**NoteNest - The Note Taking App**

The NoteNest App back-end is designed to offer users a reliable and secure environment for managing their personal notes. It features user authentication with JWT-based token security, ensuring that each user's data remains private and protected. The API enables users to perform CRUD (Create, Read, Update, Delete) operations on their notes, allowing them to organize information effectively.

The backend is developed using Node.js and Express.js, leveraging MongoDB as the database to store user information and notes efficiently. Mongoose ODM is utilized to interact with MongoDB, simplifying data manipulation. Middleware such as express-validator is used for request validation, while bcryptjs ensures password encryption for secure authentication.

One of the core functionalities of the Notes App back-end is its authentication mechanism. Users must sign up and log in to access their notes. Each API request is validated using JWT authentication, ensuring that only authorized users can manage their data.

The Notes App back-end also includes error handling mechanisms to provide informative responses in case of invalid requests or server issues. This ensures a smooth user experience by preventing unauthorized access and handling common issues gracefully.

Designed to be modular and scalable, this back-end can be extended to include additional features such as note sharing, categorization, and reminders. It serves as an excellent foundation for developers who want to build a fully functional note-taking application.

Scenario Based Case Study

Background: Amit, a second-year Computer Science student, struggles with organizing his study materials. He often finds it difficult to keep track of important lecture notes, project ideas, and personal to-do lists. He has tried using traditional notebooks and various note-taking apps, but they either lack essential features or have cluttered interfaces. Amit is looking for a simple yet effective solution that allows him to store, manage, and retrieve his notes securely from any device.

Problem: During his semester exams, Amit realized that he had lost some important notes on a specific programming concept. His notes were scattered across different platforms—some written in physical notebooks, some stored in his mobile notes app, and others saved in multiple cloud drives. This inefficiency resulted in wasted time and stress, making his revision process difficult. Amit needed a centralized, easy-to-use platform where he could store, organize, and retrieve all his study materials efficiently.

Solution: Amit discovered the Notes App, a simple yet powerful web-based note-taking application with a secure back-end. After signing up, he was able to create, update, and organize his notes effortlessly. With user authentication and authorization, his data remained safe. The search and filter features helped him quickly find the required notes. Additionally, he could categorize his notes using tags like "Lecture Notes," "To-Do," and "Project Ideas," making organization seamless.

One day, Amit accidentally deleted an important note, but thanks to the automatic backup feature, he was able to recover it quickly. He also found that he could access his notes from multiple devices, allowing him to review his study material on the go.

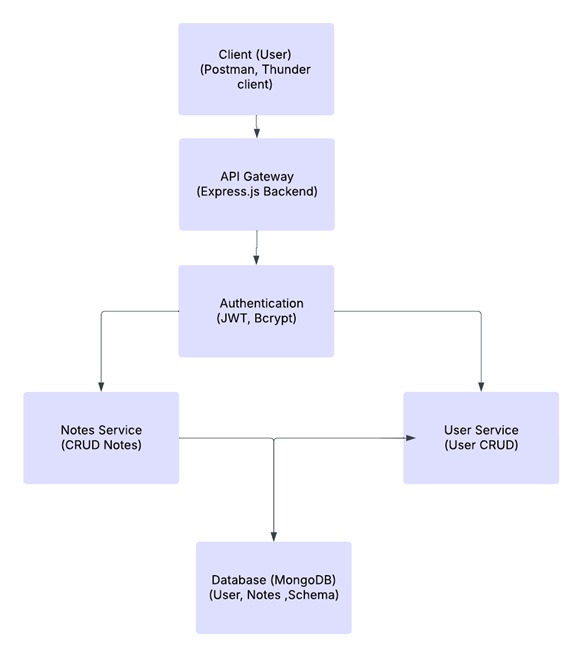
Usage:

* Student Usage: Amit uses the app to take notes on various subjects and organize them with relevant tags.
* Professional Usage: His senior, working as a software developer, uses the app to store project ideas, meeting notes, and technical documentation.
* Personal Usage: Amit’s friend, Riya, uses it to maintain a daily journal, grocery lists, and travel plans.

Outcome:

With the Notes App, Amit has streamlined his study process. He no longer worries about losing important information, as all his notes are securely stored and easily accessible. The app's intuitive interface and useful features have enhanced his productivity, helping him focus more on learning and less on managing scattered notes.

Technical Architecture:



Client (Frontend or API Testing Tools)

The user interacts with the application through a web or mobile front-end, or directly via

**ER-Diagram**

A diagram of a computer program

AI-generated content may be incorrect.

**Key Features:**

**1. User Dashboard**: A central dashboard where users can view, create, edit, and delete their notes in an organized manner.

**2. Note Creation**: Users can create and save notes with a title, description, and optional tags for easy categorization.

**3. Edit & Update Notes**: Existing notes can be updated with new content without creating a new entry.

**4. Delete Notes**: Users can remove notes that are no longer needed.

**5. User Authentication**: Secure login and signup system using JWT (JSON Web Token) authentication.

**6. Secure Data Storage**: All user notes are securely stored in a MongoDB database with unique user associations.

**7. Tagging System**: Notes can be categorized using tags for easier organization and filtering.

**8. Search & Filter**: Users can quickly find notes by searching based on title, content, or tags.

**9. Date Tracking**: Each note includes a timestamp to track when it was created or last updated.

**PRE REQUISITES**

To develop this **Notes App** using **Node.js, Express.js, MongoDB**, and **JWT authentication**, ensure you have the following prerequisites installed:

**1. Node.js and npm**

· Install **Node.js**, which includes **npm (Node Package Manager)**, required for running JavaScript on the backend.

**· Download:** [Node.js Official Website](https://nodejs.org/en/download/)

**· Installation Guide:** [Node.js Installation](https://nodejs.org/en/download/package-manager/)

“Npm init”

**2. MongoDB**

· Set up **MongoDB** to store notes securely.

· Install MongoDB locally or use a cloud-based service like **MongoDB Atlas**.

**· Download:** [MongoDB Community Edition](https://www.mongodb.com/try/download/community)

**· Installation Guide:** [MongoDB Installation Docs](https://docs.mongodb.com/manual/installation/)

**3. Express.js**

· Express.js is a web framework for **Node.js**, handling routing, middleware, and API development.

Install Express.js via npm:

“npm install express”

**4. Mongoose (MongoDB ODM)**

· Mongoose helps interact with MongoDB efficiently.

· Install Mongoose:

“npm install mongoose”

**5. JSON Web Token (JWT)**

· JWT is used for user authentication.

· Install JWT package:

“npm install jsonwebtoken”

**6. bcrypt.js (Password Encryption)**

· Used to securely hash user passwords.

· Install bcrypt:

“npm install bcryptjs”

**7.Postman or Thunder Client(Vs extension)**

**Purpose**: A comprehensive API platform for designing, testing, and documenting APIs.

**Download**: Access the latest version suitable for your operating system from the [Postman Downloads](https://www.postman.com/downloads/) page.

**Installation Guide**: Follow the instructions provided in the [Postman Installation Guide](https://www.postman.com/downloads/) to set up the application on your system.

**Roles and Responsibility**



·  Develop and maintain the **server-side logic** using **Node.js** and **Express.js**.

·  Implement **user authentication and authorization** using **JWT** and **bcrypt.js**.

·  Design and manage the **MongoDB database schema** using **Mongoose**.

·  Create **RESTful APIs** for handling CRUD operations (Create, Read, Update, Delete) for notes.

·  Ensure **data security** and **user access control** mechanisms are in place.

·  Optimize **database queries** to improve performance.

·  Implement **middleware** for logging, request validation, and error handling.

·  Ensure **API documentation** is maintained for frontend developers.

**Project Flow**

Let’s start with the project development with the help of the given activities.

**Project Setup and Configuration**

**1. Install Required Tools and Software**

Ensure you have the following installed on your system:

* **Node.js** (Server-side JavaScript runtime)

Download: <https://nodejs.org/en/download/>

* **MongoDB** (Database)

Download: <https://www.mongodb.com/try/download/community>

**2. Create Project Folders and Files**

/notes-app-backend

??? /models          # Database models (schemas)

?    ??? User.js         # User schema

?    ??? Note.js         # Note schema

??? /routes          # API route files

?    ??? authRoutes.js   # Routes for authentication (register, login)

?    ??? noteRoutes.js   # Routes for CRUD operations on notes

??? /controllers     # Business logic for handling requests

?    ??? authController.js # Handles authentication-related logic

?    ??? noteController.js # Handles notes-related logic

??? /middleware      # Middleware functions for authentication & error handling

?    ??? authMiddleware.js # Middleware to protect routes (JWT authentication)

?    ??? errorMiddleware.js # Centralized error handling

??? /config          # Configuration files (DB connection, environment variables)

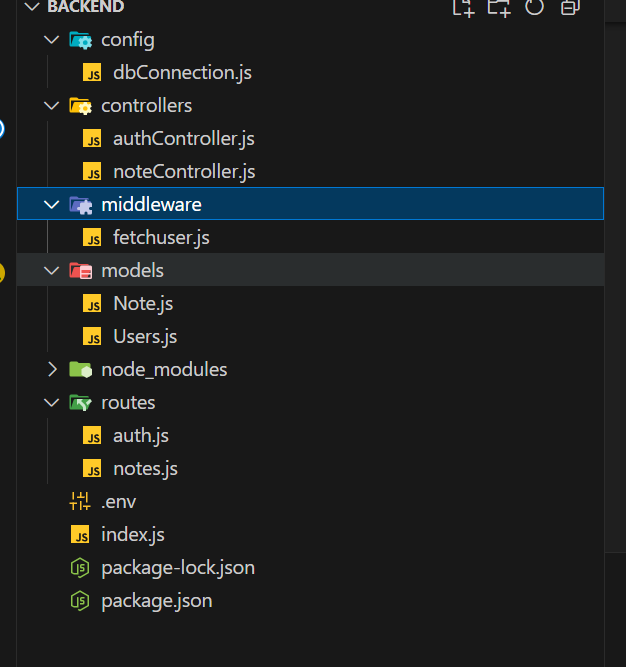
?    ??? db.js          # MongoDB connection setup

??? /public          # Static files (if needed)

??? .env             # Environment variables (MongoDB URI, JWT secret)

??? index.js         # Main entry point of the backend

??? package.json     # Project dependencies



**3. Install Required npm Packages**

**Backend npm Packages**

Run the following command inside your project directory:

“**npm install express mongoose cors dotenv bcryptjs jsonwebtoken express-validator nodemon**”

* **Express** - Web framework for Node.js
* **Mongoose** - ODM for MongoDB
* **Dotenv** - For managing environment variables
* **Bcrypt.js** - For password encryption
* **Jsonwebtoken (JWT)** - For authentication
* **Express-validator** - Middleware for request validation

**Backend Development**

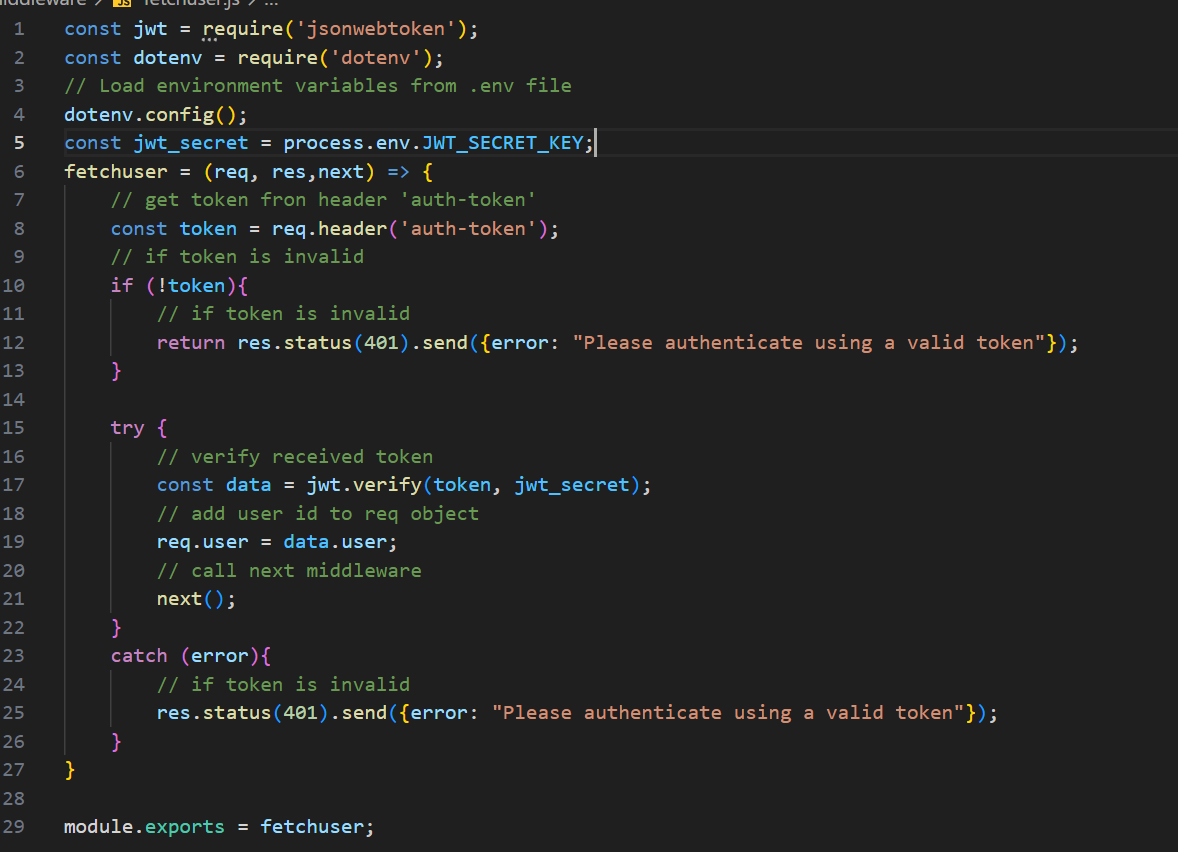
* **1. Setup Express Server**
  + Create an index.js file in the root folder.
  + Configure the server with necessary middleware (cors, body-parser, express.json()).
  + Start the server using nodemon.

**Example index.js Code:**



**2. User Authentication**

* + Implement routes and middleware for **user registration, login, and authentication**.
  + Protect sensitive routes using **JWT-based authentication**.



**3. Define API Routes**

* + Create separate files inside the /routes folder for different functionalities:
  + authRoutes.js (User Authentication)
  + noteRoutes.js (CRUD operations for notes)



**4. Implement Error Handling**

* + Use a centralized error-handling mechanism.
  + Return appropriate **error responses** with **HTTP status codes**.

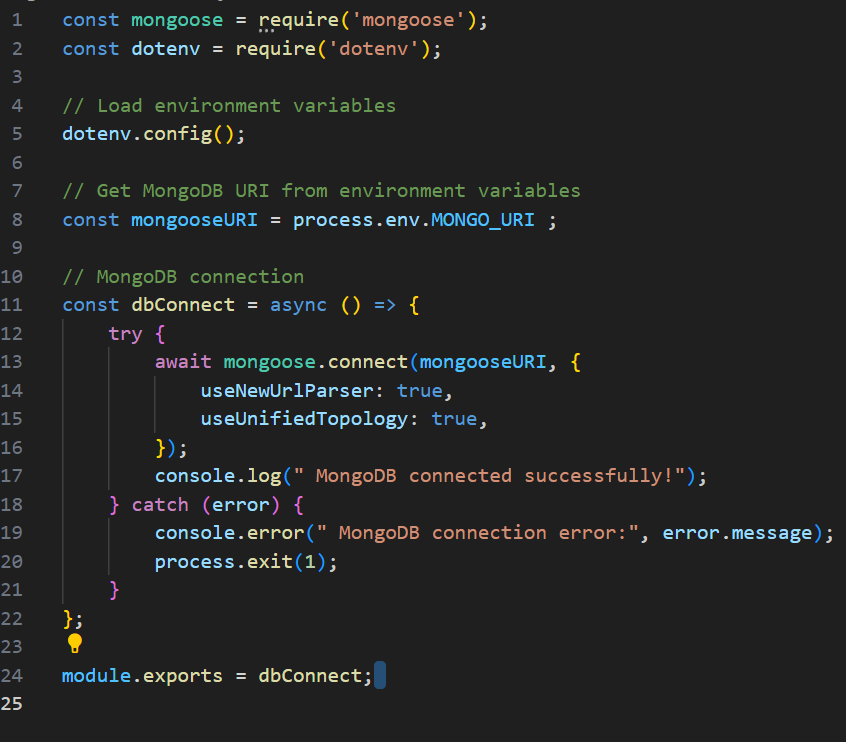
**Database**

**Mongdb Connection and Model creation:**[**Demo link**](https://www.loom.com/share/7b66e1670b434bf6b5ef29ff62b4248b?sid=146fddb9-6d67-4381-86f6-9f0521066bb5)

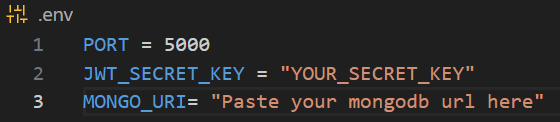
1. **Install and Configure MongoDB**

* Install **Mongoose** (npm install mongoose).
* Create a **database connection file** inside the /config folder.

**Database Connection File (**config/db.js**)**

****

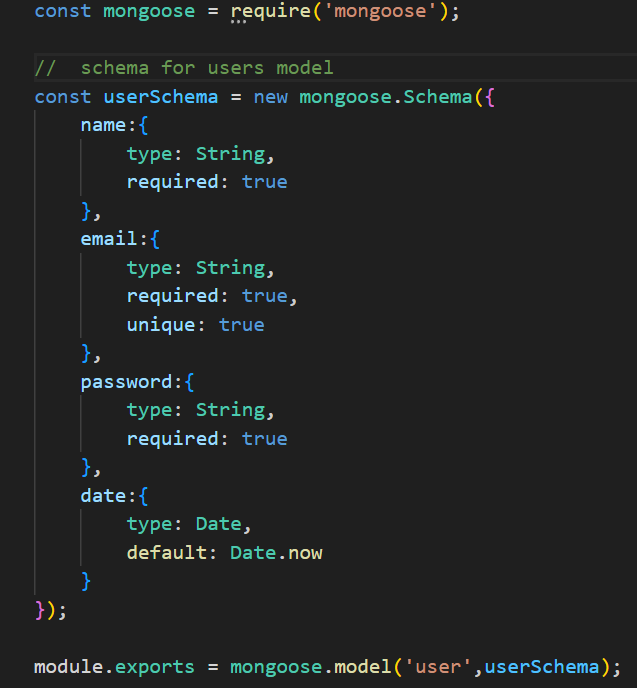
Create a .env file in the root folder and add the MongoDB connection string:



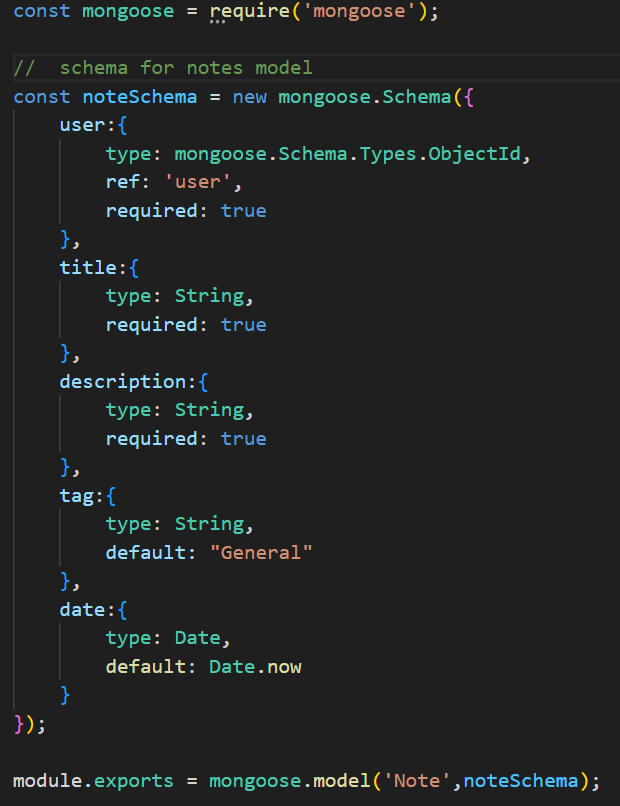
**2. Define Database Models**

Inside /models, create User.js and Note.js.

**User\_Model:**

****

**Note\_Model:**



**3. Connect Database to Backend**

Modify index.js to ensure database is connected before starting the server.

**4. Start the Server**

Run the following command:

“npm start”

**API Testing**

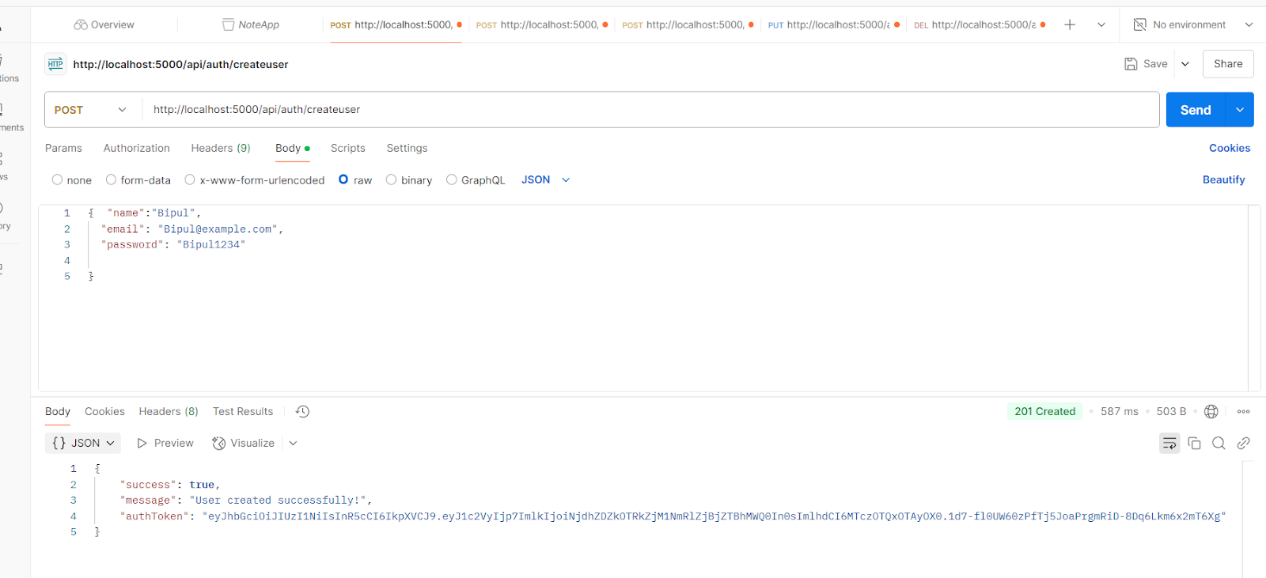
·  Install Postman or Thunder Client (VS Code extension).

·  Start the backend server:

“npm start”

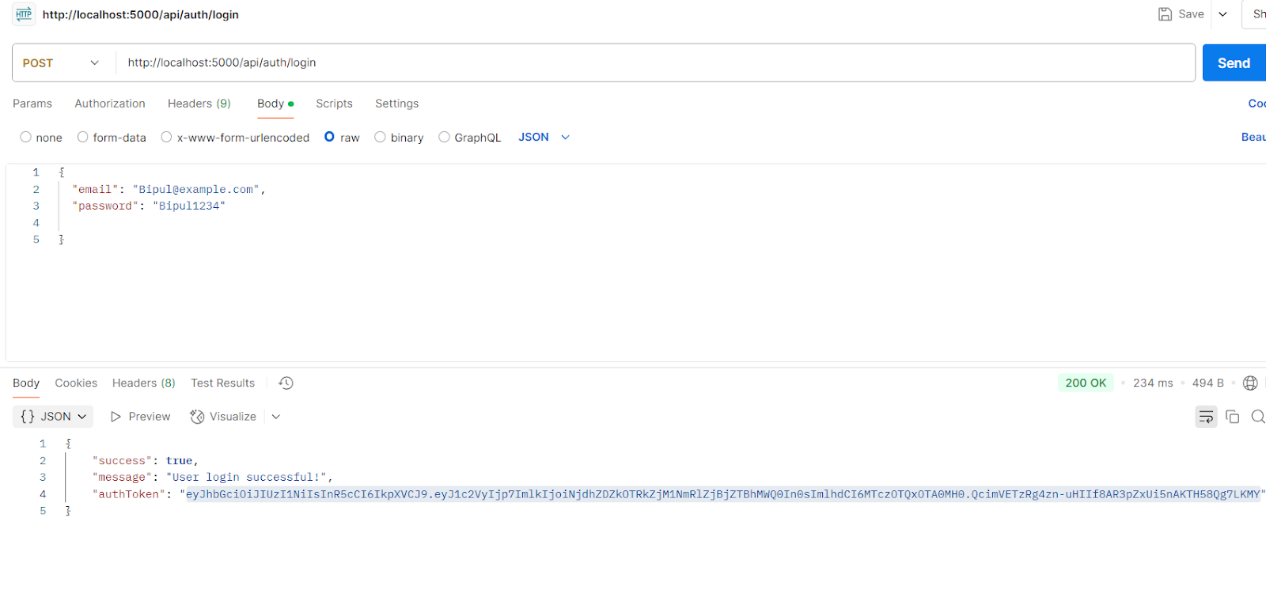
**1.Test Authentication APIs:**

* **Register a new user:**The POST /api/auth/createuser endpoint allows clients to register a new user by submitting their details to the server.



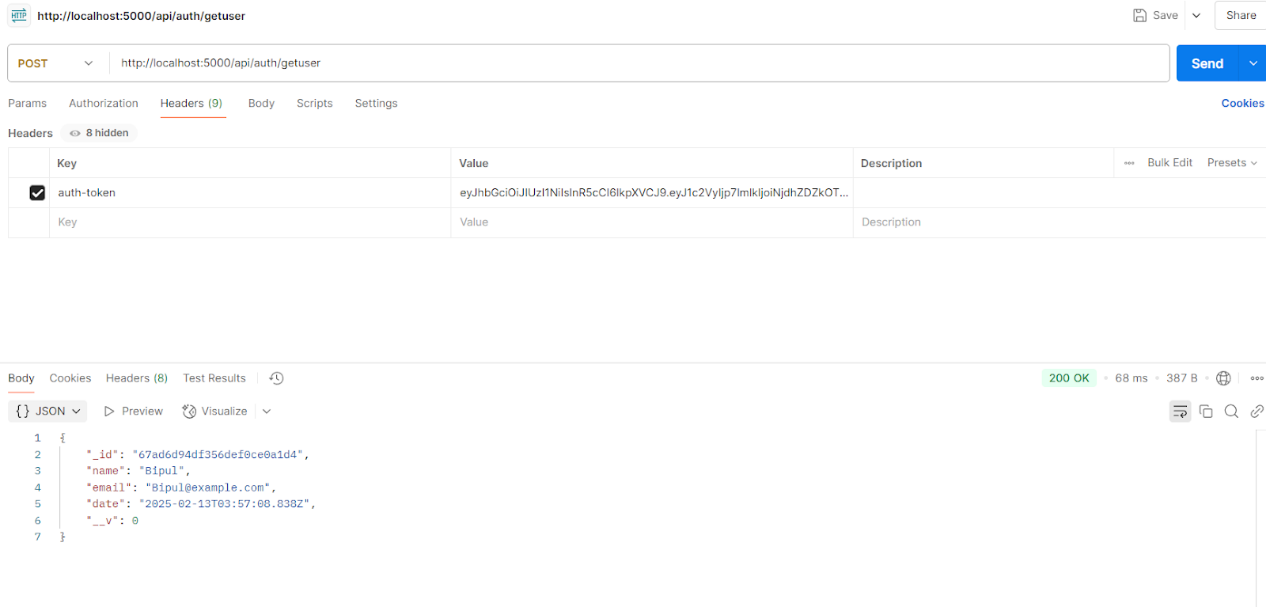
**Login User:**

The POST /api/auth/login endpoint allows users to authenticate by submitting their credentials, typically returning a token upon successful login.



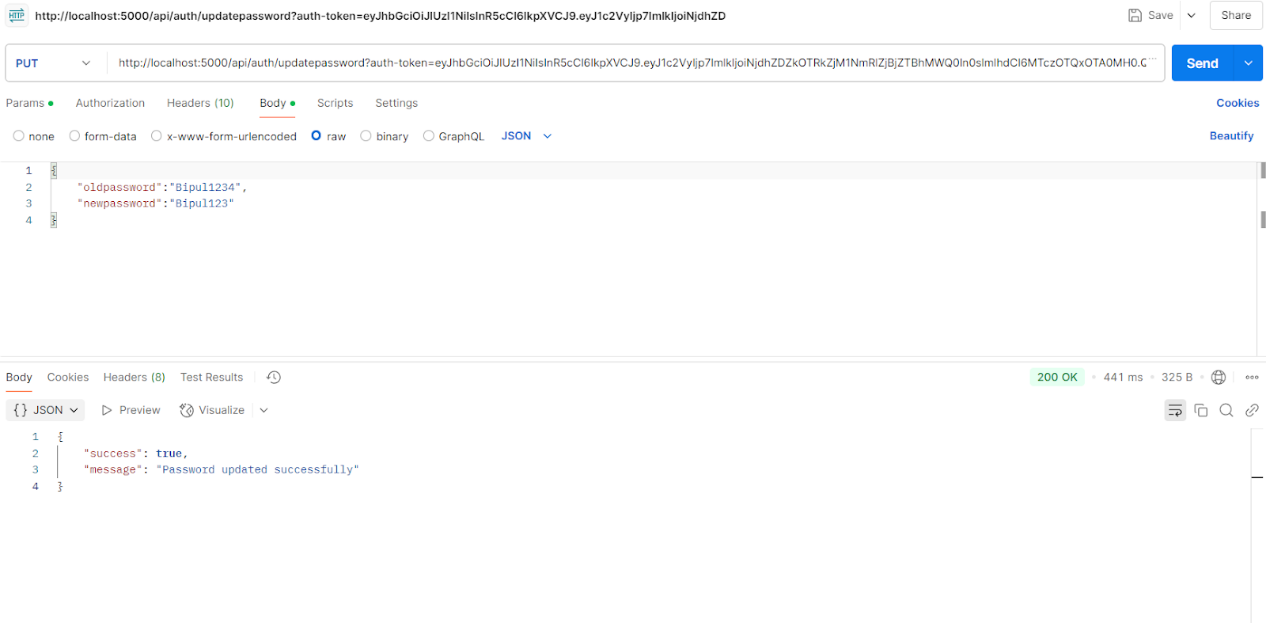
**Get User Details**

The POST /api/auth/getuser endpoint retrieves the authenticated user's details, typically requiring a valid authentication token in the request headers.



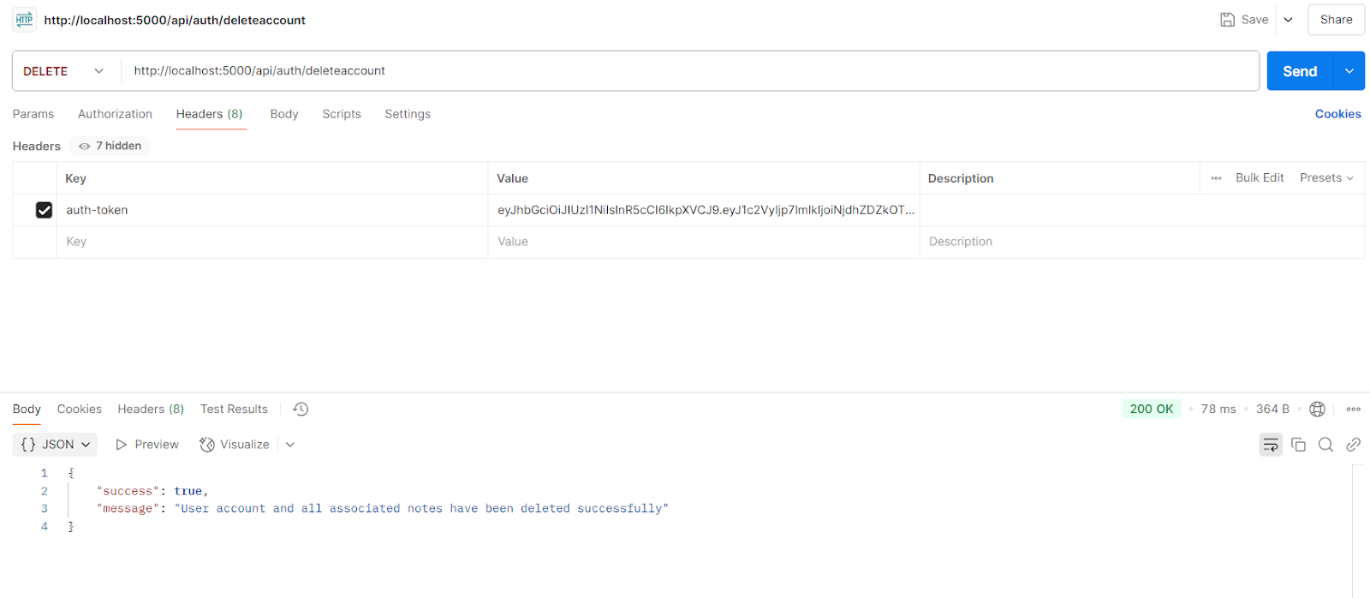
**Update Password**

The PUT /api/auth/update-password endpoint allows authenticated users to securely change their current password by providing their existing password and a new password.



**Delete Account**

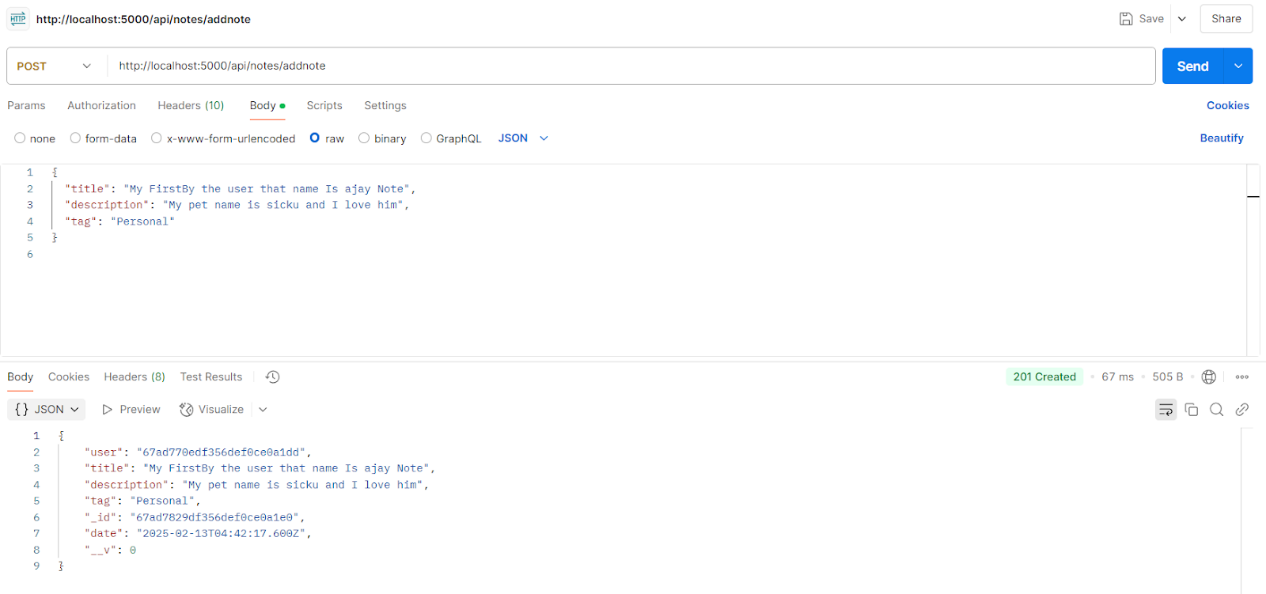
The DELETE /api/auth/deleteaccount endpoint allows authenticated users to permanently delete their account from the Database.



1. **Test Notes APIs:**

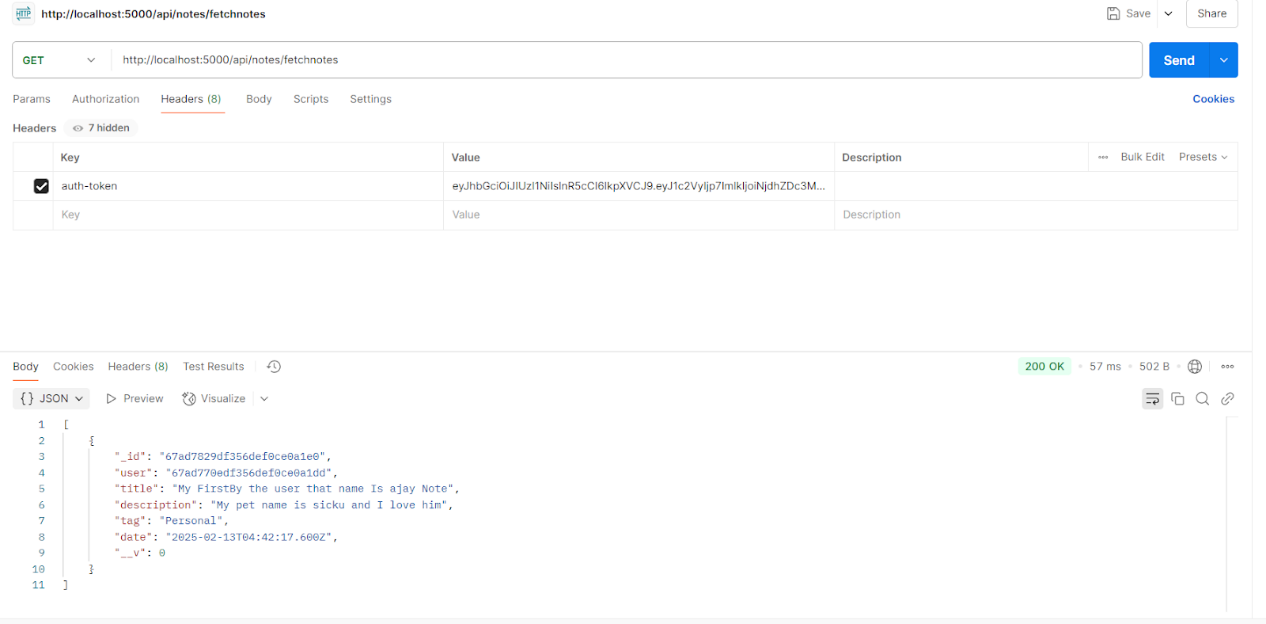
**Create a Note:**

The POST /api/notes/addnote endpoint allows authenticated users to create a new note by submitting its content to the server.



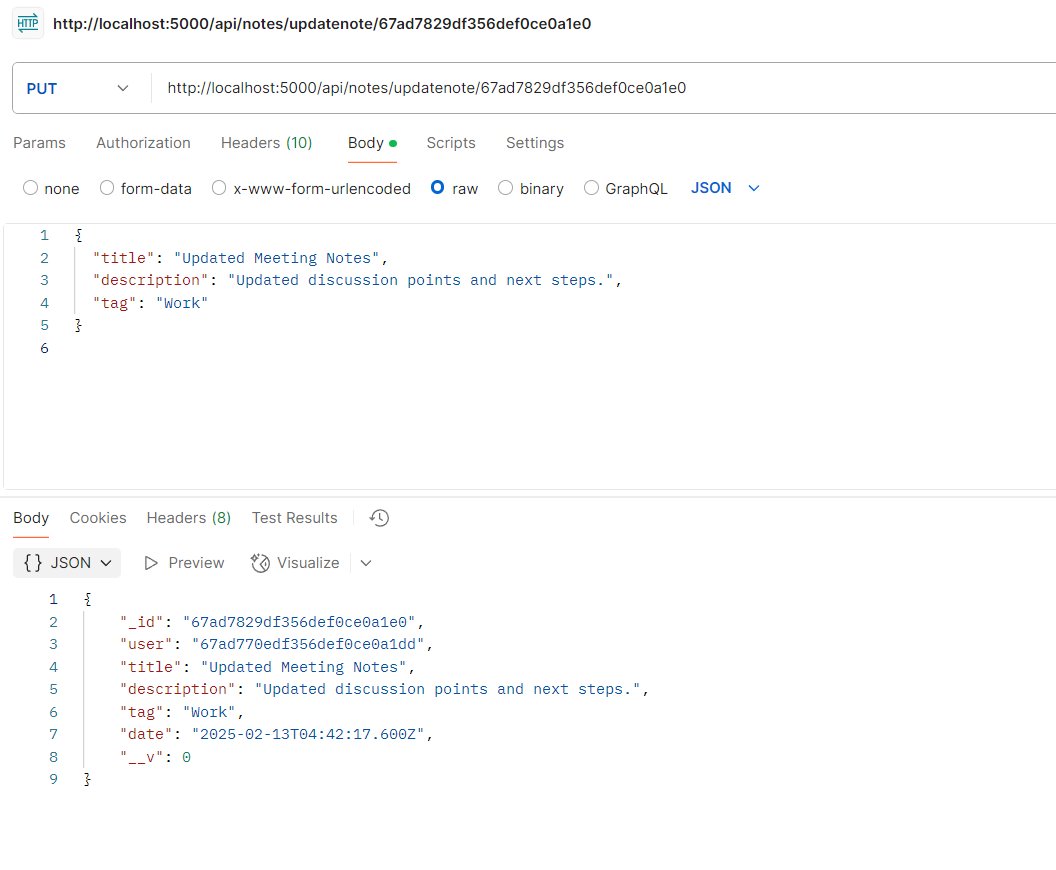
**Fetch All Notes**

The GET /api/notes/fetchnotes endpoint allows authenticated users to retrieve all their saved notes from the server.



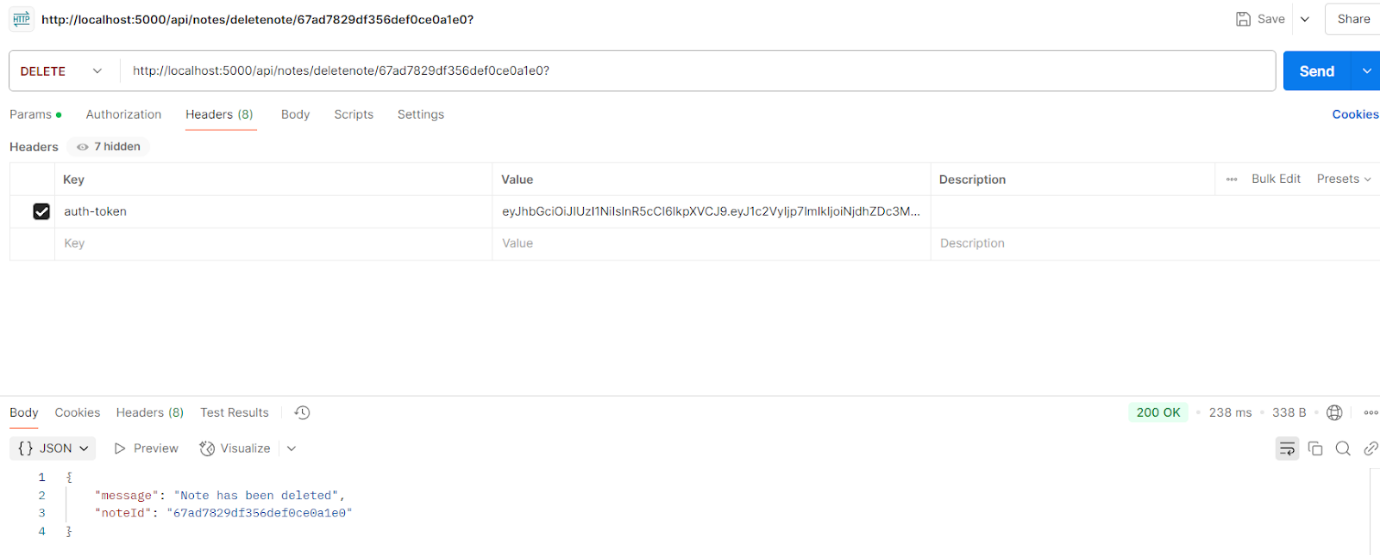
**Update a Note**

The PUT /api/notes/updatenote/:id endpoint allows authenticated users to modify an existing note identified by its unique ID.



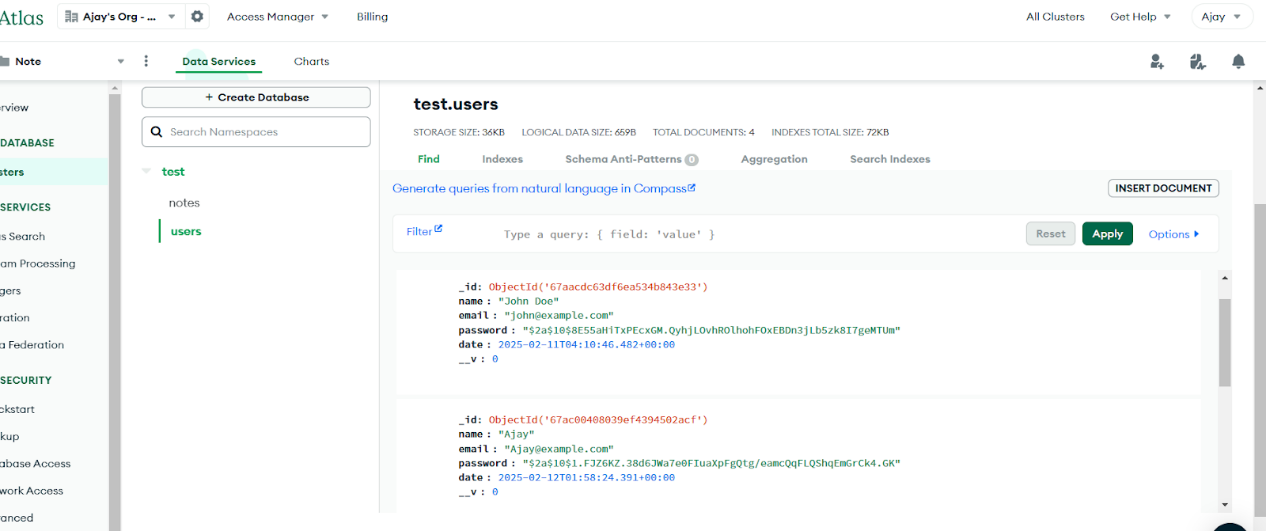
**Delete a Note**

The DELETE /api/notes/deletenote/:id endpoint allows authenticated users to remove a specific note identified by its unique ID.

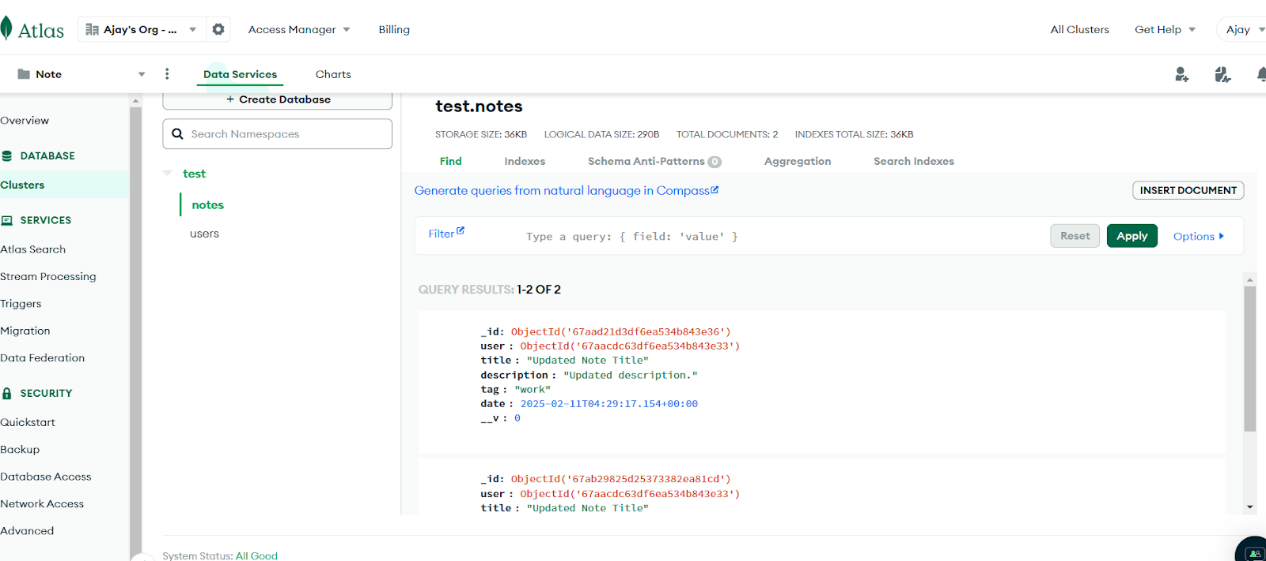


**Database Schema Overview**

**User:**

****

**Notes:**

****

**Conclusion :**

In conclusion, this note-taking application stands as a testament to the effective integration of modern web technologies to create a tool that enhances productivity and information management. Its intuitive design, coupled with robust backend support, ensures that users can rely on the application for their daily note-taking needs, ultimately contributing to improved organization and efficiency in both personal and professional contexts.