of this report is to outline the database design and implementation details for the Freelancing Application MERN project, including schema design and database management system (DBMS) integration.

Technologies Used

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

Design the Database Schema

The database schema is designed to accommodate the following entities and relationships:

1. Users

Attribute list:[firstname ,lastname ,email ,password , account ,type ,contact ,location ,profiledUrl cvUrl , jobTitle ,about]

2. jobs

Attribute list:[company,jobtitle,jobtype,location,salary,vacancies,experience,detail,application]

3. Company

Attribute list :[name,email,password,contact,location,about,profileUrl,jobposts]

Implement the Database using MongoDB

The MongoDB database is implemented with the following collections and structures:

Database Name:jobfinder

1. Collection: users

```
 firstName: {  type: String,
 required: [true, "First Name is Required!"],  },
lastName: {  type: String,
 required: [true, "Last Name is Required!"],  },
email: {      type: String,
 required: [true, " Email is Required!"],
 unique: true,
 validate: validator.isEmail,  },
password: {      type: String,
 required: [true, "Password is Required!"],
 minlength: [6, "Password length should be greater than 6 character"],
 select: true,  },
accountType: { type: String, default: "seeker" },
contact: { type: String },
location: { type: String },
profileUrl: { type: String },
cvUrl: { type: String },
jobTitle: { type: String },
about: { type: String }, },
```

2. Collection: posts

```
company: { type: Schema.Types.ObjectId, ref: "Companies" },
jobTitle: { type: String, required: [true, "Job Title is required"] },
jobType: { type: String, required: [true, "Job Type is required"] },
location: { type: String, required: [true, "Location is required"] },
salary: { type: Number, required: [true, "Salary is required"] },
vacancies: { type: Number },
```

experience: { type: Number, default: 0 },

detail: [{ desc: { type: String }, requirements: { type: String } }],

application: [{ type: Schema.Types.ObjectId, ref: "Users"

3. Collection: comments

name: {

type: String,

required: [true, "Company Name is required"], },

email: {type: String,

required: [true, "Email is required"],

unique: true,

validate: validator.isEmail, },

password: { type: String,

required: [true, "Password is required"],

minlength: [6, "Password must be at least"],

select: true, },

contact: { type: String },

location: { type: String },

about: { type: String },

profileUrl: { type: String },

jobPosts: [{ type: Schema.Types.ObjectId, ref: "Jobs" }

Integration with Backend

- Database connection: Give Screenshot of Database connection done using Mongoose

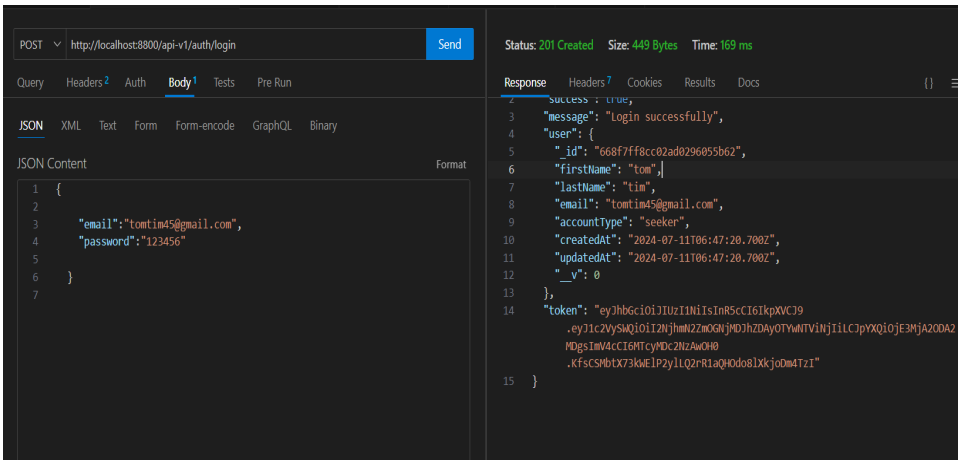
server > dbConfig > JS dbConnection.js > dbConnection

```
1  import mongoose from "mongoose";
2
3  const dbConnection = async () => {
4    try {
5      const dbConnection = await mongoose.connect(process.env.MONGODB_URL);
6
7      console.log("DB Connected Successfully");
8    } catch (error) {
9      console.log("DB Error: " + error);
10   }
11 };
12
13 export default dbConnection;
14
```

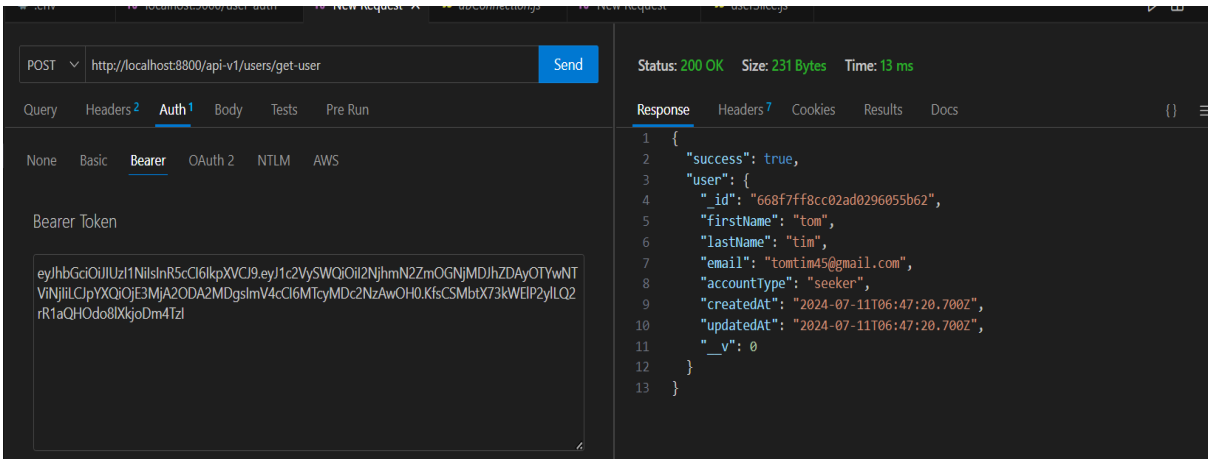
<u>The backend APIs interact with MongoDB using Mongoose ODM Key interactions include</u>:

User Management:

User login

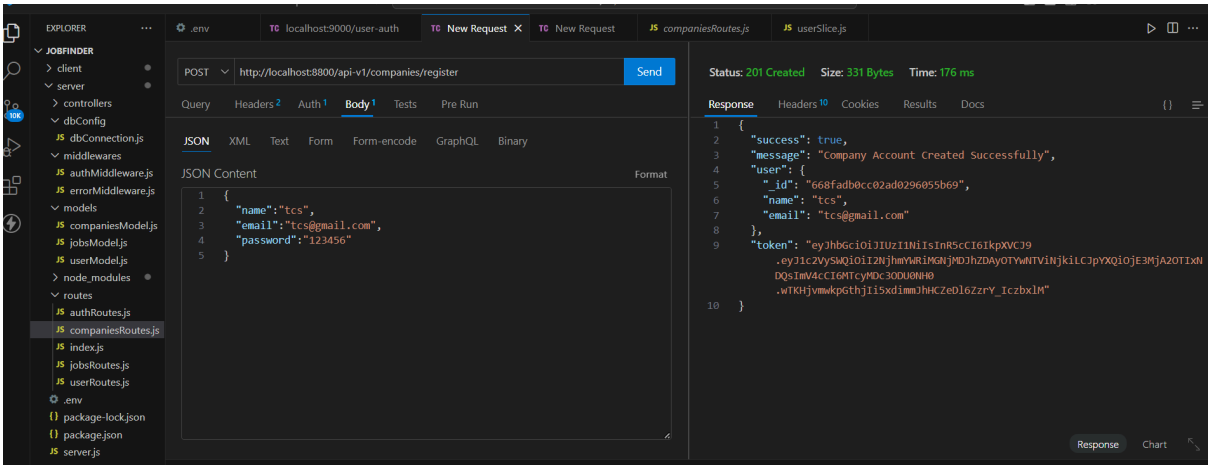


Get user

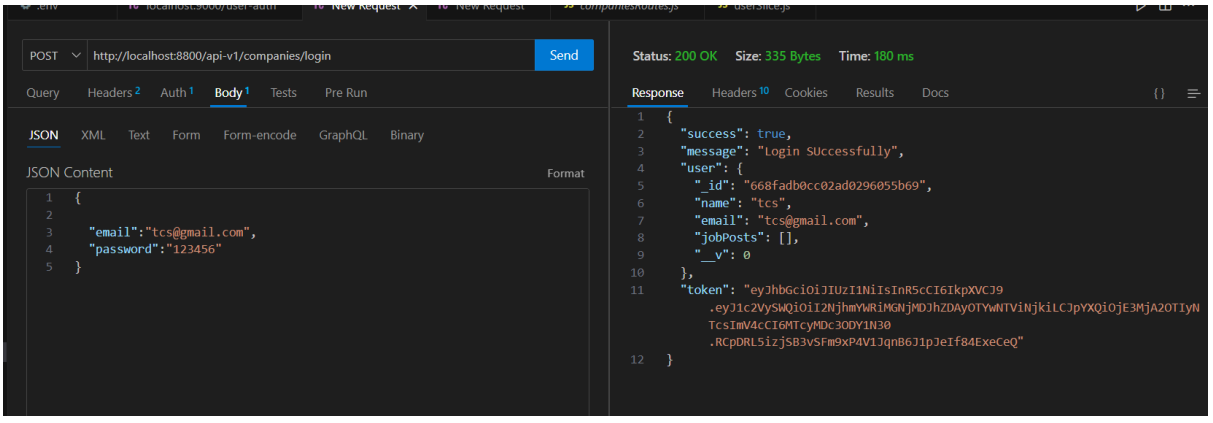


Company collection:

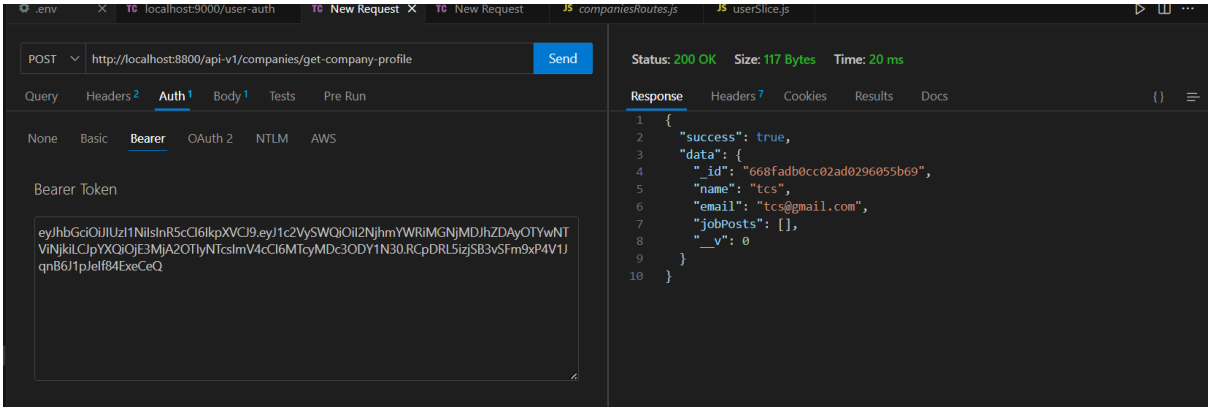
Company creation:



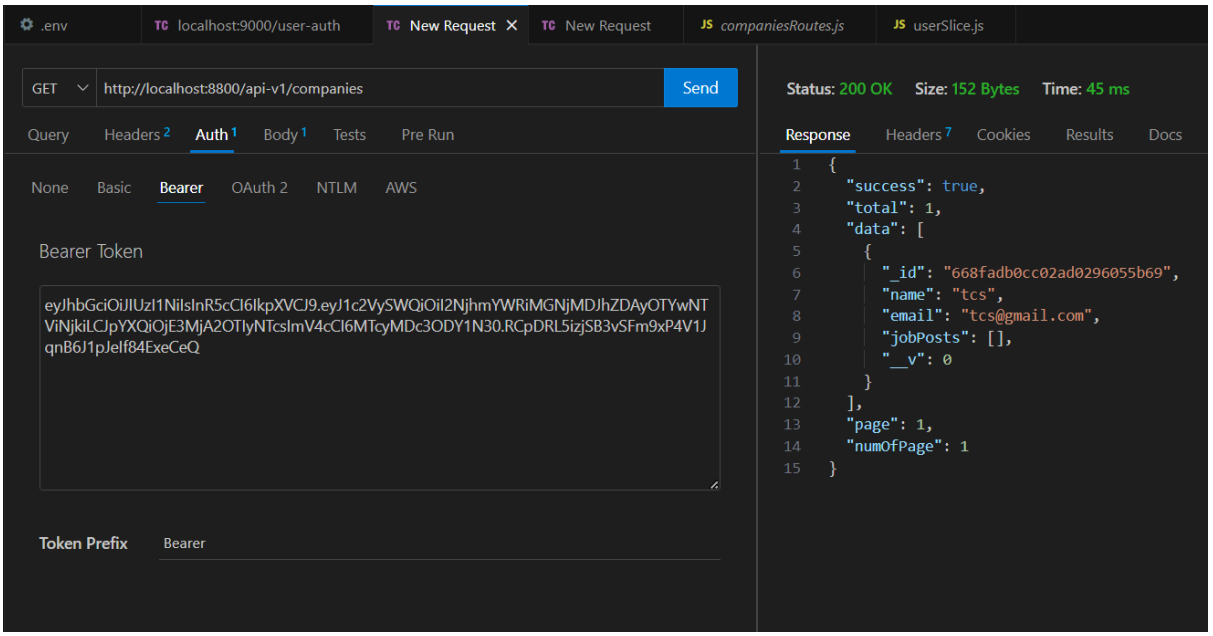
Company login



Fetching company

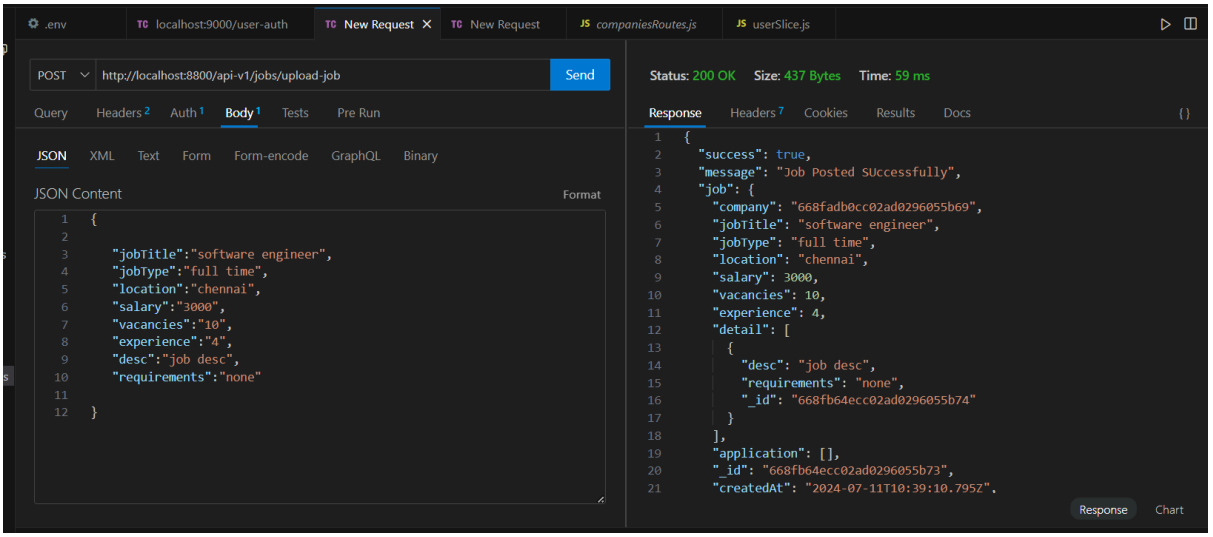


Getting total number of companies

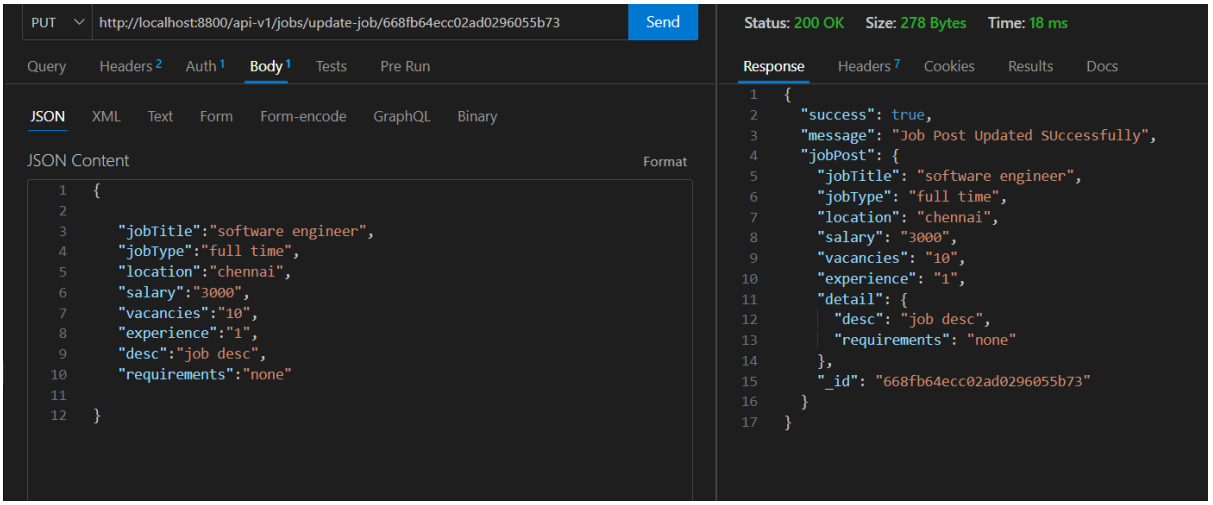


Jobs collection

Upload job



Update job



Find jobs