

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
```

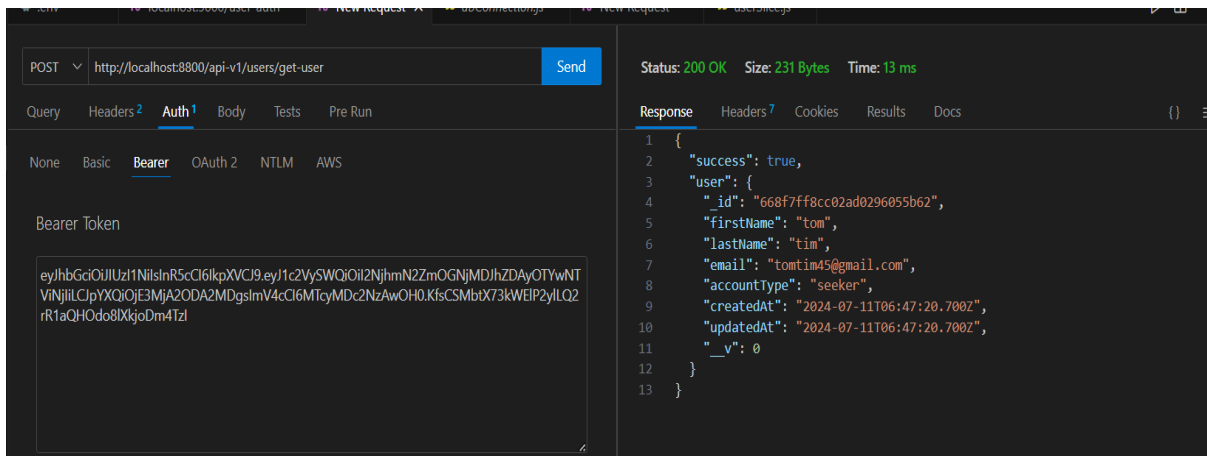
The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:

User Management:

User login

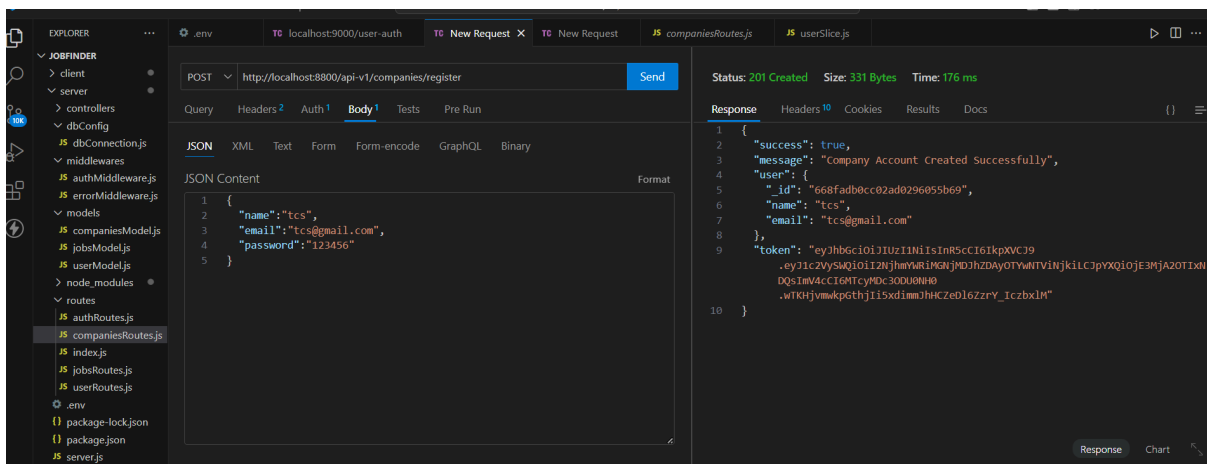
The screenshot shows the Swagger UI for the login endpoint. The request is a POST to `http://localhost:8000/api-v1/auth/login`. The request body is a JSON object with the following fields: `email` (tontims5@gmail.com), `password` (123456). The response is a 201 Created status with a JSON body containing: `message` (Login successfully), `user` (an object with id, firstName, lastName, email, and accountType), `createdAt`, `updatedAt`, and `token` (a long alphanumeric string).

Get user

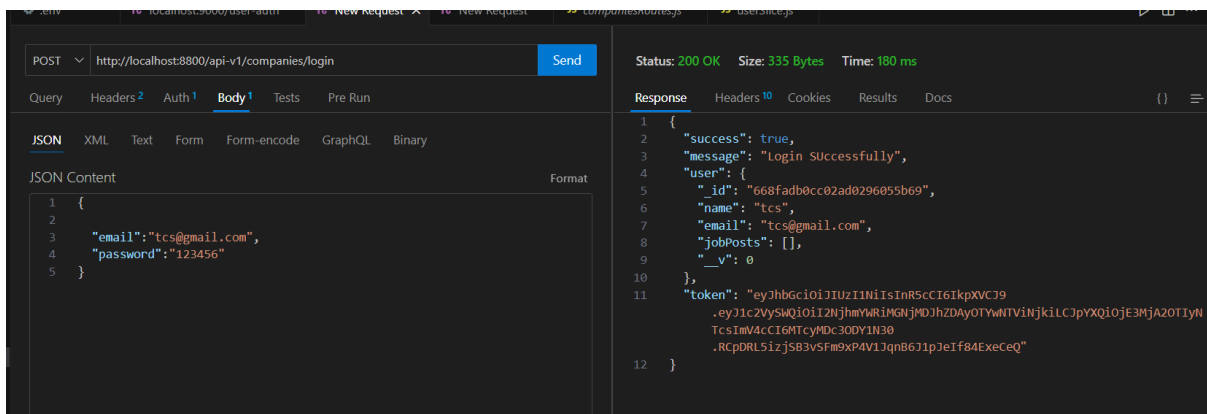


Company collection:

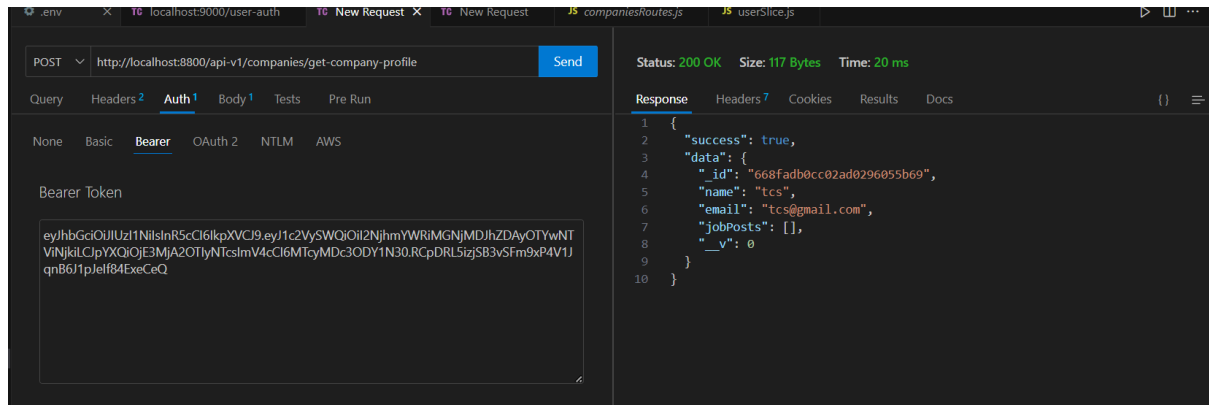
Company creation:



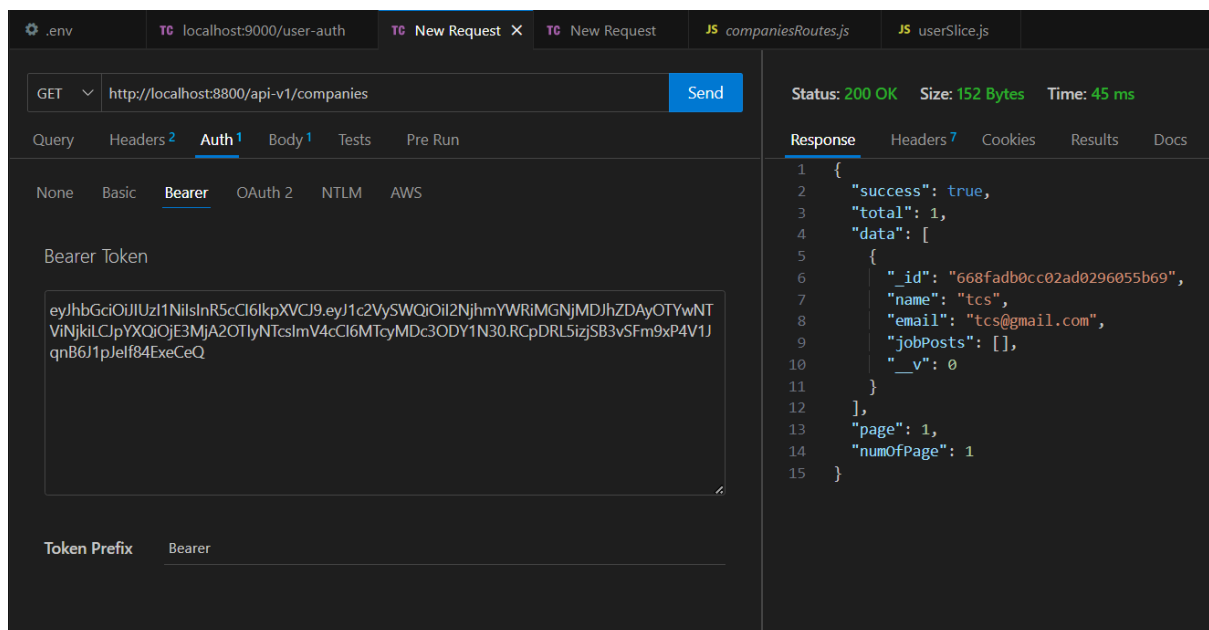
Company login



Fetching company



Getting total number of companies



Jobs collection

Upload job

.env localhost:9000/user-auth TC New Request X TC New Request JS companiesRoutes.js JS userSlice.js

POST http://localhost:8800/api-v1/jobs/upload-job Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2
3   "jobTitle": "software engineer",
4   "jobType": "full time",
5   "location": "chennai",
6   "salary": "3000",
7   "vacancies": "10",
8   "experience": "4",
9   "desc": "job desc",
10  "requirements": "none"
11 }
12 }
```

Status: 200 OK Size: 437 Bytes Time: 59 ms

Response Headers 7 Cookies Results Docs ()

```
1 {
2   "success": true,
3   "message": "Job Posted Successfully",
4   "job": {
5     "company": "668fadb0cc02ad0296055b69",
6     "jobTitle": "software engineer",
7     "jobType": "full time",
8     "location": "chennai",
9     "salary": "3000",
10    "vacancies": "10",
11    "experience": "4",
12    "detail": [
13      {
14        "desc": "job desc",
15        "requirements": "none",
16        "_id": "668fb64ecc02ad0296055b74"
17      }
18    ],
19    "application": [],
20    "_id": "668fb64ecc02ad0296055b73",
21    "createdAt": "2024-07-11T10:39:10.795Z",
22  }
```

Response Chart

Update job

PUT http://localhost:8800/api-v1/jobs/update-job/668fb64ecc02ad0296055b73 Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2
3   "jobTitle": "software engineer",
4   "jobType": "full time",
5   "location": "chennai",
6   "salary": "3000",
7   "vacancies": "10",
8   "experience": "1",
9   "desc": "job desc",
10  "requirements": "none"
11 }
12 }
```

Status: 200 OK Size: 278 Bytes Time: 18 ms

Response Headers 7 Cookies Results Docs

```
1 {
2   "success": true,
3   "message": "Job Post Updated Successfully",
4   "jobPost": {
5     "jobTitle": "software engineer",
6     "jobType": "full time",
7     "location": "chennai",
8     "salary": "3000",
9     "vacancies": "10",
10    "experience": "1",
11    "detail": {
12      "desc": "job desc",
13      "requirements": "none"
14    },
15    "_id": "668fb64ecc02ad0296055b73"
16  }
17 }
```

Find jobs

GET http://localhost:8800/api-v1/jobs/find-jobs Send

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2
3   "jobTitle": "software engineer",
4   "jobType": "full time",
5   "location": "chennai",
6   "salary": "3000",
7   "vacancies": "10",
8   "experience": "1",
9   "desc": "job desc",
10  "requirements": "none"
11 }
12 }
```

Status: 200 OK Size: 786 Bytes Time: 47 ms

Response Headers 7 Cookies Results Docs

```
1 {
2   "success": true,
3   "totalJobs": 2,
4   "data": [
5     {
6       "application": [],
7       "_id": "668f74e85aaa253c919015d5",
8       "jobTitle": "manager",
9       "jobType": "work-from-home",
10      "location": "remote",
11      "salary": "44000",
12      "vacancies": 5,
13      "experience": 2,
14      "detail": [
15        {
16          "_id": "668fba19cc02ad0296055b7d",
17          "desc": "it manager",
18          "requirements": "nil"
19        }
20      ],
21    },
22    {
23      "_id": "668fb64ecc02ad0296055b73",
24      "company": {
25        "_id": "668fadb0cc02ad0296055b69",
26        "name": "tcs",
27        "email": "tcs@gmail.com",
28        "jobPosts": [
29          "668fb64ecc02ad0296055b73"
30        ],
31        "v": 0
32      },
33      "jobTitle": "software engineer",
34      "jobType": "full time",
35      "location": "chennai",
36      "salary": "3000",
37    }
38  ]
39 }
```

