# Full Stack Development with MERN

# Database Design and Development Report

| | |
|---|---|
| Date | 18th July 2024 |
| Team ID | SWTID1719938419 |
| Project Name | Social Media App (MERN) |
| Maximum Marks | |

**Project Title**: Social Media Application

**Date**: 18th July 2024

**Prepared by**: Priyan

**Objective**

The objective of this report is to outline the database design and implementation details for the Social Media Application 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**

- Attributes: _id,fullName, username, email, followers, following, profileImg, coverImg

**2. Posts**

- Attributes: user: userId, text, img

**3. Comments**

- Attributes: _id, text, post, author, createdAt, updatedAt

**Implement the Database using MongoDB**

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

Database Name: [your_database_name]

**1. Collection: Notifications**

```
import mongoose from "mongoose";

const notificationSchema = new mongoose.Schema(
  {
    from: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    to: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    type: {
      type: String,
      required: true,
      enum: ["follow", "like"],
    },
    read: {
      type: Boolean,
      default: false,
```

```
    },
  },
  { timestamps: true }
);

const Notification = mongoose.model("Notification", notificationSchema);

export default Notification;
```

**2. Collection: posts**

```
 import mongoose from "mongoose";

const postSchema = new mongoose.Schema(
  {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    text: {
      type: String,
    },
    img: {
      type: String,
    },
    likes: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
```

```
        },
      ],
      comments: [
        {
          text: {
            type: String,
            required: true,
          },
          user: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "User",
            required: true,
          },
        },
      ],
    },
    { timestamps: true }
);

const Post = mongoose.model("Post", postSchema);

export default Post;
```

**3. Collection: Users**

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
  {
    username: {
```

```
    type: String,
    required: true,
    unique: true,
},
fullName: {
    type: String,
    required: true,
},
password: {
    type: String,
    required: true,
    minLength: 6,
},
email: {
    type: String,
    required: true,
    unique: true,
},
followers: [
    {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
        default: [],
    },
],
following: [
    {
        type: mongoose.Schema.Types.ObjectId,
        ref: "User",
```

```
            default: [],
        },
    ],
    profileImg: {
        type: String,
        default: "",
    },
    coverImg: {
        type: String,
        default: "",
    },
    bio: {
        type: String,
        default: "",
    },

    link: {
        type: String,
        default: "",
    },
    likedPosts: [
        {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Post",
            default: [],
        },
    ],
},
{ timestamps: true }
```

```
);

const User = mongoose.model("User", userSchema);

export default User;
```

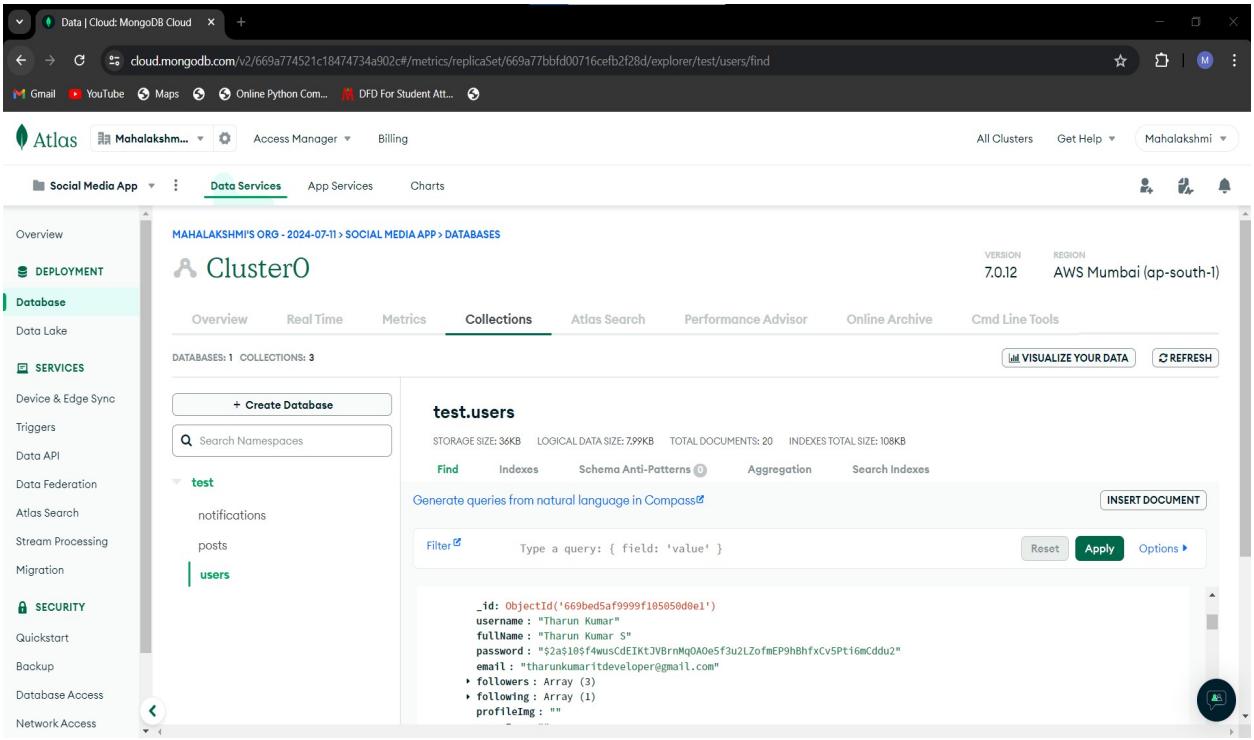
**Integration with Backend**

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

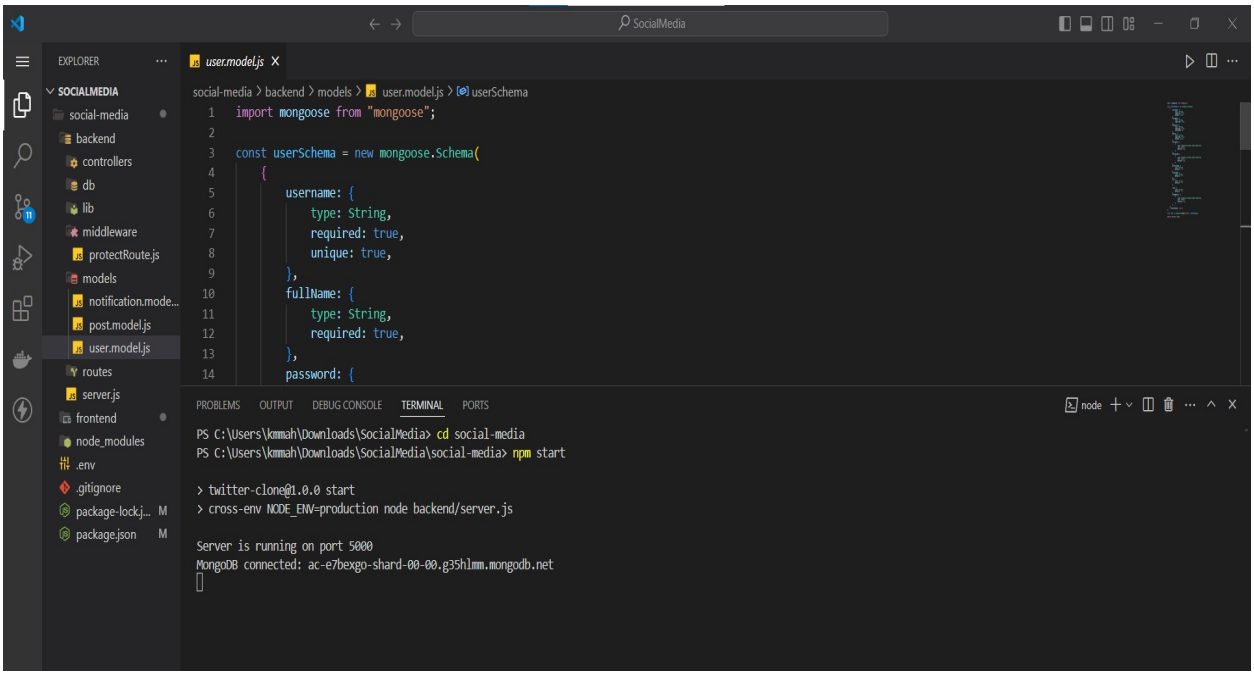
**Code:**

```
import mongoose from "mongoose";

const connectMongoDB = async () => {
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI);
        console.log(`MongoDB connected: ${conn.connection.host}`);
    } catch (error) {
        console.error(`Error connection to mongoDB: ${error.message}`);
        process.exit(1);
    }
};

export default connectMongoDB;
```

## atlas MongoDB :




## Connection:



- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
    - User Management: CRUD operations for users.
    - Post Management: CRUD operations for posts, with user authentication.
    - Comment Management: CRUD operations for comments associated with posts.
    - Notification Management: CRUD operations for comments associated with notification.