

User Management API with MongoDB Integration

This project builds on the initial User Management API by adding MongoDB as a database, providing persistent storage for user data. Mongoose is used to manage the connection, define schemas, and validate data.

Key Aspects:

- ❖ **MongoDB Connection:** Connects Node.js to MongoDB through Mongoose, enabling persistent data storage.
- ❖ **User Schema:** Defines a structured schema with Mongoose, setting required fields and data types to enforce data integrity.
- ❖ **CRUD Routes:**
 - **GET /users** – Fetches all users from MongoDB.
 - **GET /users/**
 - Retrieves a user by their MongoDB ObjectId.
 - **POST /user** – Adds a new user and saves it to the database.
 - **PUT /user/**
 - Updates a user's information.
 - **DELETE /user/**
 - Deletes a user by ObjectId.

Challenges:

- **Error Handling:** Requires careful async error handling for database interactions.
- **ObjectId Usage:** Manages MongoDB's ObjectId format, different from simple integer IDs.
- **Middleware Enhancements:** Handles more complex request logging and input validation.

This transition also involves advanced middleware to manage requests and respond with relevant status codes. The additional complexity of handling asynchronous database operations adds a level of difficulty, making it important to implement proper async/await handling and error-catching mechanisms.

GitHub Link:

<https://github.com/Ashmita2282/userAPI-MongodbMngt>

Project Structure

```
UserAPIManagement/  
├── controller/  
│   └── userController.js  
├── middleware/  
│   └── logger.js  
├── routes/  
│   └── userRoutes.js  
├── config/  
│   └── database.js  
├── models/  
│   └── userModel.js  
├── nodes/  
│   └── various file  
├── .env  
├── index.js  
└── package.json
```

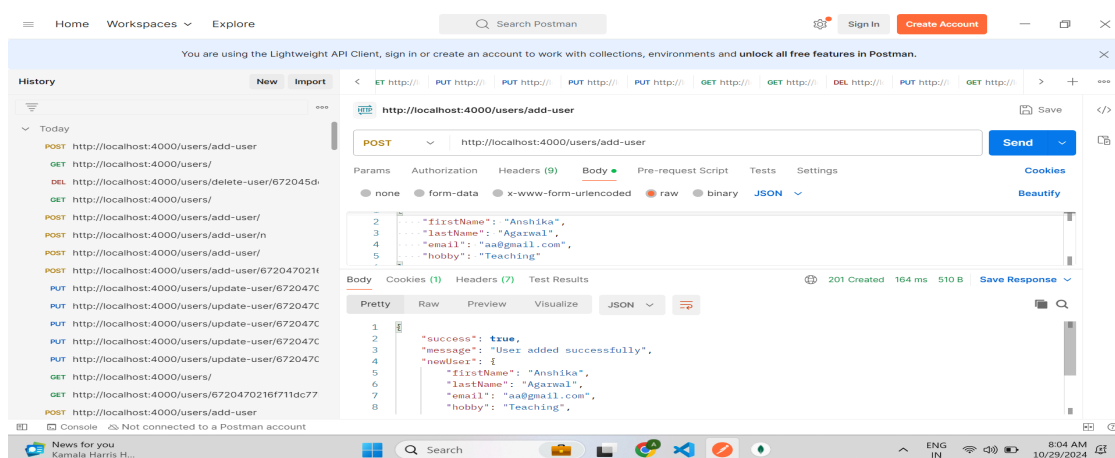
ScreenShots →

Having screenshot of PostMan Testing and also For MongoDB Compass for all the API's whether it is adding user → POST, Updating User → PUT, Deleting User → DELETE, fetch all user details → GET, fetch only one user by id → GET

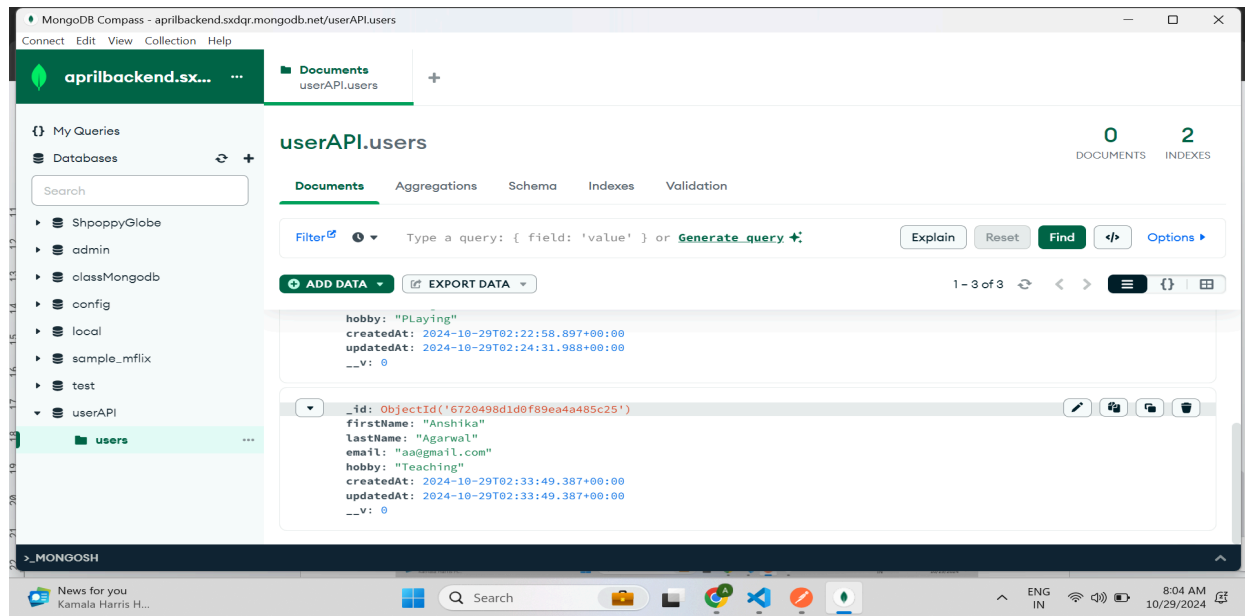
API CALL for Add → POST

1. Added new user with name : Anshika and hobby : teaching with Unique email Id too

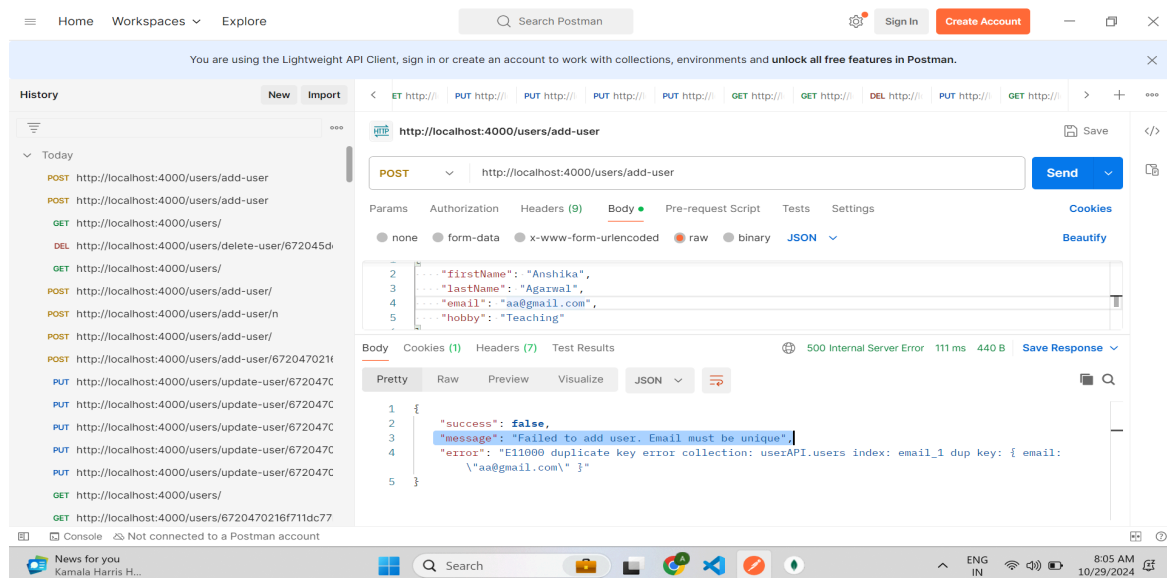
POSTMAN SS



MONGODB COMPASS Showing added user



Error for email should be unique (Email Validation)



API CALL for Update User by id → PUT

1. Updated user with email :ag@gmail.com and hobby: coding

The screenshot shows the Postman interface with a PUT request to `http://localhost:4000/users/update-user/6720498d1d0f89ea4a485c25`. The request body is a JSON object: `{ "firstName": "Anshika", "lastName": "Agarwal", "email": "ag@gmail.com", "hobby": "Coding" }`. The response is a JSON object: `{ "success": true, "msg": "User Updates Successfully", "updatedUser": { "_id": "6720498d1d0f89ea4a485c25", "firstName": "Anshika", "lastName": "Agarwal", "email": "ag@gmail.com" } }`.

The screenshot shows the MongoDB Compass interface for the `userAPI.users` collection. It displays two documents. The first document has `hobby: "Playing"`. The second document, which is highlighted, has `firstName: "Anshika", lastName: "Agarwal", email: "ag@gmail.com", hobby: "Coding"`, matching the data from the Postman request.

API CALL for Fetch User by id → GET

1. Get the user with id

The screenshot shows the Postman interface with a GET request to `http://localhost:4000/users/6720498d1d0f89ea4a485c25`. The request is sent, and the response is displayed in the Body tab. The response is a JSON object with the following structure:

```
{  "success": true,  "msg": "Get the user by id",  "user": {    "_id": "6720498d1d0f89ea4a485c25",    "firstName": "Anshika",    "lastName": "Agarwal",    "email": "ag@gmail.com",    "hobby": "Coding"  }}
```

The status bar at the bottom indicates a 200 OK response with a 134 ms latency and 490 B of data.

The screenshot shows the MongoDB Compass interface for the `userAPI.users` collection. The collection contains two documents. The first document is highlighted, showing the following fields:

```
{  "hobby": "Playing",  "createdAt": "2024-10-29T02:22:58.897+00:00",  "updatedAt": "2024-10-29T02:24:31.988+00:00",  "__v": 0}
```

The second document is also visible, showing the following fields:

```
{  "_id": ObjectId('6720498d1d0f89ea4a485c25'),  "firstName": "Anshika",  "lastName": "Agarwal",  "email": "ag@gmail.com",  "hobby": "Coding",  "createdAt": "2024-10-29T02:33:49.387+00:00",  "updatedAt": "2024-10-29T02:37:29.000+00:00",  "__v": 0}
```

The interface shows 0 documents and 2 indexes for the collection.

API CALL for Fetch All User → GET

1. Get all user

The screenshot shows the Postman application interface. On the left, the 'History' panel lists several API requests. The main workspace displays a GET request to `http://localhost:4000/users/`. The 'Body' tab is selected, showing a JSON response with a status of 200 OK. The response body is formatted as JSON and contains the following data:

```
{
  "success": true,
  "msg": "Get all the user",
  "users": [
    {
      "_id": "672046fc16f711dc77bdd63e",
      "firstName": "Ashmita",
      "lastName": "Shrivas",
      "email": "a18@gmail.com",
      "hobby": "Coding",
      "createdAt": "2024-10-29T02:22:52.119Z",
      "updatedAt": "2024-10-29T02:22:52.119Z",
      "__v": 0
    }
  ]
}
```

The screenshot shows the MongoDB Compass application interface. The 'Documents' tab is selected for the `userAPI.users` collection. The interface displays two documents in the collection:

```
{
  "_id": ObjectId('6720470216f711dc77bdd63e'),
  "firstName": "Ashmita",
  "lastName": "Shri",
  "email": "a13@gmail.com",
  "hobby": "PLAYing",
  "createdAt": 2024-10-29T02:22:58.897+00:00,
  "updatedAt": 2024-10-29T02:24:31.988+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('6720498d1d0f89ea4a485c25'),
  "firstName": "Anshika",
  "lastName": "Ananya",
  "email": "a13@gmail.com",
  "hobby": "PLAYing",
  "createdAt": 2024-10-29T02:22:58.897+00:00,
  "updatedAt": 2024-10-29T02:24:31.988+00:00,
  "__v": 0
}
```

API CALL for Delete User by id → Delete

1. Deleted user with name : Anshika email :ag@gmail.com and hobby: coding

The screenshot shows the Postman application interface. On the left, the 'History' panel lists several API requests. The main workspace displays a DELETE request to `http://localhost:4000/users/delete-user/6720498d1d0f89ea4a485c25`. The 'Body' tab is selected, showing a JSON response in 'Pretty' format:

```
{
  "success": true,
  "message": "User deleted successfully",
  "deletedUser": {
    "_id": "6720498d1d0f89ea4a485c25",
    "firstName": "Anshika",
    "lastName": "Agarwal",
    "email": "ag@gmail.com",
    "hobby": "Coding",
    "createdAt": "2024-10-29T02:33:49.387Z",
    "updatedAt": "2024-10-29T02:37:29.000Z",
    "__v": 0
  }
}
```

The status bar at the bottom indicates the response is 200 OK, 86 ms, and 509 B. The system tray shows the date as 10/29/2024.

The screenshot shows the MongoDB Compass application. The 'userAPI.users' collection is selected, showing 2 documents. The 'Documents' tab is active, displaying the following JSON documents:

```
{
  "_id": ObjectId('672046fc16f711dc77bdd63c'),
  "firstName": "Ashmita",
  "lastName": "Shrivas",
  "email": "a1@gmail.com",
  "hobby": "Coding",
  "createdAt": 2024-10-29T02:22:52.119+00:00,
  "updatedAt": 2024-10-29T02:22:52.119+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('6720470216f711dc77bdd63e'),
  "firstName": "Ashmita",
  "lastName": "Shri",
  "email": "a13@gmail.com",
  "hobby": "Playing"
}
```

The left sidebar shows the database structure, including collections like 'ShpoppyGlobe', 'admin', 'classMongodb', 'config', 'local', 'sample_mflix', 'test', and 'userAPI'. The system tray shows the date as 10/29/2024.