**EXPERIMENT NO:**

**AIM**

REST API Design with MongoDB + Mongoose Integration.

**THEORY**

**1. What is a REST API?**

● REST (Representational State Transfer) is an architectural style for building web services.

● A **REST API** uses HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources.

● Each resource is represented by a **URL (endpoint)**, e.g. /api/users.

**Example mappings:**

● GET /api/users → Fetch all users

● POST /api/users → Add new user

● PUT /api/users/:id → Update user by ID

● DELETE /api/users/:id → Delete user by ID

**Why MongoDB?**

● **MongoDB** is a NoSQL database that stores data in flexible **JSON-like documents**. ● Useful for modern applications where schema can change dynamically. ● Example MongoDB document:

**3. Why Mongoose?**

● **Mongoose** is an ODM library for MongoDB and Node.js.

● It allows developers to:

○ Define **schemas** for data.

○ Create **models** to interact with MongoDB collections.

○ Perform CRUD operations easily.

**4. REST API Workflow with MongoDB + Mongoose**

1. **Client** sends requests (e.g., GET /api/users).

2. **Express server** receives request.

3. **Route handler** calls a **Mongoose model** to interact with MongoDB.

4. **The database** responds with data.

5. **Express** sends the response back to the client in JSON format.

**5. Advantages of this Integration**

● **Scalability**: MongoDB handles large datasets.

● **Flexibility**: JSON-like structure matches REST responses.

● **Productivity**: Mongoose simplifies queries with built-in methods. ● **Separation of Concerns**: REST API design keeps client and server independent.

**STEPS**

1) Create project and install deps

mkdir rest-experiment

cd rest-experiment

npm init -y

npm i express mongoose dotenv morgan express-async-errors

npm i -D nodemon

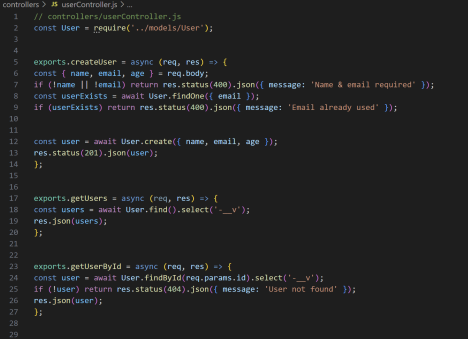
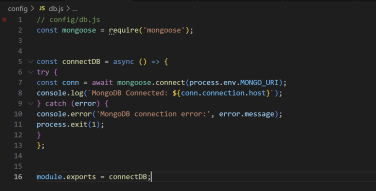
2) Create .env file

MONGO\_URI=your\_mongodb\_uri\_here

PORT=5000

NODE\_ENV=development

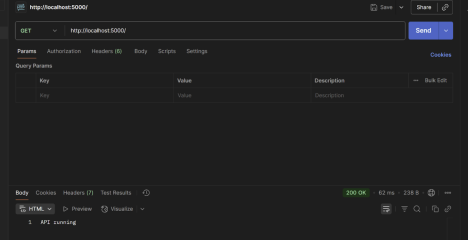
**SOURCE CODE**

****

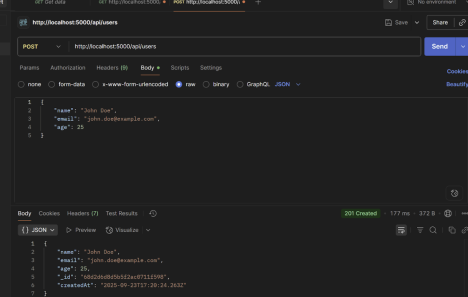
****

**OUTPUT**

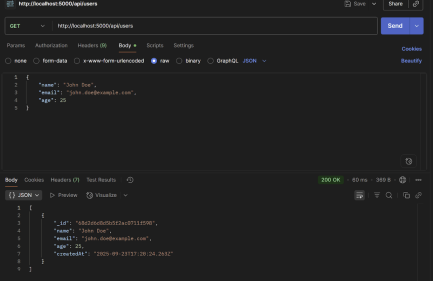
Request 1: Check if server is alive



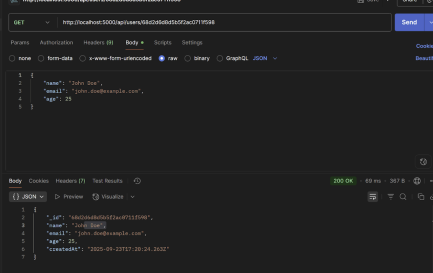
Request 2: Create a new user



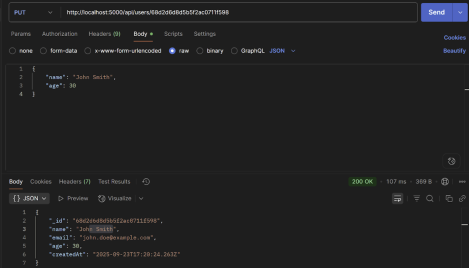
Request 3: Get all users



Request 4 : Get user by ID



Request 5: Update user



**CONCLUSION**

This experiment demonstrated how to design a RESTful API using **Node.js, Express, MongoDB, and Mongoose**. It showed how CRUD operations can be performed efficiently with well-structured endpoints and how Mongoose simplifies interaction with MongoDB. The integration provides a scalable and flexible way to build modern backend applications.