**Social Media App Documentation:**

**Project Part 3:**

In this project iteration we have developed various functionalities such as post management, real-time messaging, and community engagement through channels. The backend leverages Node.js, Express, MongoDB, and other relevant technologies to ensure a robust and scalable infrastructure.

**Technology Stack:**

Backend Framework: Node.js, Express.js

Database: MongoDB

Authentication: JWT, bcrypt

Real-time Communication: Socket.io

File Handling: Multer

Validation: Yup

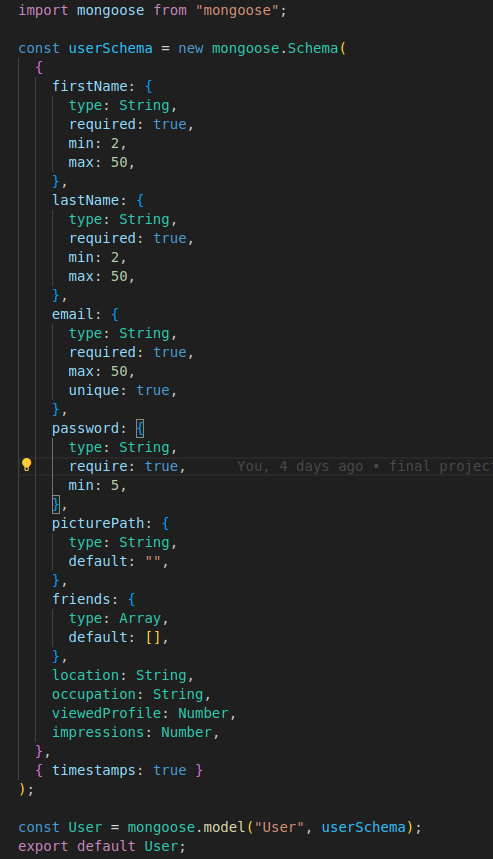
**PROGRESS REPORT:**

**1. MongoDB Modals Documentation:**

The models folder contains the Mongoose schema definitions for the MongoDB collections used in the application:

1. **User Model Schema:**

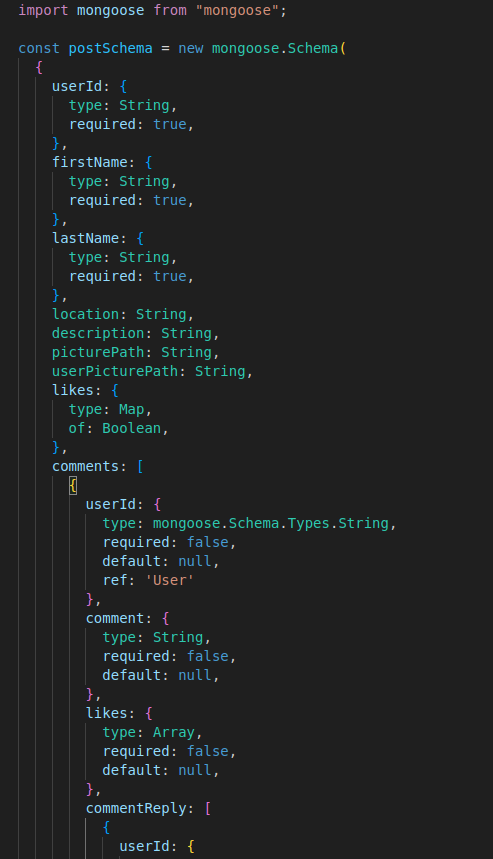
This section describes the User model schema used in the application, which is implemented using Mongoose. The schema defines the structure of the user documents stored in the MongoDB collection.



Screenshot of User Schema:

**B. Post Model Schema:**

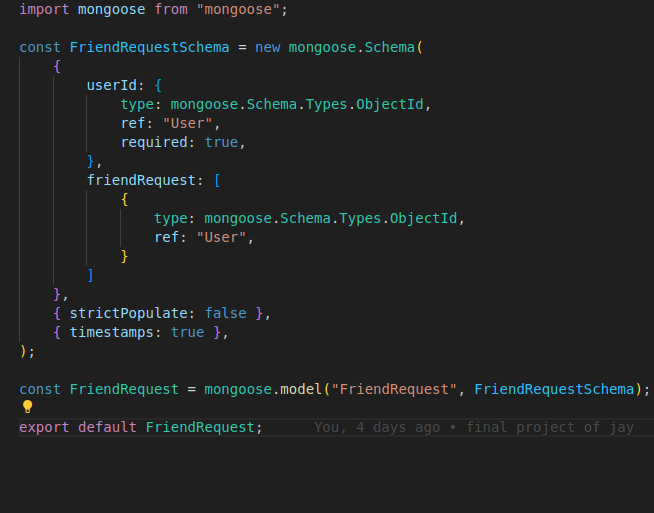
This section describes the Post model schema used in the application, which is implemented using Mongoose. The schema defines the structure of the post documents stored in the MongoDB collection.



Screenshot of Post modal schema:

**C. FriendRequest Model Schema:**

This section describes the FriendRequest model schema used in the application, which is implemented using Mongoose. The schema defines the structure of the friend request documents stored in the MongoDB collection.



Screenshot of FriendRequest Schema:

**D. Communication Model Schema:**

The Communication model schema defines the structure of communication documents stored in the MongoDB collection. This schema handles the details of communications between users, including members, messages, and metadata.

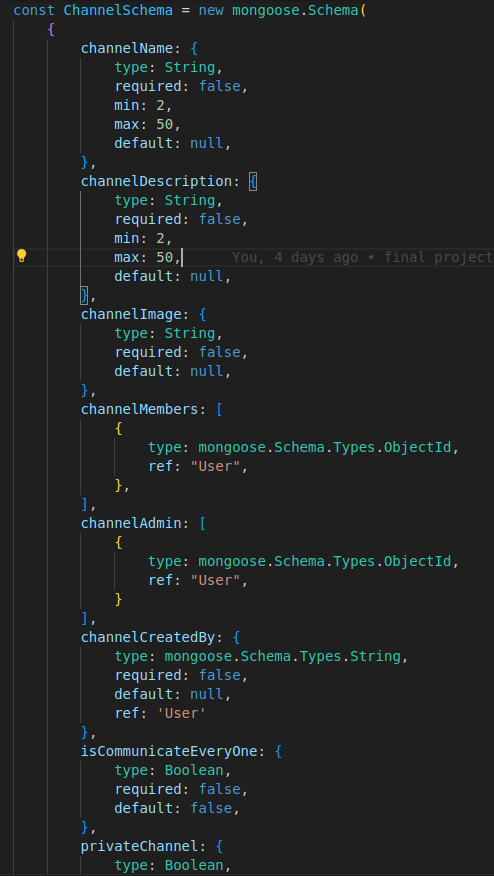
Logging and Parsing:



Screenshot of Communication Schema:

1. **Channel Model Schema:**

The Channel model schema defines the structure of channel documents stored in the MongoDB collection. This schema manages the details of channels within the application, including members, administrators, messages, and metadata.



Screenshot of Channel Schema:

1. **Posts Routes Documentation:**

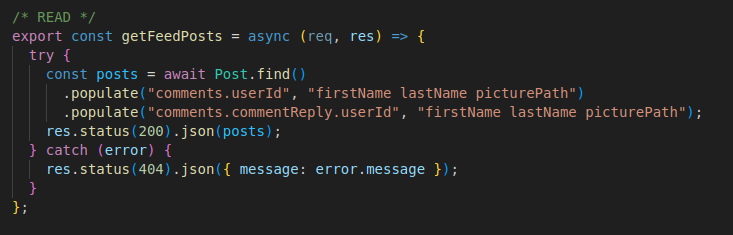
These routes used for handling posts in the application, including retrieving, updating, and managing comments and replies on posts. Each route has specific functionality and is secured with token verification to ensure that only authenticated users can access these endpoints.



Posts Routes file:

1. **Posts Routes Documentation:**

This endpoint retrieves all posts from the database, along with details of comments and comment replies associated with each post. It returns a JSON array of posts, where each post object includes information about the post itself and its related comments and replies.



Get Posts Routes file:

**Postman Screenshot of Get Posts API:**

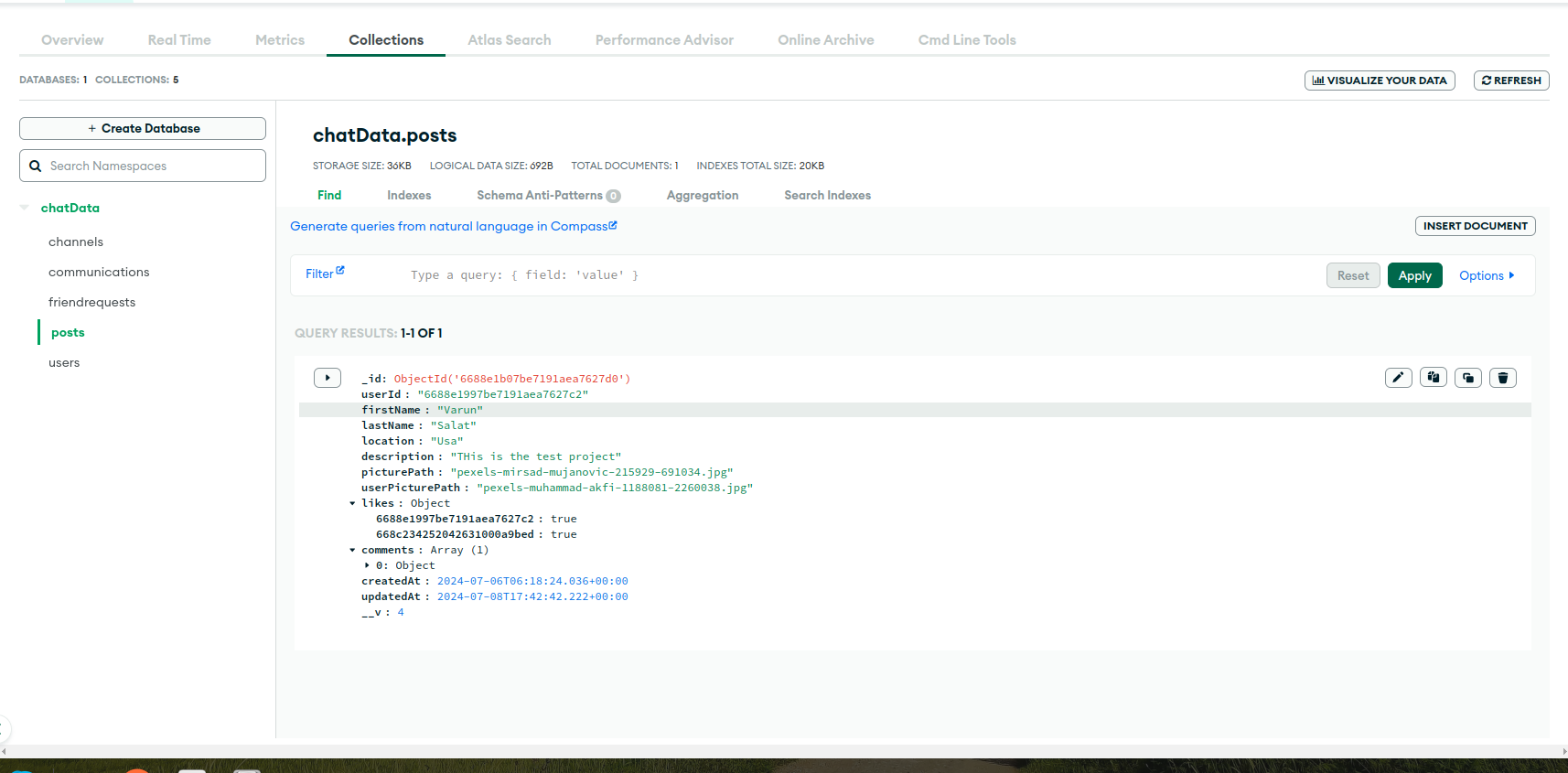
This Postman request retrieves a feed of posts from the server. Each post includes details such as user information, comments, and likes. The endpoint is protected and requires a valid JWT token in the Authorization header to authorize access.

**HTTP Method and URL:**

The URL is set to the API endpoint:

http://localhost:8000/posts/

**MongoDB Screenshot of Get Posts API:**



Posts Modal MongoDB: