**📘 Project Summary**

**Document Management Application** is a secure, role-based REST API built with Spring Boot. It allows users to register, authenticate, upload and manage documents, perform content-based Q&A search, and handle file storage. Admin users have elevated permissions to manage users and server-side storage directly.

This application is modular, scalable, and uses best practices like JWT authentication, layered architecture, and validation. The system is suitable for enterprise-level document processing and retrieval.

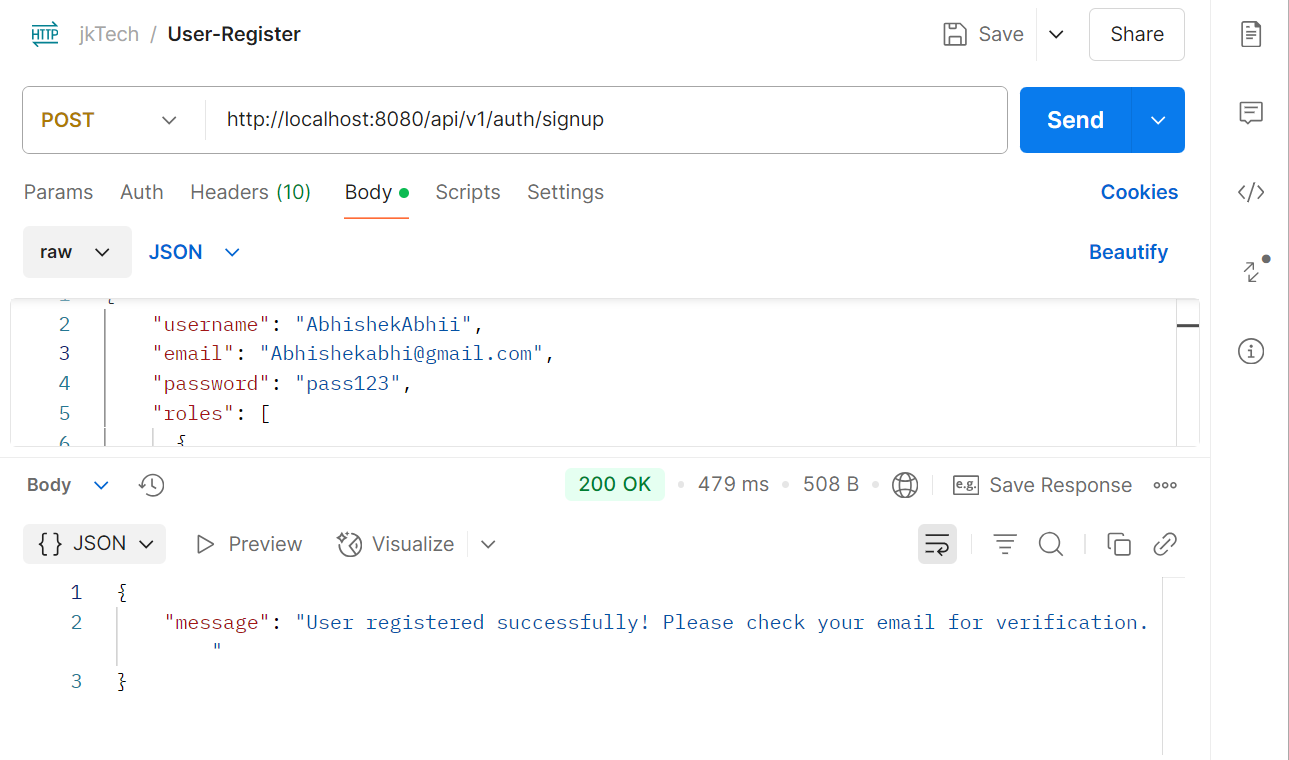
**🧠 Database Configuration Explanation (MySQL)**

You only need to provide the **username**, **password**, and **port number** of your MySQL database.  
The application will automatically connect and **create the database and tables** as needed — you don't need to create the database manually.

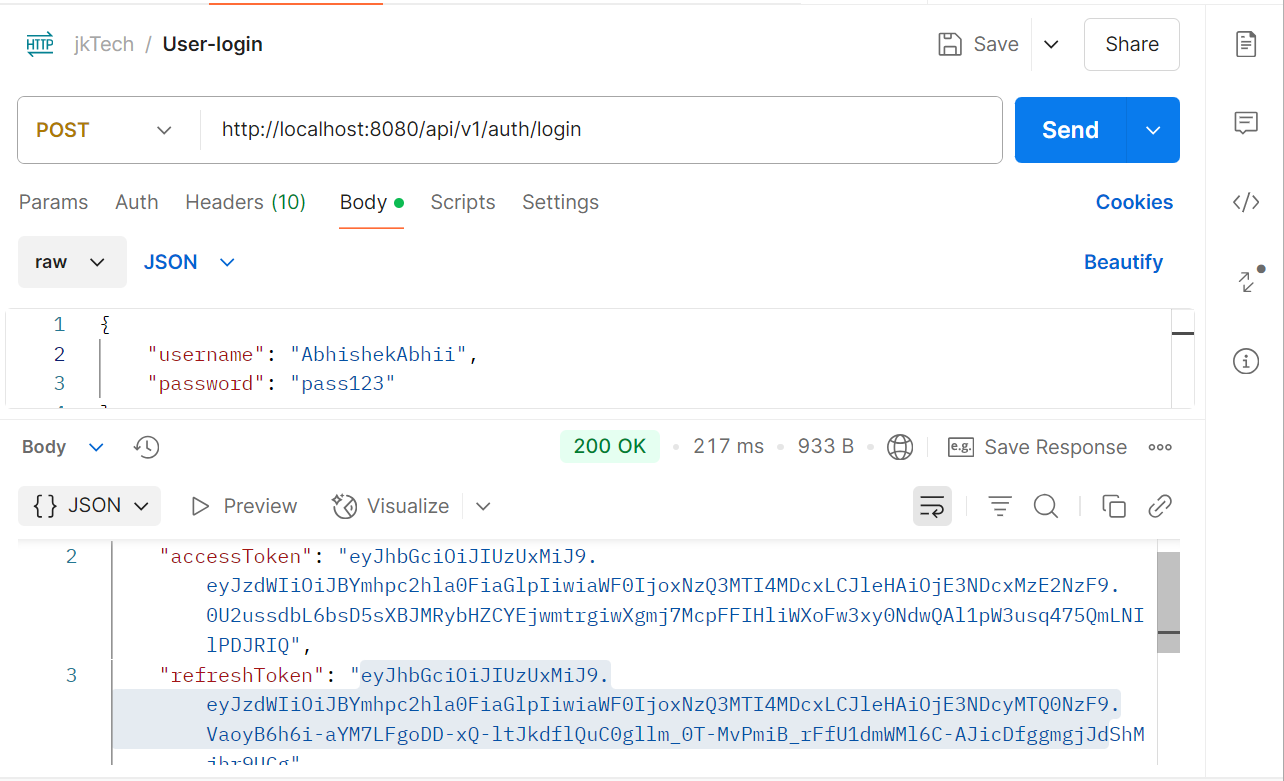
This makes setup easy and fully automated during the first run of the application.

**🔐 User Authentication Flow**

1. **Signup First**  
   You must first hit the **Signup API** (/api/v1/auth/signup)  
   ➤ Provide all required user details (e.g., username, email, password, role).  
   ➤ After successful registration, the user is saved in the database.



1. **Login with Credentials**  
   Hit the **Login API** (/api/v1/auth/login)  
   ➤ Provide the registered **username and password**



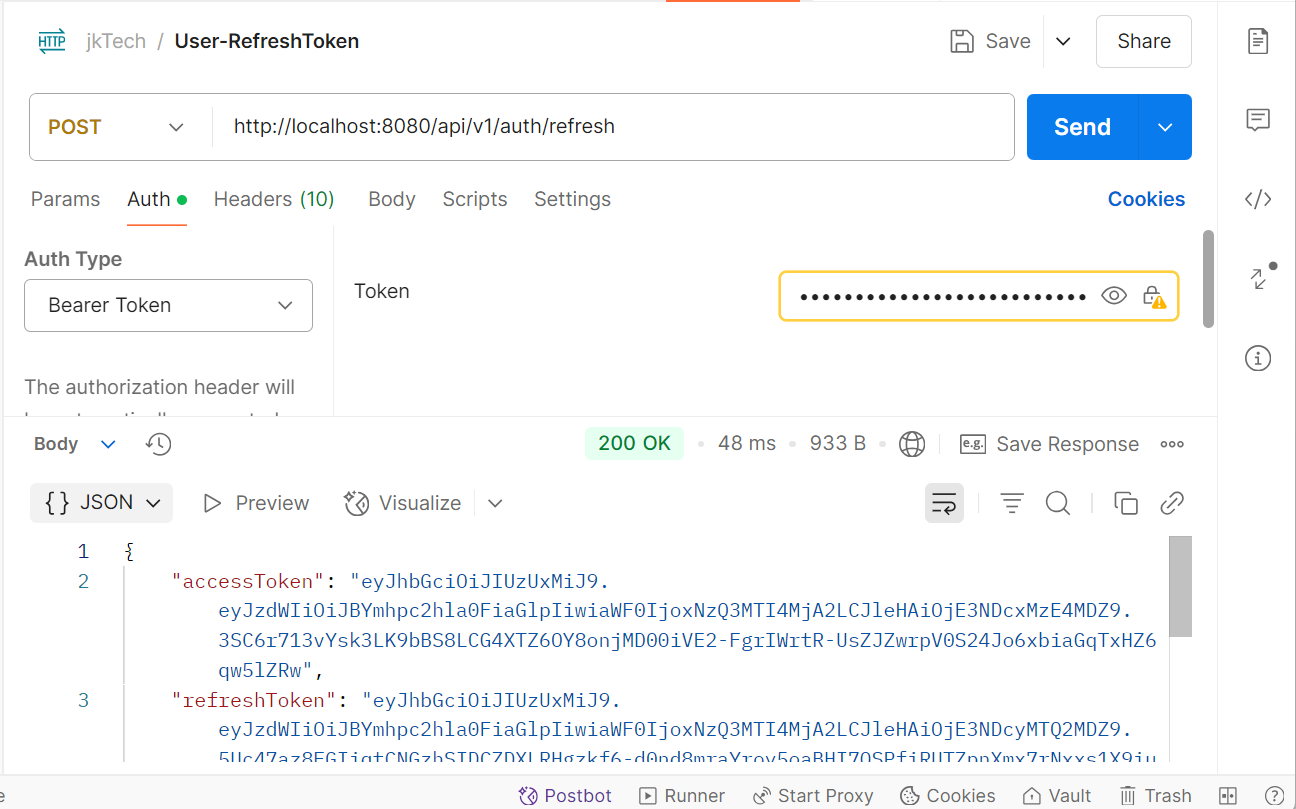
1. **Receive Tokens**  
   If login is successful, the API response will return:
   * ✅ **Access Token** (used to access secure APIs)
   * 🔁 **Refresh Token** (used to get a new access token when it expires)
   * ⏱ **Token Expiry Time**
   * 🧑 **User Info** (username, roles, etc.)
2. **Use Access Token in APIs**
   * Add the token in headers for every secured API request:

makefile

CopyEdit

Authorization: Bearer <access\_token>

1. **Refresh Token Flow**  
   When the access token expires, call the **refresh API** (/api/v1/auth/refresh)  
   ➤ Pass the **refresh token** in the Authorization header  
   ➤ A new access token will be returned



**6.🚪 Logout**  
To log out the user, call the:

* POST /api/v1/auth/logout endpoint  
  ➤ This will invalidate the session or tokens (depending on implementation)

📄 **API Endpoint Summary**

**🔐 AuthController – Authentication APIs**

| **HTTP Method** | **Endpoint** | **Description** | **Access** |
| --- | --- | --- | --- |
| POST | /api/v1/auth/signup | Register new user | Public |
| POST | /api/v1/auth/login | Login with username & password | Public |
| POST | /api/v1/auth/refresh | Get new access token using refresh | Authenticated |
| POST | /api/v1/auth/logout | Logout and invalidate session/token | Authenticated |

**📁 DocumentController – Document Management**

| **HTTP Method** | **Endpoint** | **Description** | **Access** |
| --- | --- | --- | --- |
| POST | /api/v1/documents | Upload a single document | Authenticated |
| POST | /api/v1/documents/batch | Upload multiple documents | Authenticated |
| GET | /api/v1/documents/{id} | Get document details by ID | Authenticated |
| GET | /api/v1/documents | Get all documents (paged) | Authenticated |
| GET | /api/v1/documents/search?keyword=xyz | Search documents by keyword | Authenticated |
| GET | /api/v1/documents/user | Get documents uploaded by current user | Authenticated |
| GET | /api/v1/documents/{id}/download | Download a document by ID | Authenticated |
| DELETE | /api/v1/documents/{id} | Delete a document by ID | Authenticated |

**🤖 QAController – Q&A API**

| **HTTP Method** | **Endpoint** | **Description** | **Access** |
| --- | --- | --- | --- |
| POST | /api/v1/qa/ask | Ask a question for document context | Authenticated |
| GET | /api/v1/qa/documents/relevant?question=xyz | Get relevant documents for a question | Authenticated |

**👥 UserController – User Management (Admin/Self)**

| **HTTP Method** | **Endpoint** | **Description** | **Access** |
| --- | --- | --- | --- |
| GET | /api/v1/users/{id} | Get user by ID | Admin |
| GET | /api/v1/users/username/{username} | Get user by username | Admin or Self |
| GET | /api/v1/users | Get all users (paged) | Admin |
| PUT | /api/v1/users/{id} | Update user profile | Admin or Self |
| DELETE | /api/v1/users/{id} | Delete a user | Admin |

**🗂️ StorageController – File System Operations (Admin Only)**

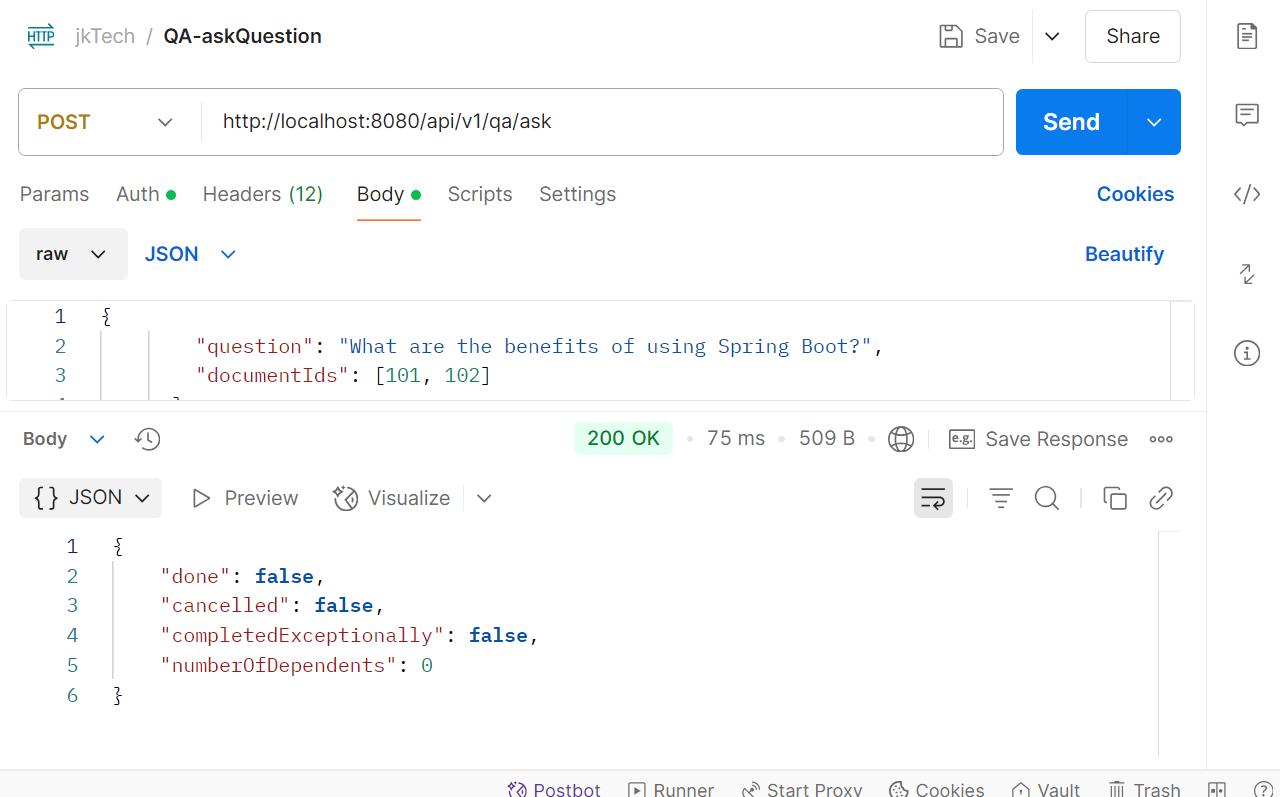
| **HTTP Method** | **Endpoint** | **Description** | **Access** |
| --- | --- | --- | --- |
| POST | /api/v1/storage/upload | Upload a file | Admin |
| GET | /api/v1/storage/files | List all stored files | Admin |
| GET | /api/v1/storage/files/{filename} | Download a stored file | Admin |
| DELETE | /api/v1/storage/files/{filename} | Delete a file by name | Admin |
| DELETE | /api/v1/storage/files | Delete all files | Admin |

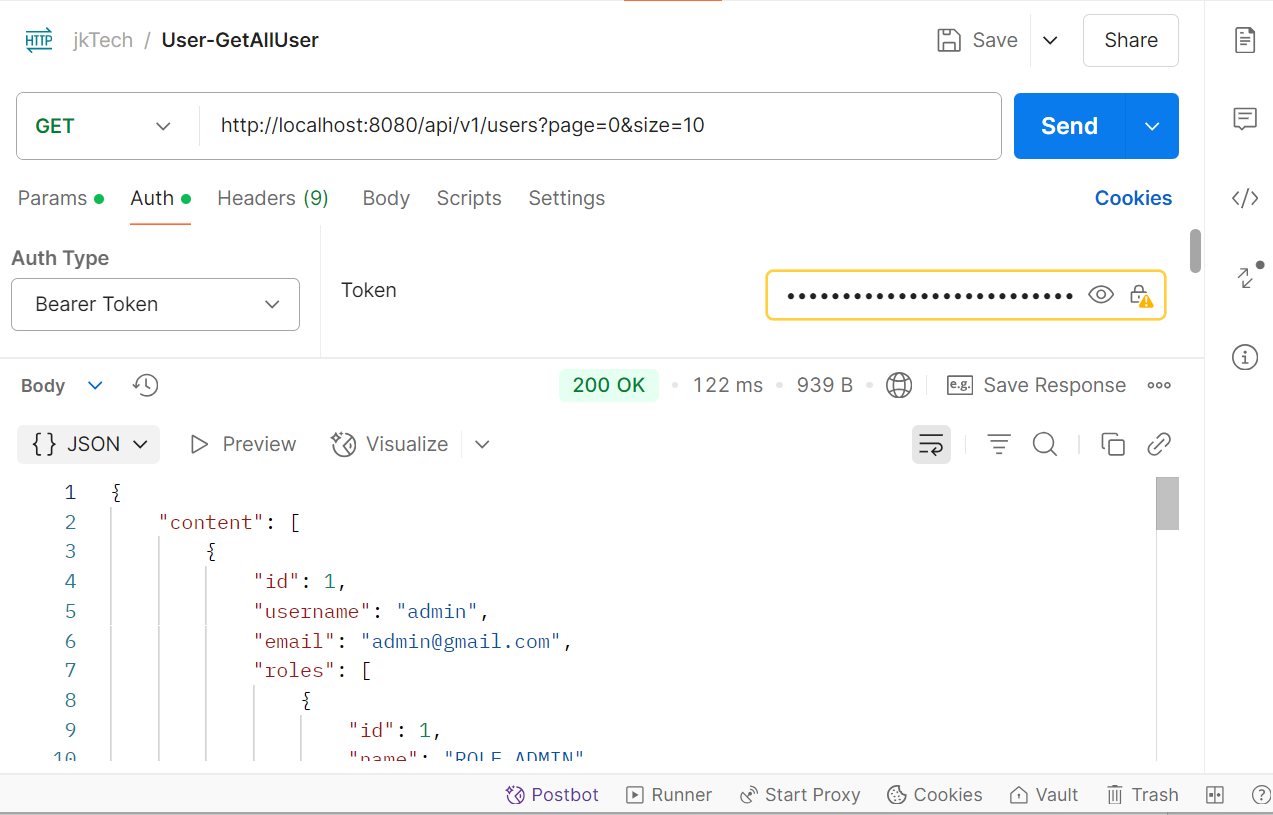
**✅ Notes:**

* You must **login** first to get an **access token** and **refresh token**
* Use the **access token** to hit any API by passing it in the header:

**Authorization: Bearer <access\_token>**

**Here are some images showing Postman running easily**

****

****

**📘 Database Tables Overview**

**This project uses 7 primary tables to manage documents, user roles, and related metadata. The tables included in the schema are:**

* **documents**
* **document\_content**
* **document\_keywords**
* **document\_type**
* **roles**
* **user**
* **user\_roles**