**Major project**

**On**

**Note-Vault**



**A Major Project Report Submitted to SAGE University, Indore**

**Towards Partial fulfilment for the award of**

**Master of Computer Application (MCA) degree**

**Guided by Submitted by**

Prof. <Guide name> Sumit Sharma

<Enrollment number>

**Institute of Computer Application**

**SAGE University, Indore**

[www.sageuniversity](http://www.sageuniversity)

**SAGE University, Indore**



**Approval Sheet**

The project entitled “**Note-vault**” submitted by **sumit sharma** approved as partial fulfilment for the award of the **MASTER OF COMPUTER APPLICATION (MCA)** by SAGE University, Indore.

**Internal Examiner External Examiner**

**Date: Date:**

**SAGE University, Indore**



**CERTIFICATE**

This is to certify that the project work entitled “**Sumit Sharma**” has been carried out by **Sumit Sharma** student of **MASTERS OF COMPUTER APPLICATION** under my supervision and guidance. He have submitted this project report towards partial fulfilment for the award of the **Master of Computer Application** by **SAGE University**, Indore.

**<HOD Name> <Guide Name>**

**(HOD) (Guide)**

**<HOI Name>**

**(HOI)**

**SAGE University, Indore**



**RECOMMENDATION**

The project entitled “**note-vault**” submitted by **Sumit Sharma** is a satisfactory account of the bona fide work done under my supervision is recommended towards partial fulfilment for the award of the **Master of Computer Application** by **SAGE University**, Indore.

**Date:**

**<HOD Name> <guide name>**

**(HOD) (Guide)**

**<HOI Name>**

**(HOI)**

**SAGE University, Indore**



**ACKNOWLEDGEMENT**

First and foremost, I would like to express my thankfulness towards **<Guide name>** of INSTIUTE OF COMPUTER APPLICATION for extending all the facilities needed to carry out this work, I take pride in saying that I have successfully completed my project work under his/her able guidance. He was a major support to us throughout project, being available at odd hours with his/her ideas, inspiration and encouragement. It is through his/her masterful guidance that I have been able to complete my project work.

I am also thankful to **<HOI name>(HOI) and**  **<HOD name>(HOD),** for giving their guidance throughout the Project phase.

**Sumit Sharma**

**(STUDENT)**

**SAGE University, Indore**



**CANDIDATE DECLARATION**

I hereby declare that the work which is being presented in this project report entitled “Note-Vault” in partial fulfilment for the award of **Master of Computer Application** is an authentic record of my own work carried out under the supervision and guidance of **<guide name>, SAGE University**, Indore.

I am fully responsible for the matter embodied in this report and it has not been submitted elsewhere for the award of any other degree.

**Date: Sumit Sharma**

**Table of Contents**

****

|  |  |  |
| --- | --- | --- |
| **S. NO** | **CONTENTS** | **PAGE NO.** |
| 1 | Introduction |  |
| 2 | Problem Statement/Abstract |  |
| 3 | Objectives |  |
| 4 | Hypothesis |  |
| 5 | Methodology/ Project Plan |  |
| 6 | Feasibility Study |  |
| 7 | Functional requirement |  |
| 8 | Non- functional requirement |  |
| 9 | Software Requirements |  |
| 10 | Hardware Requirements |  |
| 11 | DFD Diagram |  |
| 12 | ER Diagram |  |
| 13 | Use Case Diagram / Class Diagram |  |
| 14 | Class Diagram |  |
| 15 | Database Tables |  |
| 16 | Testing |  |
| 17 | Limitations |  |
| 18 | Future Scope |  |
| 19 | Conclusion |  |
| 20 | References |  |

## ****Introduction****

**NoteVault** is a full-stack web application developed to simplify the process of uploading, managing, and sharing academic notes online. It is aimed at students who want to store and access study material efficiently.

The platform supports two types of users:

* **Regular Users**: Can register, log in, update their profile, upload notes (title, description, college, course, semester), and view statistics such as total uploads and watch/download counts.
* **Admin**: Has a static login and can manage all users and notes, view total uploads, and monitor user activity.

NoteVault is built using the **MERN stack**:

* **MongoDB**: For storing user data and notes
* **Express.js**: For building RESTful APIs
* **React.js**: For creating a modern frontend UI
* **Node.js**: For backend logic and server handling

**Redux** is used for state management, ensuring smooth navigation between components and user roles. The system also uses **JWT** for authentication and **bcrypt** for password security.

Key features:

* Secure registration and login
* Role-based access control
* Notes uploading with metadata
* Profile update with photo
* Admin dashboard with full control
* Public API for showing all notes

This project demonstrates practical knowledge of full-stack development and addresses real-world problems faced by students in accessing and sharing study materials online.

## ****Problem Statement / Abstract****

In the current academic landscape, students often face significant difficulties in accessing and organizing educational resources, especially handwritten or self-prepared notes. Traditional methods of note sharing — such as physical distribution or informal messaging platforms — are inefficient, unreliable, and lack a centralized system for content management. These methods are further hindered by challenges such as physical distance, data loss, and lack of version control. There exists a clear gap in the availability of a structured, secure, and scalable system where students can share and retrieve study materials effortlessly.

Moreover, with the rapid shift towards digital learning, especially after the global pandemic, students now rely more on online resources than ever before. However, many existing platforms either charge high fees, limit user control, or lack proper filtering and categorization. In this context, there is a growing demand for an open academic platform where students can contribute and benefit from shared knowledge in an organized manner.

To address this gap, we have developed **NoteVault**, a full-stack web application designed to serve as a centralized repository for uploading, managing, and accessing educational notes. The platform is tailored for students, tutors, and academic contributors who want to share and explore notes across various colleges, courses, and semesters.

### Key Objectives:

* To create a secure and user-friendly platform for uploading and managing notes.
* To implement user authentication and profile management.
* To provide role-based access (Admin vs Regular User).
* To allow students to track the performance of their uploaded notes.
* To give administrators control over the entire platform’s content and users.
* To ensure scalability, security, and responsiveness across all devices.

### User Roles & Features:

1. **Regular Users:**
   * Register using full details (name, email, mobile, username, DOB, password).
   * Log in using a valid username and password.
   * Upload notes with fields such as title, description, college, course, semester, and the note file.
   * Update personal profile including username, password, and profile photo.
   * Access a personalized dashboard that shows uploaded notes, total watch count, and downloads.
   * View notes uploaded by others via a public API without logging in.
2. **Admin Panel:**
   * Static login using predefined admin credentials.
   * Ability to view all registered users and uploaded notes.
   * Control over platform data (user and note management).
   * Option to change admin credentials for better security.
   * View total user count and total uploaded notes in real time.

### Technology Stack:

* **Frontend:** React.js with Redux Toolkit for state management.
* **Backend:** Node.js and Express.js to handle routing and server-side logic.
* **Database:** MongoDB for data storage and retrieval.
* **Authentication:** JWT (JSON Web Tokens) and bcrypt for secure login and password protection.
* **File Uploads:** Multer is used to manage and store uploaded notes securely.

### System Modules:

1. **Authentication Module** – Handles secure registration and login for both users and admin.
2. **Profile Module** – Allows users to update personal information and photo.
3. **Notes Module** – Enables users to upload, update, and view notes with complete metadata.
4. **Dashboard Module** – Displays user-specific stats like total uploads and watch count.
5. **Admin Module** – Provides full control to the admin, including user and note management.

### Security Features:

* Passwords are hashed using bcrypt before storing in the database.
* JWT tokens are used for session validation.
* Input validations are performed on all fields to prevent injection and data corruption.
* Admin credentials are stored securely and can be modified only from the admin panel.

### Benefits of NoteVault:

* Centralized access to academic materials from multiple contributors.
* Time-saving and highly convenient for both students and faculty.
* Encourages collaborative learning and sharing of knowledge.
* Transparent and trackable note statistics for contributors.
* Scalable design, which can be enhanced with features like comments, ratings, or real-time chat in the future.

## ****Objectives****

The primary aim of the **NoteVault** project is to create a reliable, secure, and user-friendly platform that simplifies the process of uploading, managing, and accessing academic notes online. The project is designed to meet the needs of students and administrators by offering features that encourage collaboration, knowledge sharing, and digital academic resource management.

1. **To develop a centralized platform for academic note sharing.**
   * Enable students to upload handwritten or digital notes.
   * Provide categorized access based on college, course, and semester.
2. **To implement a secure authentication system.**
   * Ensure secure registration and login using encrypted credentials.
   * Maintain user sessions using JWT-based authentication.
3. **To support role-based access control.**
   * Allow regular users to access student functionalities (profile update, upload, dashboard).
   * Provide a separate admin login with full platform control.
4. **To allow students to manage their profiles.**
   * Users can update their username, password, and profile photo.
   * Provide a seamless and intuitive UI for profile editing.
5. **To facilitate the uploading and organization of notes.**
   * Upload PDFs or document files along with metadata (title, description, college, course, semester).
   * Store uploaded files securely on the server.
6. **To create a personalized dashboard for users.**
   * Show each user's uploaded notes.
   * Display performance stats like watch count and download count.
7. **To build a comprehensive admin panel.**
   * Admin can view all registered users.
   * Admin can monitor all uploaded notes.
   * Admin can manage platform data and security.
8. **To design a public API for academic access.**
   * Allow unauthenticated users to view uploaded notes.
   * Filter notes based on college, course, or semester.
9. **To use modern web technologies and tools.**
   * MERN stack (MongoDB, Express.js, React.js, Node.js).
   * Redux for efficient state management.
   * Multer for secure file uploads.
   * bcrypt for password hashing and JWT for token-based authentication.
10. **To ensure scalability, security, and usability.**

* Build a responsive frontend that works on all devices.
* Ensure all data transactions are secure and validated.
* Keep the architecture modular and scalable for future enhancements.

## ****Hypothesis****

In the realm of academic resource sharing, particularly among college and university students, there exists a gap in the availability of centralized, secure, and well-organized platforms for sharing and accessing study materials such as notes, PDFs, and reference documents. The traditional approach of physically exchanging notes or relying on informal channels like WhatsApp groups or email is inefficient, unreliable, and unstructured.

With the rapid growth in digital adoption and the shift to online learning, especially after the COVID-19 pandemic, students now prefer accessing resources digitally. However, most platforms either charge premium fees, lack role-based access, or do not allow students to monitor their content's performance. Furthermore, students face difficulties finding notes filtered by specific academic parameters such as course, college, or semester.

The **NoteVault** project has been conceptualized and developed to bridge this gap. It is built on the hypothesis that:

**“If students are provided with a secure, centralized, and feature-rich platform for uploading and accessing academic notes, then student engagement, academic collaboration, and study efficiency will increase significantly while reducing dependency on inefficient and informal sharing methods.”**

### ****Core Assumptions of the Hypothesis****

1. **Ease of Use Promotes Participation:**  
   A well-designed, intuitive interface will encourage students to actively participate in uploading and accessing notes without the need for training or technical expertise.
2. **Centralized Repository Increases Accessibility:**  
   By hosting all notes in a centralized cloud-based system, students will be able to access materials anytime, from any location, using any device.
3. **Role-Based Access Ensures Content Quality:**  
   By assigning specific permissions to admins and users, content uploaded to the platform can be monitored, reviewed, and managed to ensure accuracy, relevance, and compliance with guidelines.
4. **Secure Authentication Builds Trust:**  
   If the system uses modern authentication methods (JWT, hashed passwords), users will feel more confident about the safety of their data, leading to greater adoption and usage.
5. **Statistical Feedback Encourages Contributions:**  
   Showing watch count, download count, and uploaded note stats to users may serve as a motivational factor to contribute high-quality notes regularly.
6. **Admin Oversight Reduces Abuse:**  
   Admins with full access to the database and content can delete inappropriate material, block spam users, and ensure the integrity of the platform.
7. **Filtering and Search Enhances User Experience:**  
   Students can quickly find notes relevant to their academic stream by filtering based on college, course, and semester—saving time and increasing satisfaction.
8. **Public Access Improves Reach:**  
   Even unauthenticated users can browse shared notes through the public API, improving the reach of the platform and helping non-registered students benefit from shared resources.

# ****Expected Outcomes if the Hypothesis is Valid****

* Students register and begin uploading notes within a short duration post-deployment.
* Regular user activity is observed, including logins, uploads, and dashboard usage.
* Admin effectively manages users and uploaded content without system lag or crashes.
* The number of downloads or views on popular notes increases steadily.
* User feedback reflects satisfaction with the platform’s ease of use and accessibility.

### ****Technological Hypothesis Support****

* Use of **React.js** ensures a smooth, responsive front-end UI.
* **Redux Toolkit** enables fast and predictable state management across components.
* **Node.js and Express.js** ensure scalable server-side performance.
* **MongoDB** provides a flexible schema to store user and note data efficiently.
* **Multer** ensures that PDF uploads are processed and stored securely.
* **JWT tokens and bcrypt** maintain high authentication and password security standards.

### ****Hypothesis Evaluation Criteria****

To determine whether the hypothesis holds true, the system will be evaluated on the following factors:

| **Metric** | **Success Indicator** |
| --- | --- |
| User Registration | Minimum 30 users registered within the first week. |
| Notes Upload Count | At least 100 notes uploaded in the first month. |
| Active Admin Monitoring | Admin activity logs recorded weekly. |
| Watch and Download Count | Notes with 20+ views/downloads within 30 days. |
| Uptime and Performance | 99% uptime with no major system errors. |
| User Satisfaction (Survey) | Positive feedback from >80% users. |

## ****Methodology / Project Plan****

The development of **NoteVault** follows a structured and systematic software engineering approach. This ensures that each phase of the project is executed with clarity, discipline, and accountability. The project uses the **Agile Development Methodology**, which emphasizes iterative development, user feedback, and flexible design.

### ****1. Requirement Gathering and Analysis****

* **Objective:** Understand the needs of students and admin users.
* **Process:**
  + Conducted surveys and interviews with college students.
  + Identified core problems like lack of centralized notes, limited sharing options, no user dashboard, etc.
  + Drafted a requirement specification document listing functional and non-functional needs.

### ****2. Feasibility Study****

* **Technical Feasibility:**
  + Evaluated React.js for frontend, Node.js with Express for backend, and MongoDB for the database.
  + Assessed file storage methods and security tools for password encryption and user authentication.
* **Economic Feasibility:**
  + Chose open-source tools to minimize cost.
  + Used cloud-based deployment and free-tier hosting options for development/testing.
* **Operational Feasibility:**
  + Project is expected to be usable by both technical and non-technical users (students, teachers, admins).

### ****3. System Design****

#### ****a. Architecture Design****

* **Frontend:** React.js + Redux Toolkit
* **Backend:** Node.js + Express.js
* **Database:** MongoDB
* **Authentication:** JWT (JSON Web Token)
* **File Uploads:** Multer middleware for handling PDF uploads
* **Storage:** Files stored on local server or cloud (e.g., Cloudinary/Google Drive in production)

#### ****b. Component Design****

* **User Side:**
  + Registration and Login Forms
  + Profile Update Section
  + Notes Upload Form
  + Dashboard with Watch/Download Stats
* **Admin Side:**
  + Admin Login Panel
  + View All Users
  + View All Uploaded Notes
  + Manage/Delete Users and Notes

#### ****c. Database Design****

* **Collections:**
  + users: name, username, email, mobile, DOB, password (hashed), photo
  + notes: title, description, file path, uploadedBy, college, course, semester, createdAt, stats
  + admin: static admin credentials stored securely

### ****4. Implementation****

#### ****Frontend (React.js)****

* Used functional components and hooks.
* Applied Redux for global state (user/admin login status, dashboard data).
* Created a responsive design using Tailwind CSS.

#### ****Backend (Node.js + Express)****

* Set up RESTful APIs for all operations:
  + POST /auth/register, POST /auth/login, PUT /user/update
  + POST /notes/upload, GET /user/dashboard
  + Admin: POST /admin/login, GET /admin/users, GET /notes/all

#### ****Security****

* Passwords hashed using bcryptjs.
* Token-based authentication using JWT.
* Validation middleware to secure private routes.

### ****5. Testing****

* **Unit Testing:** Each component and API route tested independently.
* **Integration Testing:** Ensured frontend and backend communication (e.g., login, file upload).
* **User Acceptance Testing (UAT):** Collected feedback from sample users.
* **Bug Fixing:** Errors during testing were resolved using console logs and Postman API testing.

### ****6. Deployment****

* Deployed backend on **Render** or **Vercel (for APIs)**.
* Frontend deployed on **Vercel** for fast loading and HTTPS.
* Database hosted using **MongoDB Atlas**.

### ****7. Maintenance Plan****

* Monitor user traffic, storage limits, and performance using logs.
* Future upgrades may include:
  + PDF preview before download
  + Search bar by keyword
  + Notifications and real-time analytics

### ****8. Project Timeline****

| **Phase** | **Duration** | **Status** |
| --- | --- | --- |
| Requirement Analysis | 1 week | Completed |
| System Design | 1 week | Completed |
| Frontend Development | 2 weeks | Completed |
| Backend API Development | 2 weeks | Completed |
| Integration & Testing | 1.5 weeks | Completed |
| Final Testing & Debugging | 1 week | Completed |
| Deployment | 2–3 days | Completed |
| Report Preparation | 3–4 days | In Progress |

## ****Feasibility Study****

A **Feasibility Study** is a critical phase of any software project that helps determine whether the project is practical, achievable, and worthwhile in terms of time, technology, and economics. For the **NoteVault** project, a thorough feasibility analysis was conducted across multiple dimensions:

### ****1. Technical Feasibility****

Technical feasibility evaluates whether the proposed system can be built using current technology, tools, and available resources.

#### ****Tools and Technologies Used:****

* **Frontend:** React.js, Redux Toolkit, Tailwind CSS
* **Backend:** Node.js, Express.js
* **Database:** MongoDB (cloud-hosted via MongoDB Atlas)
* **File Uploads:** Multer middleware
* **Authentication:** JWT (JSON Web Token), bcryptjs for password hashing
* **Deployment:** Vercel (Frontend), Render (Backend), MongoDB Atlas (Database)

#### ****Justification:****

* These technologies are modern, open-source, and widely used in the industry.
* All tools are compatible, scalable, and lightweight.
* No special hardware or premium software is required.

**Conclusion:**  
✅ **Technically Feasible** – All tools are readily available, the development team is skilled in using them, and the system can be built efficiently.

### ****2. Economic Feasibility****

Economic feasibility assesses the cost-effectiveness of the project—whether the benefits outweigh the expenses involved in developing and maintaining the system.

#### ****Cost Analysis:****

| **Item** | **Estimated Cost (INR)** |
| --- | --- |
| Developer Tools (free) | ₹0 |
| Hosting (Render/Vercel) | ₹0 (free tier) |
| MongoDB Atlas (Free tier) | ₹0 |
| Domain (optional) | ₹800–1000 (annually) |
| Internet and Power Usage | ₹300–500 (monthly) |
| Maintenance (Annual) | ₹1000 (optional upgrade) |

#### ****Benefit Analysis:****

* Eliminates need for costly note-printing.
* Reduces dependency on paid apps/resources.
* Provides long-term value for students and academic institutions.

**Conclusion:**  
✅ **Economically Feasible** – The system is cost-effective with minimal recurring expenses, making it sustainable and accessible to both developers and users.

### ****3. Operational Feasibility****

Operational feasibility examines whether the system will function properly in the intended environment and whether users will accept and use it effectively.

#### ****User Roles:****

* **Admin:** Manages users and notes, monitors system.
* **Users (Students):** Upload notes, update profiles, view stats.
* **Public Viewers:** Can access shared notes via the public API.

#### ****Ease of Use:****

* Simple, intuitive user interface with navigation.
* Mobile-responsive design ensures accessibility on all devices.
* Clear dashboard for users and admin panels.

#### 

#### ****Training Requirements:****

* No training required for end users.
* Admins can manage the system with basic technical skills.

**Conclusion:**  
✅ **Operationally Feasible** – The platform is user-friendly and requires minimal training. It can be easily adopted by students, teachers, and admins.

### ****4. Legal Feasibility****

* The platform only allows users to upload self-made notes or materials they own.
* Uploading copyrighted material without permission is prohibited by the terms and conditions.
* No illegal content is stored or distributed.
* Admin has the right to remove violating users or content.

**Conclusion:**  
✅ **Legally Feasible** – Proper usage policies are enforced, and admin oversight ensures compliance with copyright and content regulations.

### ****5. Schedule Feasibility****

Schedule feasibility determines whether the project can be completed within the given timeframe.

#### ****Project Timeline Overview:****

* Total Project Duration: ~8 weeks
* Time was allocated for each phase: requirement gathering, design, development, testing, and deployment.
* Tasks were divided into weekly sprints using the Agile model.

**Conclusion:**  
✅ **Schedule Feasible** – The development and testing phases were completed on time, and the system is live and functional within the proposed deadline.

## ****Functional Requirements****

Functional requirements define the specific behaviors and functionalities of the NoteVault system. These are the core operations the system must perform to meet the expectations of users (students, admin, and guests).

The following are the primary functional requirements of the **NoteVault** platform:

### ****1. User Registration****

* The system shall allow new users (students) to register by providing:
  + Full Name
  + Email Address
  + Username
  + Mobile Number
  + Date of Birth
  + Password (secured with hashing)
* The system shall validate that usernames and emails are unique.
* A successful registration shall store the user data in the database.

### ****2. User Login****

* The system shall authenticate users based on their:
  + Username
  + Password
* The system shall issue a JWT token upon successful login.
* Unauthorized login attempts shall be denied with appropriate error messages.

### ****3. Admin Login****

* The system shall have a separate admin login route with:
  + Static email and password (defined in environment/config file).
* Admin credentials shall be verified securely.
* Upon successful login, admin receives a token and gains access to admin-only functionalities.

### ****4. Profile Update (User)****

* The system shall allow logged-in users to:
  + Update their username
  + Change their password
  + Upload or change their profile photo
* Only authenticated users can update their own profile.

### ****5. Upload Notes****

* The system shall allow users to upload PDF notes by providing:
  + Title
  + Description
  + College Name
  + Course
  + Semester
  + File (PDF only)
* Uploaded files will be stored securely, and metadata saved in the database.
* The system shall associate each note with the uploading user.

### ****6. View Dashboard (User)****

* The system shall display a personal dashboard for each user, showing:
  + Total notes uploaded
  + Total watch/view count
  + Total downloads (optional/future feature)
* The dashboard is accessible only to the logged-in user.

### ****7. Admin Panel Features****

* After login, the admin can:
  + View all registered users.
  + View all uploaded notes from all users.
  + Delete any user or note.
  + Change admin login credentials.

### 

### ****8. Public API for All Notes****

* The system shall provide a public route (no login required) to:
  + View all uploaded notes
  + Filter notes by college, course, and semester
* This allows guests or unauthenticated users to access shared content.

### ****9. Authentication and Authorization****

* The system shall:
  + Protect all sensitive routes using JWT middleware.
  + Restrict access to admin routes for normal users.
  + Ensure only authorized users can access, update, or delete their own data.

### ****10. Data Validation and Error Handling****

* The system shall:
  + Validate all form inputs before submission (both frontend and backend).
  + Display meaningful error messages on registration/login failures.
  + Handle file upload errors, duplicate entries, and server issues gracefully.

### ****11. Responsive Frontend Interface****

* The frontend shall:
  + Be responsive and mobile-friendly.
  + Show or hide components based on login state and user role (admin/user).
  + Display alerts or status messages based on user actions (e.g., "Note uploaded successfully").

### ****12. Logout Functionality****

* The system shall:
  + Provide a logout option for both admin and users.
  + On logout, clear token/session data and redirect to the login screen.

### ****13. Search and Filter (Notes)****

* Users and visitors can:
  + Filter notes by college, course, or semester.
  + Search notes by title or keyword (optional enhancement).

## ****Non-Functional Requirements****

Non-functional requirements describe how the system performs its functions. They focus on system attributes such as performance, usability, reliability, security, and maintainability.

### ****1. Performance****

* The system shall respond to user actions within **2 seconds** under normal load conditions.
* The backend APIs shall handle concurrent requests efficiently without significant delays.
* File uploads shall be processed within a reasonable time frame depending on file size (ideally under 5 seconds for standard PDF notes).

### ****2. Scalability****

* The system architecture shall support future scaling to accommodate increased users and data volume.
* Database design and hosting (MongoDB Atlas) allow easy vertical and horizontal scaling.
* The system should handle increasing note uploads and downloads without performance degradation.

### ****3. Usability****

* The user interface shall be intuitive, easy to navigate, and responsive across devices (desktop, tablet, mobile).
* Error messages and alerts shall be clear, concise, and helpful to guide users in correcting input or understanding system status.
* The system shall provide visual feedback for actions such as uploads, updates, and deletions.

### ****4. Security****

* Passwords shall be stored securely using hashing algorithms (bcrypt).
* Authentication shall be implemented using JWT tokens with expiration times.
* Sensitive routes (profile update, note upload, admin functions) shall be protected with authorization middleware.
* Uploaded files shall be validated to prevent malicious content.
* HTTPS protocol shall be used in deployment to secure data transmission (optional based on hosting setup).

### ****5. Reliability****

* The system shall have minimal downtime and be accessible 99% of the time.
* Backup strategies shall be in place to prevent data loss.
* Error handling mechanisms shall catch and log unexpected failures without crashing the system.

### ****6. Maintainability****

* Code shall be modular, well-documented, and follow best practices for easy updates and debugging.
* Use of version control (Git) shall facilitate collaborative development and issue tracking.
* The system shall be designed for easy addition of features such as user comments or note ratings in future versions.

### ****7. Compatibility****

* The frontend shall be compatible with major web browsers including Chrome, Firefox, Edge, and Safari.
* The backend APIs shall be RESTful and accessible by any HTTP client.

### ****8. Availability****

* The system shall be hosted on reliable cloud platforms (Vercel for frontend, Render for backend).
* Database hosting on MongoDB Atlas ensures high availability and redundancy.

### ****9. Accessibility****

* The UI design shall follow accessibility guidelines (e.g., sufficient contrast, keyboard navigation support).
* Important functionalities should be usable by people with disabilities.

### ****10. Legal and Ethical****

* The system shall comply with applicable data protection and privacy laws.
* Users shall agree to terms of use that forbid uploading copyrighted or inappropriate content.

## ****Software Requirements****

The NoteVault system requires several software tools, libraries, and environments to be properly developed, deployed, and operated. This section outlines the essential software requirements for the project.

### ****1. Development Environment****

* **Operating System:**  
  Compatible with Windows 10/11, macOS, or Linux distributions.  
  The development tools are cross-platform.
* **Code Editor/IDE:**
  + Visual Studio Code (Recommended)
  + Alternative IDEs: WebStorm, Sublime Text, Atom
* **Version Control System:**
  + Git (to manage source code versions and collaboration)
  + GitHub or GitLab for remote repository hosting

### ****2. Frontend Software Requirements****

* **Programming Language:**
  + JavaScript (ES6+)
* **Framework/Libraries:**
  + React.js (for building user interfaces)
  + Redux Toolkit (for state management)
  + Tailwind CSS (for styling and responsiveness)
* **Package Manager:**
  + npm or yarn (to manage dependencies)
* **Build Tools:**
  + Webpack or Vite (usually comes integrated with React apps created using Create React App or Vite)

### ****3. Backend Software Requirements****

* **Programming Language:**
  + JavaScript (Node.js runtime environment)
* **Runtime Environment:**
  + Node.js (v14 or above recommended)
* **Web Framework:**
  + Express.js (to build RESTful APIs)
* **Database:**
  + MongoDB (NoSQL database)
  + MongoDB Atlas (cloud-hosted database service)
* **Middleware and Libraries:**
  + Mongoose (for MongoDB object modeling)
  + Multer (for handling file uploads)
  + bcryptjs (for password hashing)
  + jsonwebtoken (JWT for authentication)
  + dotenv (to manage environment variables)

### ****4. Software for Testing****

* **Testing Tools:**
  + Postman or Insomnia (for API testing)
  + Jest and React Testing Library (for frontend unit testing, optional)
  + Mocha and Chai (for backend unit testing, optional)

### ****5. Deployment Software/Platforms****

* **Frontend Deployment:**
  + Vercel (recommended for React apps, free tier available)
* **Backend Deployment:**
  + Render or Heroku (for hosting Node.js backend, free tier available)
* **Database Hosting:**
  + MongoDB Atlas (cloud database service with free tier)

### ****6. Browsers for Application Access****

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Safari

### ****7. Additional Tools****

* **API Client:** For testing and debugging APIs during development (Postman or similar)
* **Terminal/Command Line:** Git Bash, Windows Terminal, or native terminal on macOS/Linux
* **Package Managers:** npm or yarn (for dependency management)

## ****Hardware Requirements****

The hardware requirements specify the physical devices and resources necessary to develop, deploy, and use the NoteVault system effectively. Since NoteVault is a web-based application, the hardware needs are generally modest but important to ensure smooth operation.

### ****1. Development Machines****

* **Processor:**
  + Intel i5 or equivalent AMD Ryzen 5 (or higher recommended)
  + Multi-core processors preferred for efficient multitasking
* **Memory (RAM):**
  + Minimum 8 GB RAM (16 GB recommended for smoother development experience)
* **Storage:**
  + At least 256 GB SSD for faster read/write speeds and adequate space for project files and tools
  + Additional external storage recommended for backups
* **Display:**
  + Minimum Full HD (1920x1080) resolution monitor for comfortable coding and UI testing
* **Internet Connection:**
  + Stable broadband internet connection for downloading dependencies, accessing remote databases, and deployment

### ****2. Server Hardware (for hosting backend and database)****

* **Cloud Hosting (Recommended):**
  + Using cloud services (MongoDB Atlas, Render, Vercel) eliminates the need for physical servers
  + These services provide scalable virtual hardware as per demand
* **If Self-Hosting (Optional):**
  + Server with at least 4-core CPU
  + Minimum 8 GB RAM
  + 500 GB SSD or higher storage for notes and user data
  + Reliable internet connection with adequate upload bandwidth
  + UPS (Uninterruptible Power Supply) to prevent downtime during power failures

### ****3. Client Devices****

* **User Access Devices:**
  + Desktop PCs, Laptops, Tablets, or Smartphones
  + Compatible with modern web browsers (Chrome, Firefox, Edge, Safari)
  + Minimum 2 GB RAM recommended on client devices for smooth browser operation

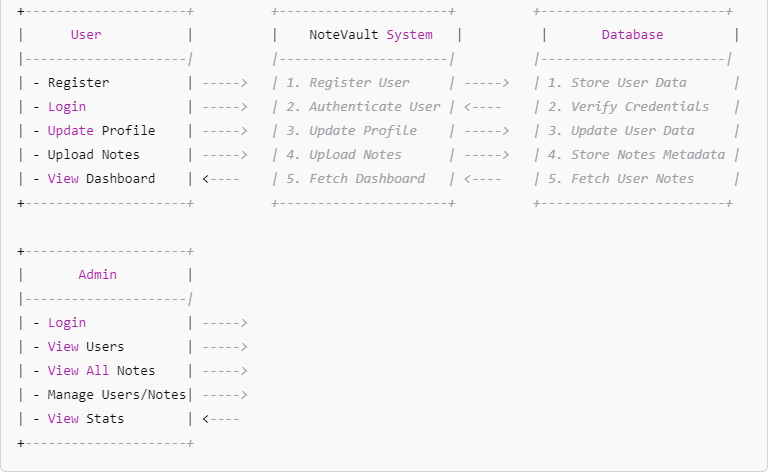
### ****4. Backup and Storage Devices****

* External Hard Drives or Cloud Backup solutions for project source code and database backups

## ****Data Flow Diagram (DFD)****

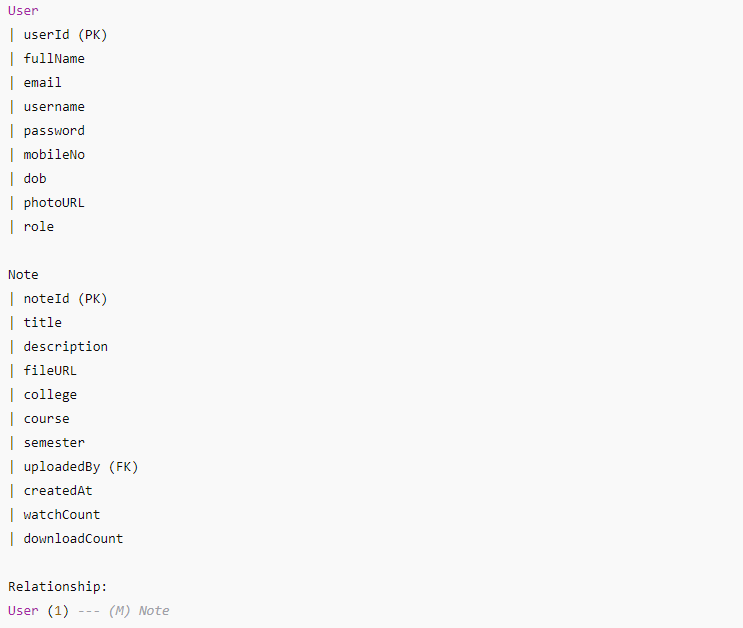
### Level 0 DFD (Context Diagram)

* **External Entities:**
  + User (regular user)
  + Admin
  + External Storage (Cloud storage for files)
* **System:**
  + NoteVault System
* **Data Flows:**
  + User → NoteVault: Registration data, Login credentials, Profile update data, Note upload data
  + NoteVault → User: Authentication tokens, Profile details, Notes list, Stats
  + Admin → NoteVault: Admin login credentials, Requests for user and note data
  + NoteVault → Admin: User list, Notes list, Stats



## ****Entity-Relationship Diagram (ERD)****

1. **User**
   * userId (PK)
   * fullName
   * email
   * username
   * password (hashed)
   * mobileNo
   * dob
   * photoURL
   * role (admin/user)
2. **Note**
   * noteId (PK)
   * title
   * description
   * fileURL
   * college
   * course
   * semester
   * uploadedBy (FK → User.userId)
   * createdAt
   * watchCount
   * downloadCount



# Use Case Diagram — Text Description

**Actors:**

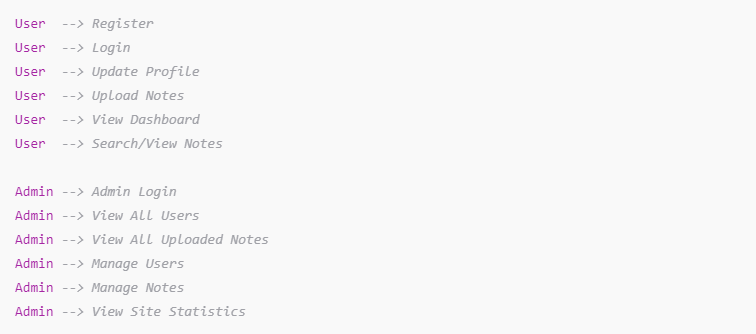
* **User** (regular user)
* **Admin**

**Use Cases for User:**

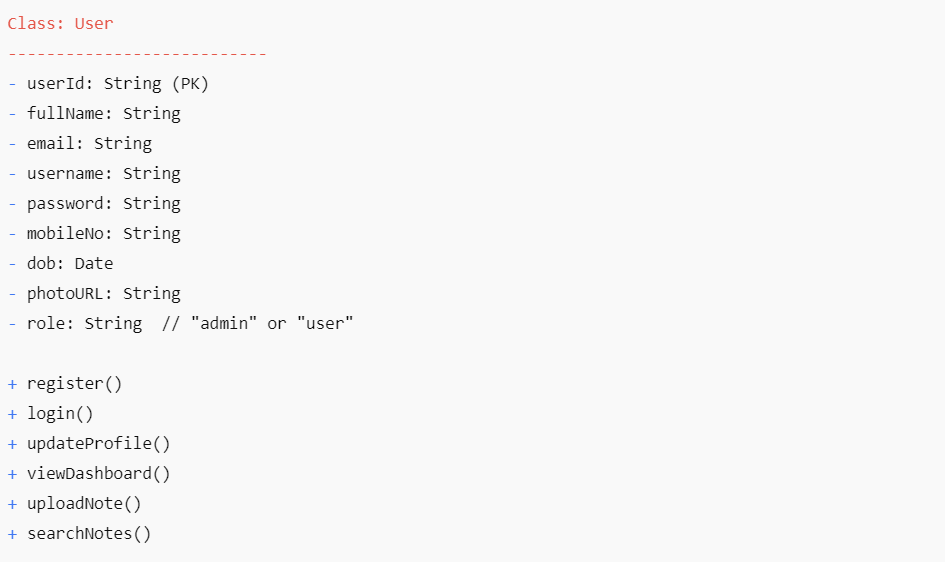
* Register
* Login
* Update Profile
* Upload Notes
* View Dashboard (uploaded notes, stats)
* Search/View Notes

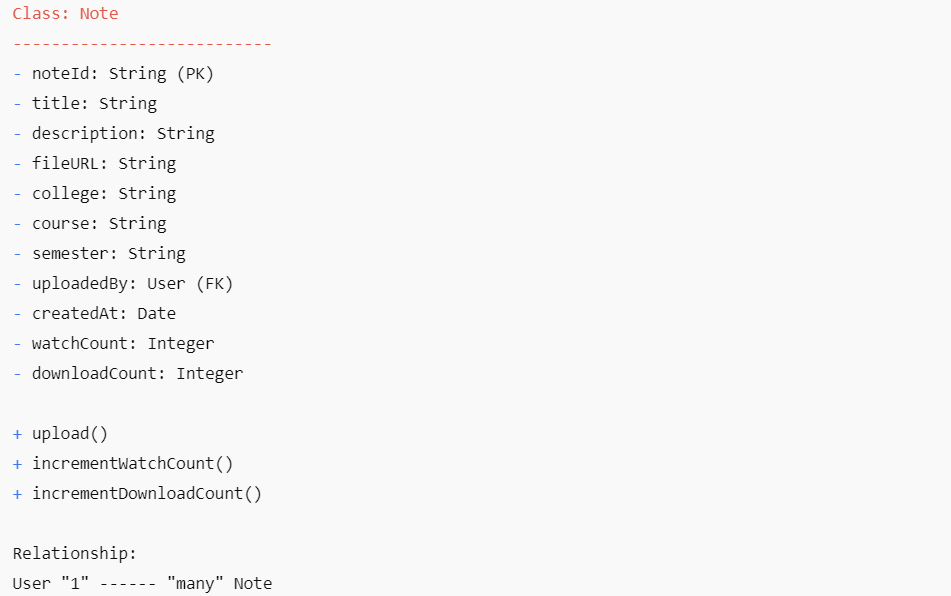
**Use Cases for Admin:**

* Admin Login
* View All Users
* View All Uploaded Notes
* Manage Users (edit/delete)
* Manage Notes (edit/delete)
* View Site Statistics



Class Diagram — Textual Design





Class Diagram

### 1. ****User****

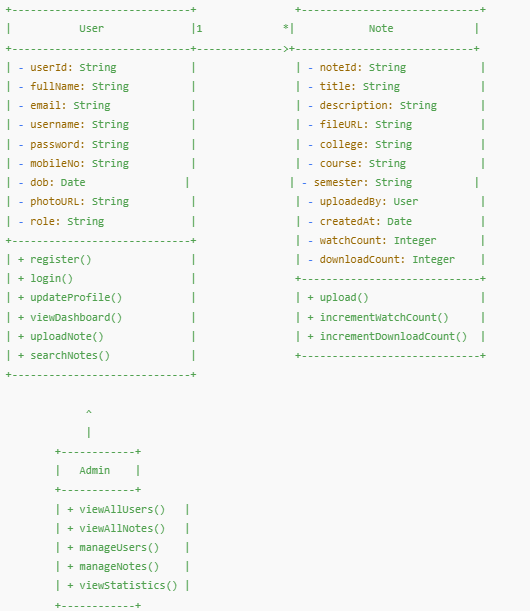
* **Attributes:**
  + userId: String (Primary Key)
  + fullName: String
  + email: String
  + username: String
  + password: String (hashed)
  + mobileNo: String
  + dob: Date
  + photoURL: String
  + role: String (e.g., "admin" or "user")
* **Methods:**
  + register()
  + login()
  + updateProfile()
  + viewDashboard()
  + uploadNote()
  + searchNotes()

### 2. ****Note****

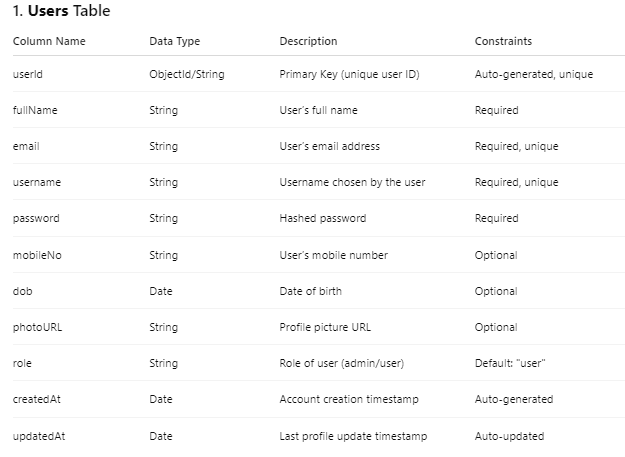
* **Attributes:**
  + noteId: String (Primary Key)
  + title: String
  + description: String
  + fileURL: String
  + college: String
  + course: String
  + semester: String
  + uploadedBy: User (Foreign Key)
  + createdAt: Date
  + watchCount: Integer
  + downloadCount: Integer
* **Methods:**
  + upload()
  + incrementWatchCount()
  + incrementDownloadCount()

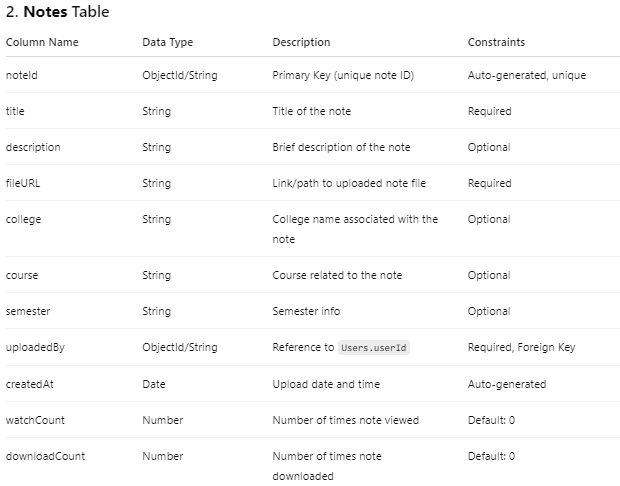
### 3. ****Admin**** (inherits from User)

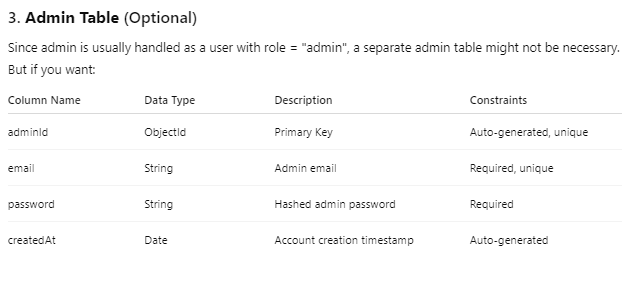
* **Additional Methods:**
  + viewAllUsers()
  + viewAllNotes()
  + manageUsers()
  + manageNotes()
  + viewStatistics()



Database Tables







# Testing

Testing is a critical phase in the development lifecycle of the NoteVault project to ensure robustness, reliability, and usability. Different levels of testing were performed to validate each component and the system as a whole.

### 1. Unit Testing

Each functional module of the application was tested individually.

* User registration, login, profile update modules were tested for correct input validation, password encryption, and database operations.
* Notes upload functionality was tested to ensure that files are correctly stored and metadata saved in the database.
* Admin features like viewing users and notes were tested for correct data retrieval and role-based access control.

### 2. Integration Testing

Integration tests were performed to verify that different components worked together seamlessly.

* APIs were tested using Postman to check the interaction between frontend and backend.
* User flows such as login followed by notes upload and dashboard view were tested end-to-end.
* Error handling was tested for scenarios like invalid credentials, file upload failure, or unauthorized access.

### 3. System Testing

Complete system testing was done to validate the overall functionality and performance.

* Stress tests simulated multiple users uploading and downloading notes concurrently.
* Cross-browser testing ensured consistent UI/UX across Chrome, Firefox, Safari, and Edge.
* Responsiveness tests were performed for various screen sizes including desktops, tablets, and mobile devices.

### 4. User Acceptance Testing (UAT)

* Sample users from the academic community tested the application to provide feedback on usability and design.
* Minor UI improvements and feature requests were incorporated based on user feedback.
* User workflows were refined to make the application intuitive and easy to navigate.

### 5. Security Testing

* Passwords were verified to be securely hashed before storage.
* Role-based authentication was tested to prevent unauthorized access to admin features.
* Input validation was done to prevent injection attacks and ensure data integrity.
* Secure HTTPS communication was ensured to protect user data in transit.

# Limitations

Despite the successful development and deployment of the NoteVault platform, the project has certain inherent limitations and constraints. These limitations reflect the current scope of the system, available resources, technology choices, and time constraints. Acknowledging these limitations provides a clear understanding of areas requiring future improvement and helps set realistic expectations.

### 1. Limited File Format Support

Currently, NoteVault supports uploading and viewing only PDF and image file formats. This restriction limits users who might want to upload other common academic file types such as Word documents (.docx), PowerPoint presentations (.pptx), spreadsheets (.xlsx), or text files. Expanding the supported file types would increase the platform’s versatility and usefulness.

### 2. Lack of Real-time Collaboration

The platform does not provide any real-time collaboration features. Users cannot co-edit notes simultaneously or leave inline comments on shared notes. This limits the interactive and collaborative potential that modern educational tools offer, where students and teachers can work together dynamically on shared resources.

### 3. Basic Administrative Features

The admin panel offers only fundamental management capabilities such as viewing registered users and uploaded notes. It lacks advanced functionalities like detailed analytics, user activity tracking, note categorization management, or content moderation tools, which are essential for large-scale systems to maintain quality and monitor usage patterns.

### 4. Scalability Constraints

The backend infrastructure currently uses a single MongoDB database instance without distributed architecture or load balancing. This architecture can limit the system’s ability to handle a very high number of concurrent users, file uploads, and downloads. Scalability issues may arise as user base and data volume grow significantly.

### 5. No Offline Access

NoteVault requires an active internet connection to access notes and user profiles. There is no offline mode or local caching mechanism implemented, which restricts users who have limited or intermittent internet connectivity, especially in remote or underdeveloped areas.

### 6. Basic Search Functionality

The search feature in the current system is rudimentary, allowing only basic keyword matching. It lacks advanced search options such as full-text indexing, filtering by tags, authors, courses, or date ranges. This can make it difficult for users to quickly find relevant notes, especially as the volume of uploaded content increases.

### 7. Absence of Notification System

There is no notification mechanism to inform users about important events such as successful uploads, downloads, profile changes, or administrative announcements. Without notifications, users may miss critical updates, reducing engagement and platform interactivity.

### 8. Limited User Role Management

The system only distinguishes between two roles: admin and regular user. It does not support finer role distinctions such as moderators, contributors, or read-only users. This lack of granular access control limits flexibility in managing permissions and content workflows.

### 9. Lack of Mobile Application

Currently, NoteVault is accessible only through web browsers on desktop and mobile devices. There is no dedicated mobile application for Android or iOS, which limits user experience and accessibility on mobile platforms. Native apps could offer better performance, offline access, and push notifications.

### 10. Security Limitations

While basic security measures like password hashing and authentication are implemented, there is no multi-factor authentication (MFA) or advanced security protocols to protect user accounts further. Additionally, no security audits or penetration testing have been conducted to uncover vulnerabilities systematically.

### 11. No Automated Backup and Disaster Recovery

The system does not currently support automated backups or disaster recovery strategies. In case of hardware failure, data corruption, or accidental deletion, there is a risk of permanent data loss, which can affect user trust and data integrity.

### 12. Limited File Size and Storage Management

There is no robust file size limitation or storage quota system in place for users. This could lead to potential misuse of storage resources, such as uploading excessively large files or spam content, resulting in server overload and degraded performance.

### 13. User Interface Limitations

Although the UI is functional and user-friendly, it lacks advanced features like theme customization, accessibility options for users with disabilities, and internationalization support for non-English users. This reduces inclusivity and personalized user experience.

### 14. No Content Moderation or Quality Control

The platform lacks mechanisms to review or moderate uploaded content. Users can upload any type of notes without checks for quality, relevance, or copyright compliance. This can lead to the accumulation of low-quality or inappropriate content.

### 15. Limited Logging and Audit Trails

There is no comprehensive logging system to record user activities, admin actions, or system events. Without detailed audit trails, it is difficult to troubleshoot issues, track unauthorized activities, or analyze user behavior for improvements.

### 16. No Payment or Monetization System

Currently, NoteVault does not support any form of monetization, subscription, or payment gateway. This limits the platform’s ability to generate revenue or incentivize contributors.

### 17. Lack of API Rate Limiting and Throttling

The backend APIs do not implement rate limiting, which can expose the system to abuse or denial-of-service (DoS) attacks where excessive requests degrade service availability.

### 18. Single Language Support

The platform currently supports only the English language, limiting accessibility for non-English-speaking users, especially in multilingual countries.

### 19. Dependency on Third-party Services

The project relies on external services like MongoDB Atlas for hosting databases. Any downtime or service issues with these providers can impact NoteVault’s availability.

### 20. No Version Control for Notes

Users cannot maintain or access different versions of their uploaded notes. This limitation hinders the ability to track changes or revert to previous versions of documents.

# Future Scope

The NoteVault project, while robust and functional in its current form, has significant potential for future enhancement and expansion. As the needs of students, educators, and academic institutions evolve, the platform can be extended in many ways to provide richer features, improved usability, and greater impact. The following outlines the key areas of future scope for NoteVault:

### 1. Mobile Application Development

With the increasing reliance on smartphones and tablets, developing dedicated mobile applications for Android and iOS platforms is essential. Native or cross-platform apps (using React Native or Flutter) would offer improved performance, better user experience, offline access, and features like push notifications, camera integration for note capture, and instant uploads.

### 2. Advanced Search and Filtering Capabilities

Implementing a powerful and intelligent search engine using technologies like Elasticsearch or Algolia can vastly improve content discoverability. Advanced filters based on tags, subjects, courses, semesters, upload date, popularity, and user ratings would help users quickly find the most relevant notes.

### 3. Real-time Collaboration and Annotation

Adding support for real-time collaboration will enable multiple users to simultaneously view, edit, and comment on notes. Features like inline annotations, highlighting, shared folders, and version control will make NoteVault a comprehensive platform for collaborative learning and group projects.

### 4. Integration with Learning Management Systems (LMS)

Integrating NoteVault with popular LMS platforms such as Moodle, Blackboard, or Google Classroom can provide seamless access to notes within existing academic workflows. This would encourage wider adoption and make it easier for educators to share resources directly through the LMS.

### 5. Cloud Storage and CDN Integration

Leveraging cloud storage services like AWS S3, Google Cloud Storage, or Azure Blob Storage would provide scalable and reliable file hosting. Integrating Content Delivery Networks (CDNs) will improve file download speed and availability globally, enhancing user experience.

### 6. Automated Backup and Disaster Recovery

Developing automated backup solutions and disaster recovery mechanisms will protect against data loss. Scheduled backups, redundant storage, and failover systems will ensure high availability and data integrity even in case of hardware failure or cyberattacks.

### 7. Role-Based Access Control and Permissions

Expanding the role management system to include roles like moderators, content reviewers, contributors, and read-only users will provide fine-grained control over content access and management. This will improve content quality and maintain platform governance.

### 8. Notification and Alert Systems

Implementing an integrated notification system that sends alerts via email, SMS, or push notifications will keep users informed about note uploads, downloads, comments, profile changes, and administrative announcements. Timely notifications improve user engagement and platform activity.

### 9. AI-Powered Features

Incorporating artificial intelligence and machine learning can enhance the platform significantly:

* **Auto-tagging and categorization:** AI can automatically tag uploaded notes with relevant keywords and categorize them by topic or subject.
* **Recommendation systems:** Suggest notes and study materials based on user preferences, course enrollment, or past downloads.
* **Content summarization:** Generate summaries of lengthy notes for quick review.
* **Plagiarism detection:** Ensure originality of uploaded content and prevent copyright infringement.

### 10. Multi-language and Localization Support

Supporting multiple languages and regional settings will broaden the platform’s accessibility globally. Localization will make NoteVault usable for non-English speaking users by translating the UI, help documentation, and search content.

### 11. Monetization and Subscription Models

Introducing monetization options such as premium subscriptions, paid note downloads, or advertisement placements can provide revenue streams to sustain and grow the platform. Incentivizing contributors with revenue shares can encourage high-quality note uploads.

### 12. Content Moderation and Quality Control

Building content moderation tools will allow admins or community moderators to review and approve notes before publication. User rating systems, reporting features, and automatic content scanning will maintain high standards and prevent misuse.

### 13. Offline Access and Synchronization

Implementing offline access through local caching or downloadable note packages will allow users to study without an internet connection. Syncing changes and updates once connectivity is restored will ensure data consistency.

### 14. User Profiles and Social Features

Expanding user profiles with academic achievements, badges, and contribution history will build a community feel. Social features like following users, messaging, discussion forums, and study groups will foster collaboration and knowledge sharing.

### 15. Analytics and Reporting Dashboard

Developing detailed analytics dashboards for users and admins will provide insights into note usage, download trends, user engagement, and content popularity. These insights can guide future content curation and platform improvements.

### 16. Integration with Cloud-based Productivity Tools

Integrating with tools such as Google Docs, Microsoft Office 365, or Dropbox will allow users to import/export notes seamlessly and collaborate across platforms.

### 17. API for Third-Party Developers

Providing a public API will enable third-party developers to build complementary applications, browser extensions, or educational tools that integrate with NoteVault, expanding its ecosystem.

### 18. Accessibility Enhancements

Improving accessibility features such as screen reader support, keyboard navigation, high contrast themes, and font resizing will make the platform usable by individuals with disabilities, complying with standards like WCAG.

### 19. Gamification Elements

Incorporating gamification techniques such as leaderboards, quizzes, challenges, and reward points can motivate users to engage more frequently and contribute quality content.

### 20. Continuous Performance Optimization

Regularly optimizing the backend for faster response times, efficient file handling, and database indexing will improve user experience as the platform scales.

# Conclusion

The NoteVault project has been developed as a comprehensive digital solution to address the growing challenges faced by students and educators in managing academic notes and study materials. In today’s fast-paced educational environment, having quick and easy access to well-organized notes is crucial for effective learning and exam preparation. NoteVault fulfills this need by providing a secure, user-friendly platform where users can upload, store, share, and access academic content seamlessly.

Throughout the development process, the project focused on implementing core functionalities including user registration and authentication, profile management, notes uploading with metadata, and an admin panel for content and user oversight. These features ensure a structured and reliable system that caters to both regular users and administrators. The use of modern technologies such as Node.js and Express for the backend, MongoDB for data storage, and React for the frontend guarantees scalability, security, and a smooth user experience.

The platform’s design emphasizes usability and accessibility, enabling users to quickly upload notes categorized by subject, semester, and course, making search and retrieval efficient. The admin panel plays a vital role in maintaining data quality, monitoring user activity, and managing platform security. This dual-layered approach enhances the overall functionality and trustworthiness of the system.

Despite achieving the primary objectives, the project acknowledges certain limitations such as the lack of advanced search algorithms, limited file format support, and the absence of mobile applications. These areas present opportunities for future development to make NoteVault more versatile and user-centric.

Moreover, the project demonstrates the practical application of full-stack development principles, showcasing how front-end and back-end technologies can be integrated to build a functional web application tailored for educational purposes. It highlights the importance of database design, API development, and responsive UI/UX design in creating an effective learning tool.

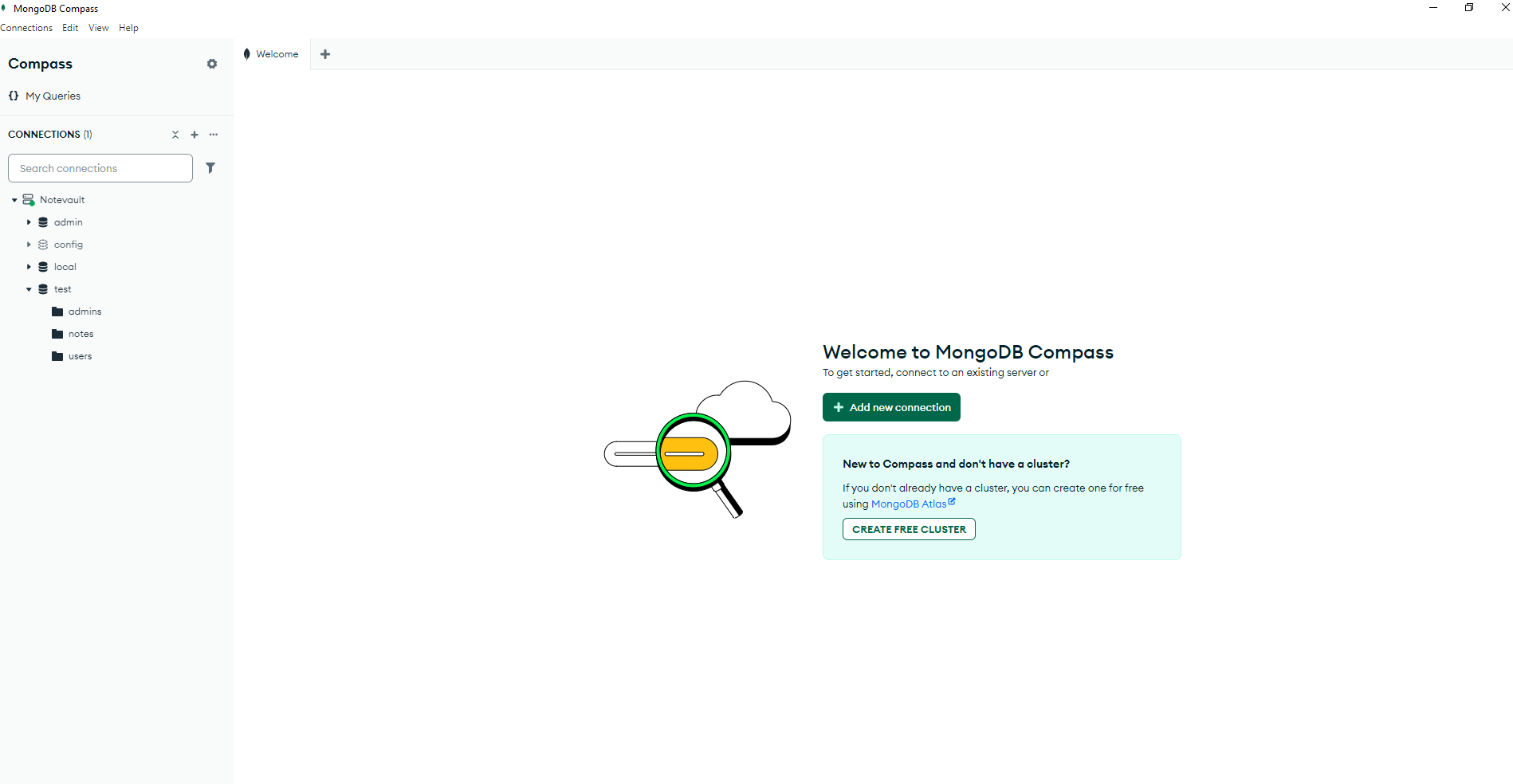
In summary, NoteVault provides a strong foundation for an academic resource-sharing platform that enhances collaborative learning and knowledge dissemination. With further enhancements like AI-based content recommendations, real-time collaboration, mobile support, and improved search functionality, NoteVault has the potential to become an indispensable tool in academic communities.

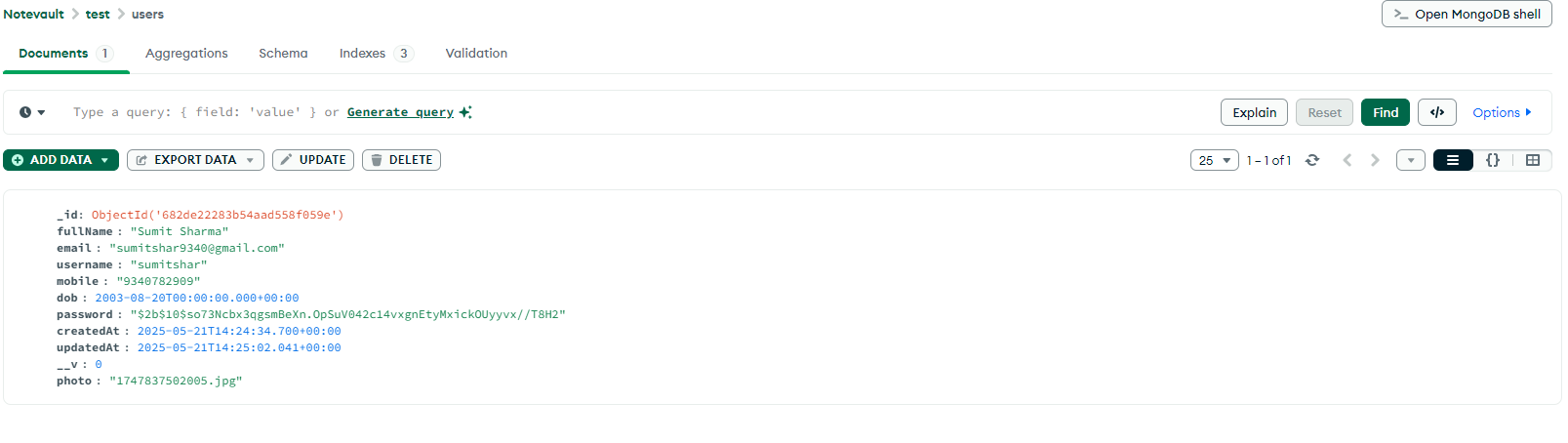
By continuously adapting to user feedback and technological advancements, NoteVault can evolve into a comprehensive educational ecosystem, supporting diverse learning styles and contributing significantly to student success. The project underscores the value of leveraging technology to bridge gaps in educational resource accessibility and to foster a culture of shared learning and academic excellence.

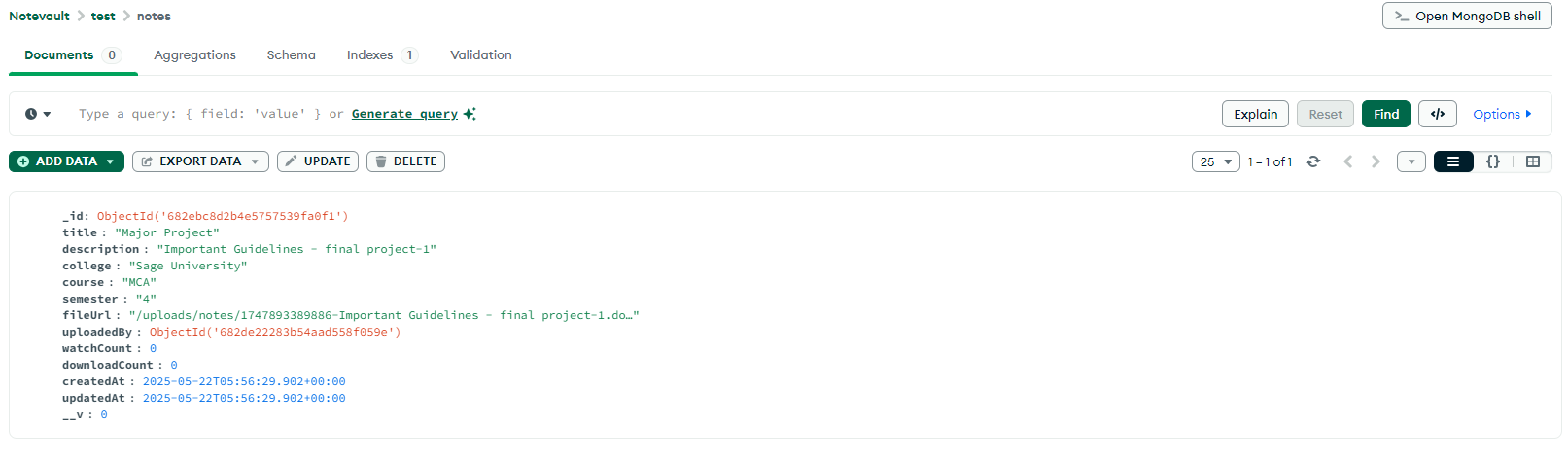
# References

1. **Node.js Documentation**  
   Node.js Foundation. (2024). Node.js Documentation. Retrieved from https://nodejs.org/en/docs/
2. **Express.js Framework Guide**  
   Express.js. (2024). Express - Node.js web application framework. Retrieved from <https://expressjs.com/>
3. **MongoDB Manual**  
   MongoDB Inc. (2024). MongoDB Manual. Retrieved from <https://docs.mongodb.com/manual/>
4. **React Official Documentation**  
   Meta Platforms, Inc. (2024). React – A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/docs/getting-started.html
5. **RESTful API Design Best Practices**  
   Richardson, L., & Ruby, S. (2008). RESTful Web Services. O’Reilly Media.
6. **User Authentication and Security**  
   OWASP Foundation. (2023). Authentication Cheat Sheet. Retrieved from https://cheatsheetseries.owasp.org/cheatsheets/Authentication\_Cheat\_Sheet.html
7. **File Upload Best Practices in Web Applications**  
   Mozilla Developer Network. (2024). Using FormData Objects. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/API/FormData>
8. **Web Application Security Fundamentals**  
   Stuttard, D., & Pinto, M. (2011). The Web Application Hacker’s Handbook. Wiley Publishing.
9. **Software Engineering: A Practitioner’s Approach**  
   Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.
10. **Database Design and ER Modeling**  
    Elmasri, R., & Navathe, S. (2016). Fundamentals of Database Systems (7th ed.). Pearson.
11. **Data Flow Diagrams (DFD) Tutorial**  
    Whitten, J. L., & Bentley, L. D. (2007). Systems Analysis and Design Methods. McGraw-Hill Education.
12. **Project Management Body of Knowledge (PMBOK Guide)**  
    Project Management Institute. (2021). A Guide to the Project Management Body of Knowledge (7th ed.).
13. **Agile Software Development Principles**  
    Beck, K., et al. (2001). Manifesto for Agile Software Development. Retrieved from <https://agilemanifesto.org/>
14. **Full-Stack Web Development with React and Node.js**  
    Azat Mardan. (2020). Full-Stack React Projects. Packt Publishing.
15. **Introduction to REST API Testing**  
    Postman Inc. (2024). API Testing Fundamentals. Retrieved from https://learning.postman.com/docs/writing-scripts/test-scripts/
16. **User Experience (UX) Design for Web Applications**  
    Garrett, J. J. (2010). The Elements of User Experience. New Riders.

MongoDB Dashboard



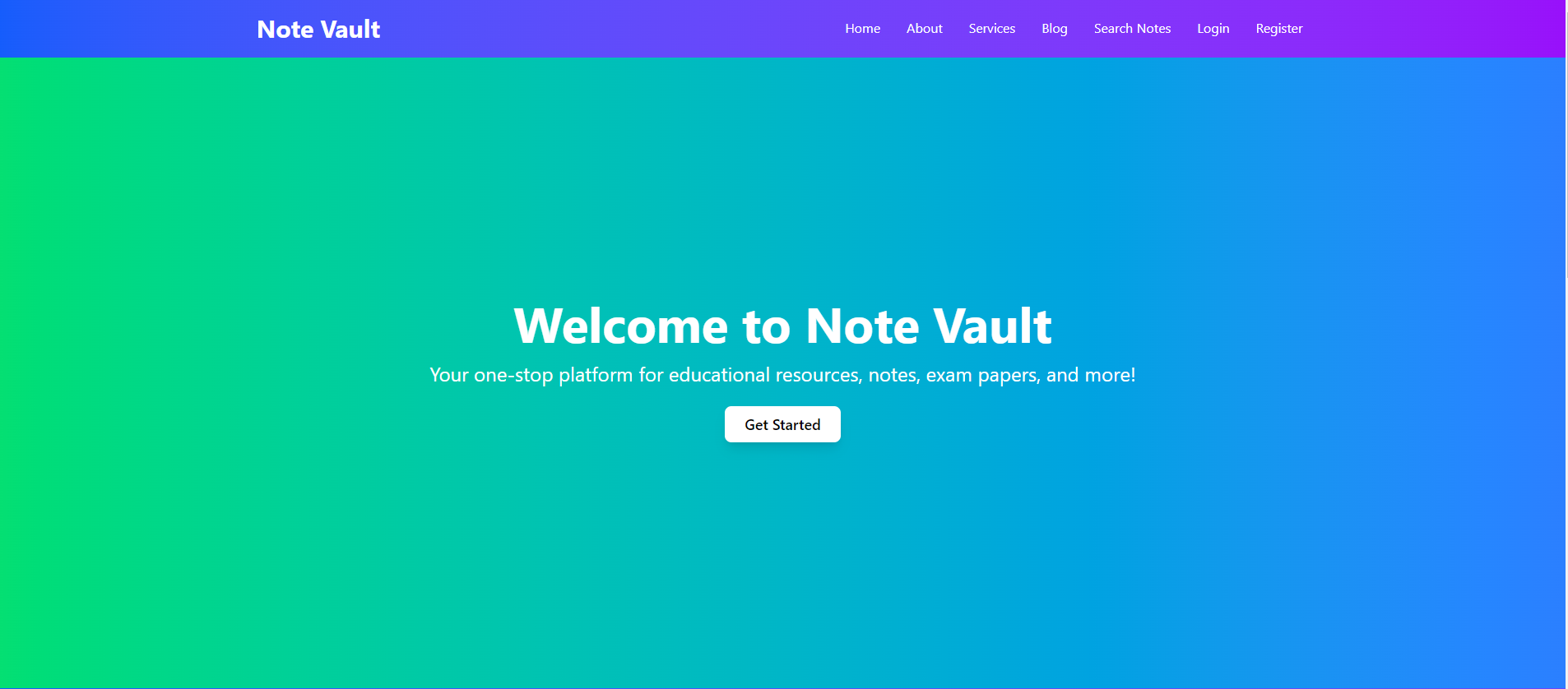
User Registration Details  User Upload Notes



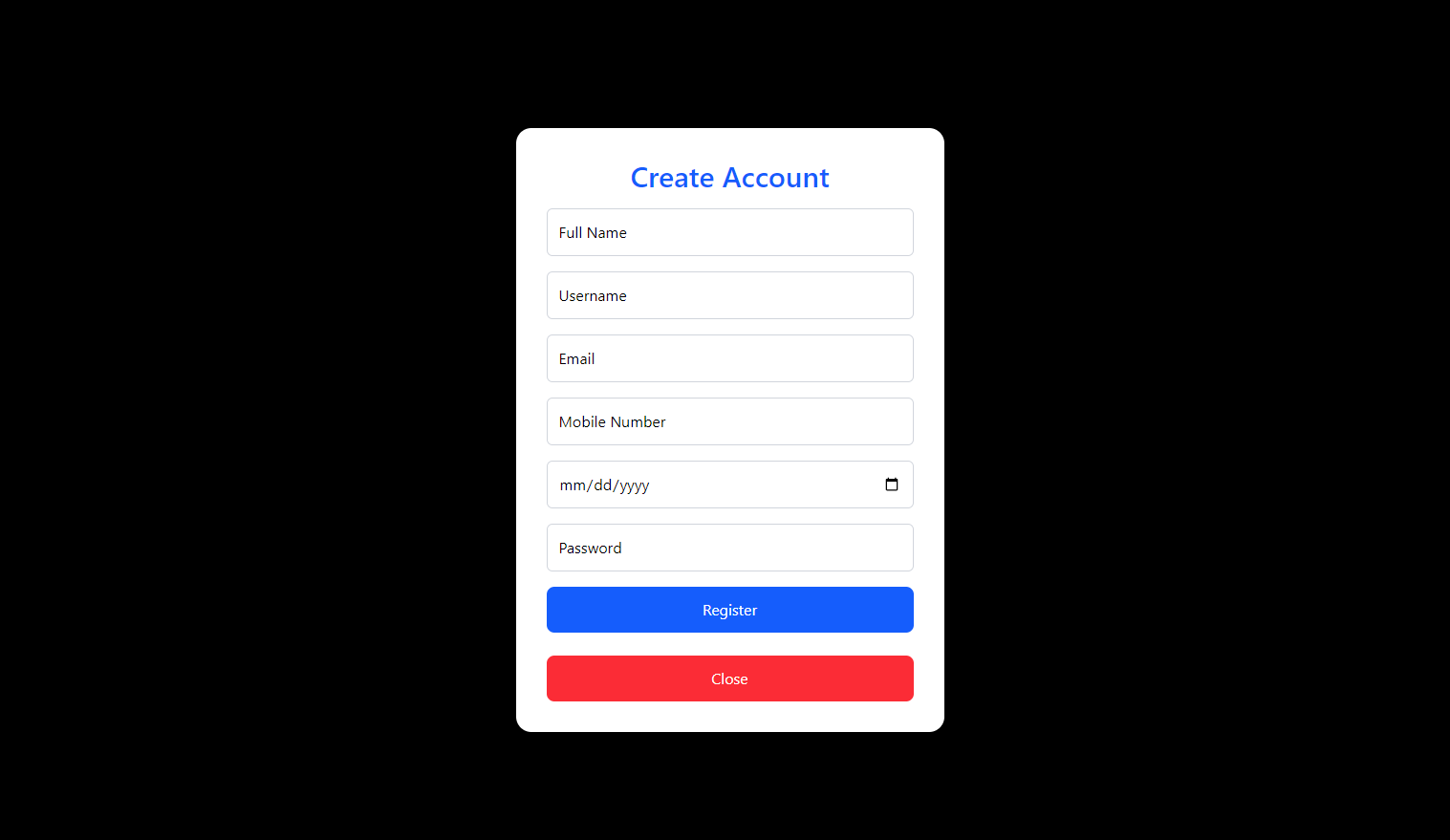
Front-end Login CMD Command



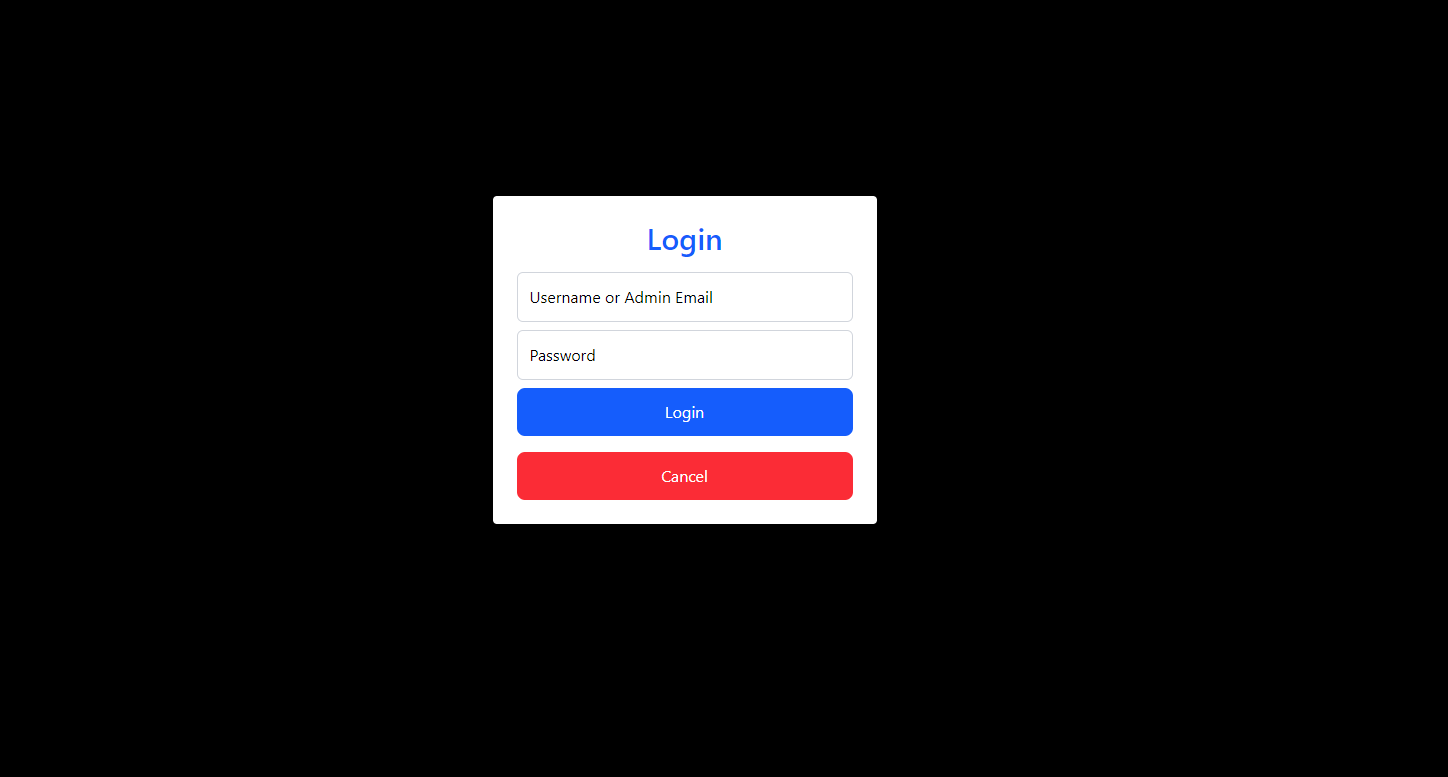
Home Page



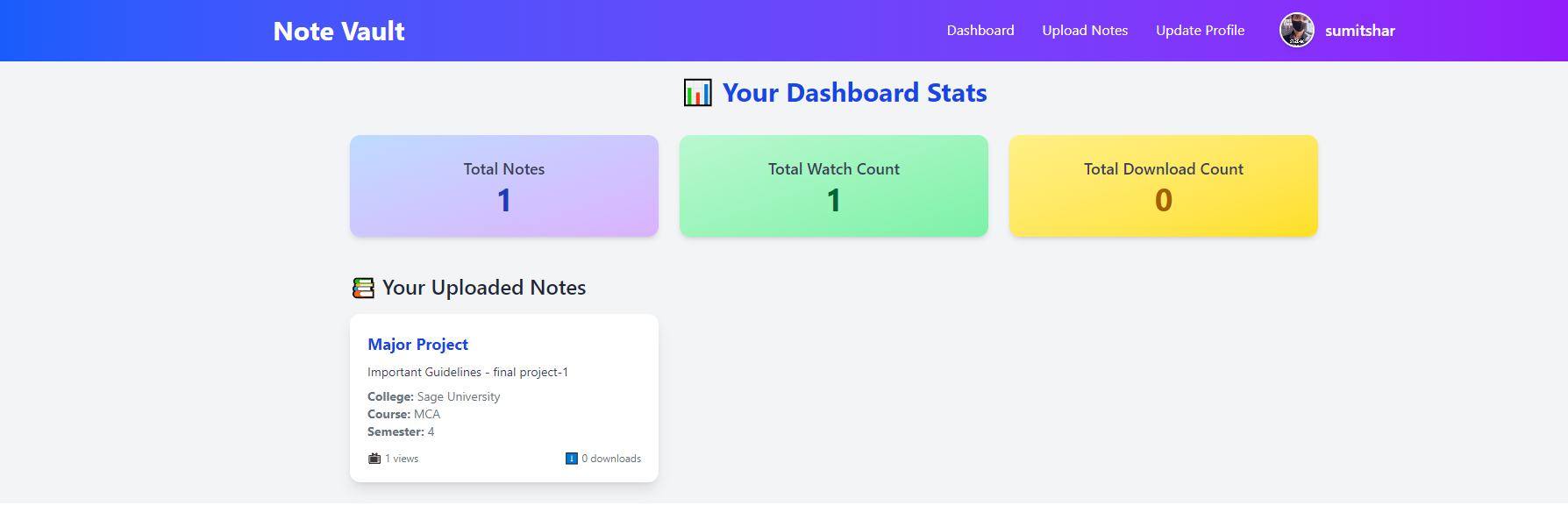
Register Page



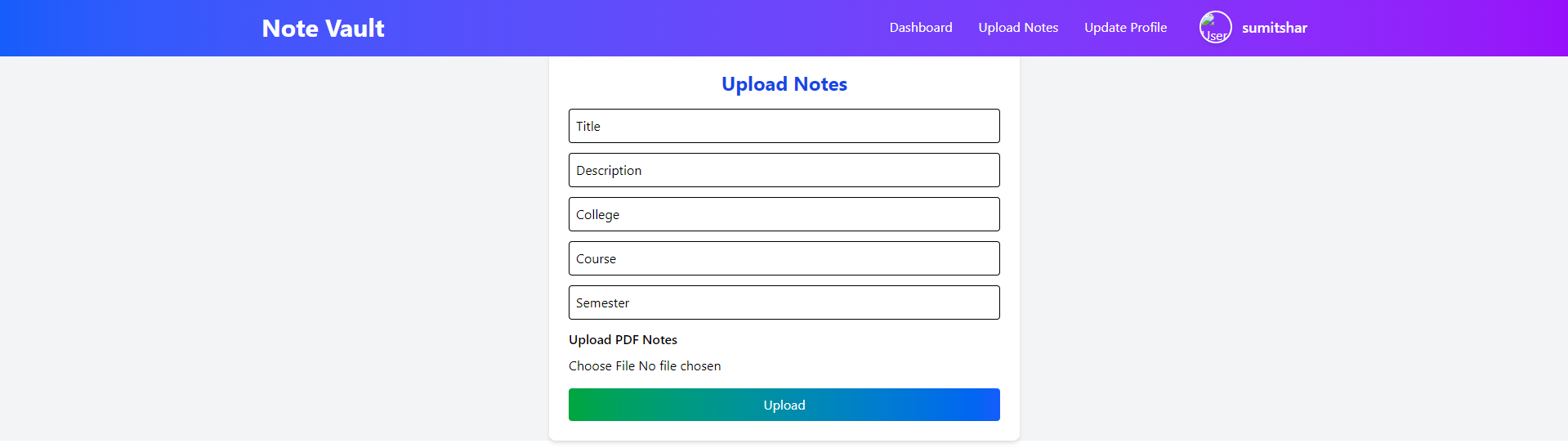
Login Page (User & Admin)



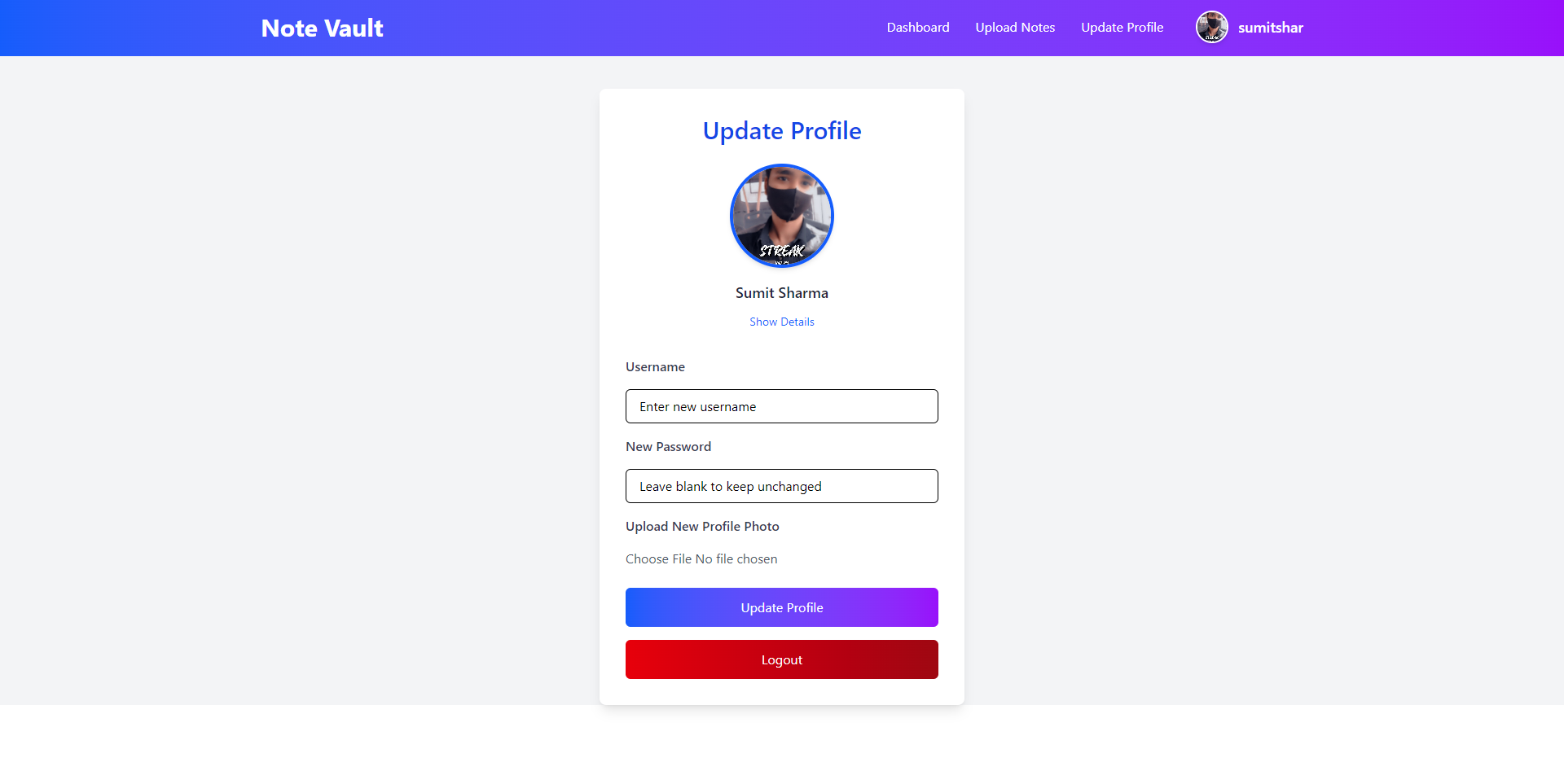
User Dashboard



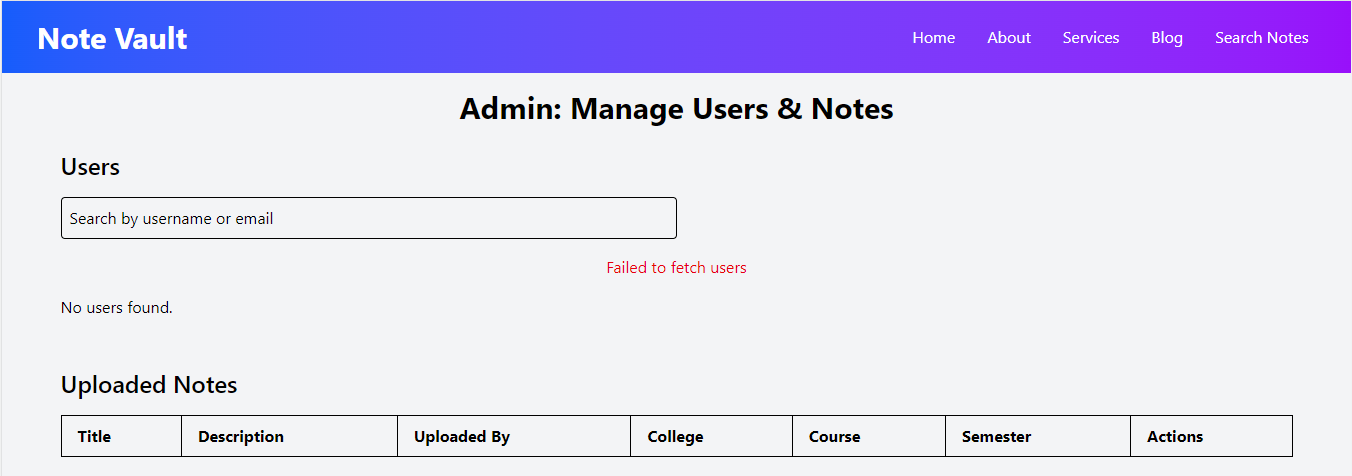
User Upload Notes



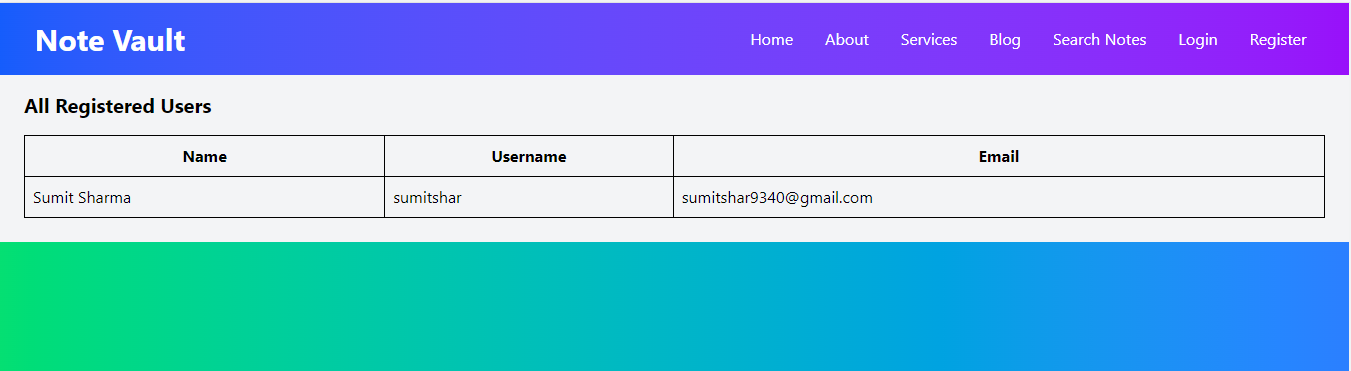
User Update Profile



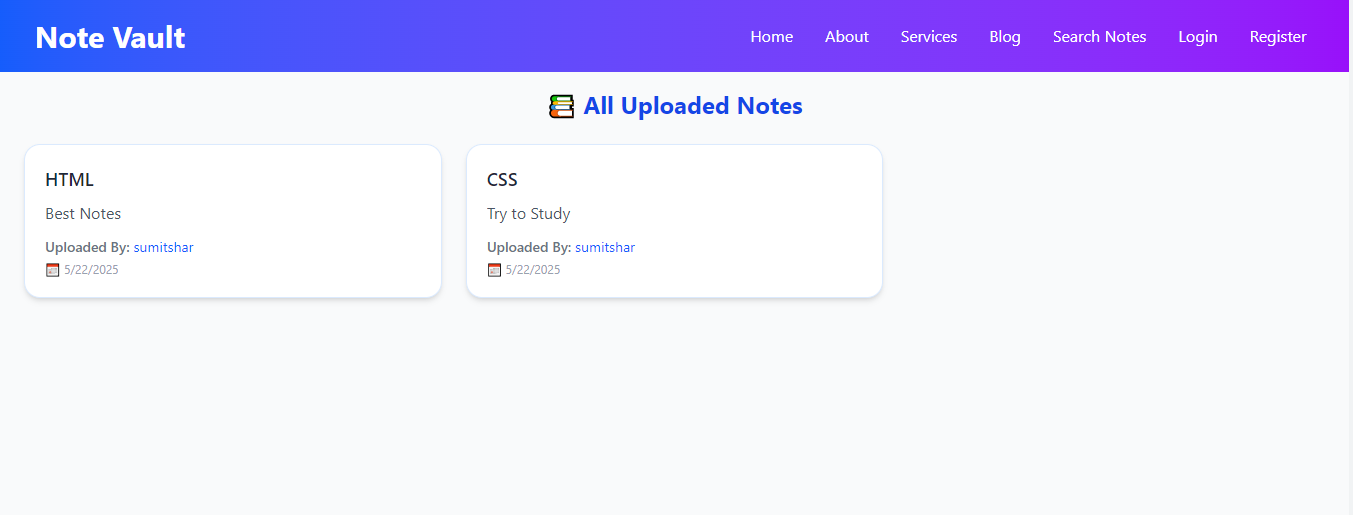
Admin Dashboard



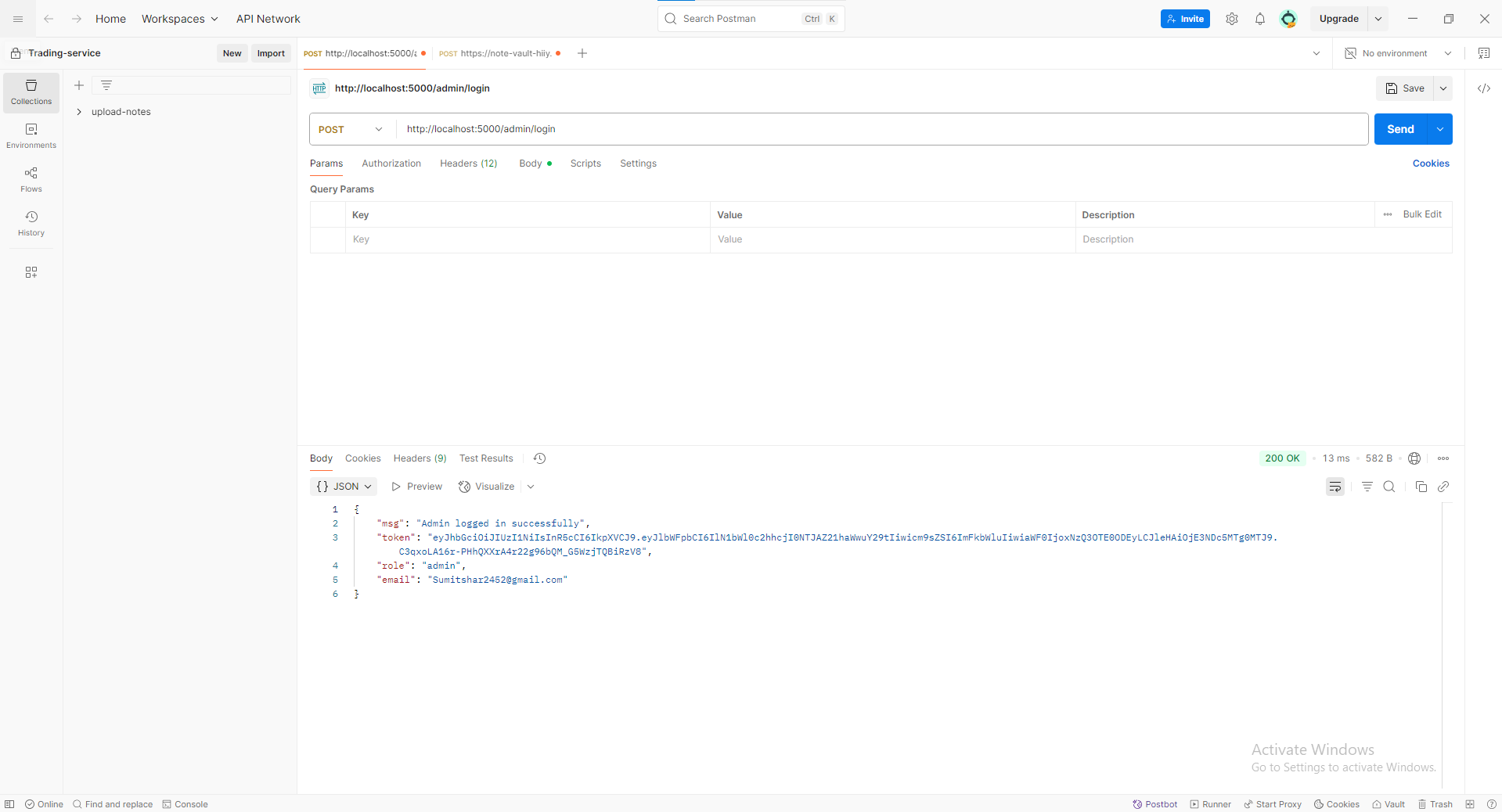
Admin All User List

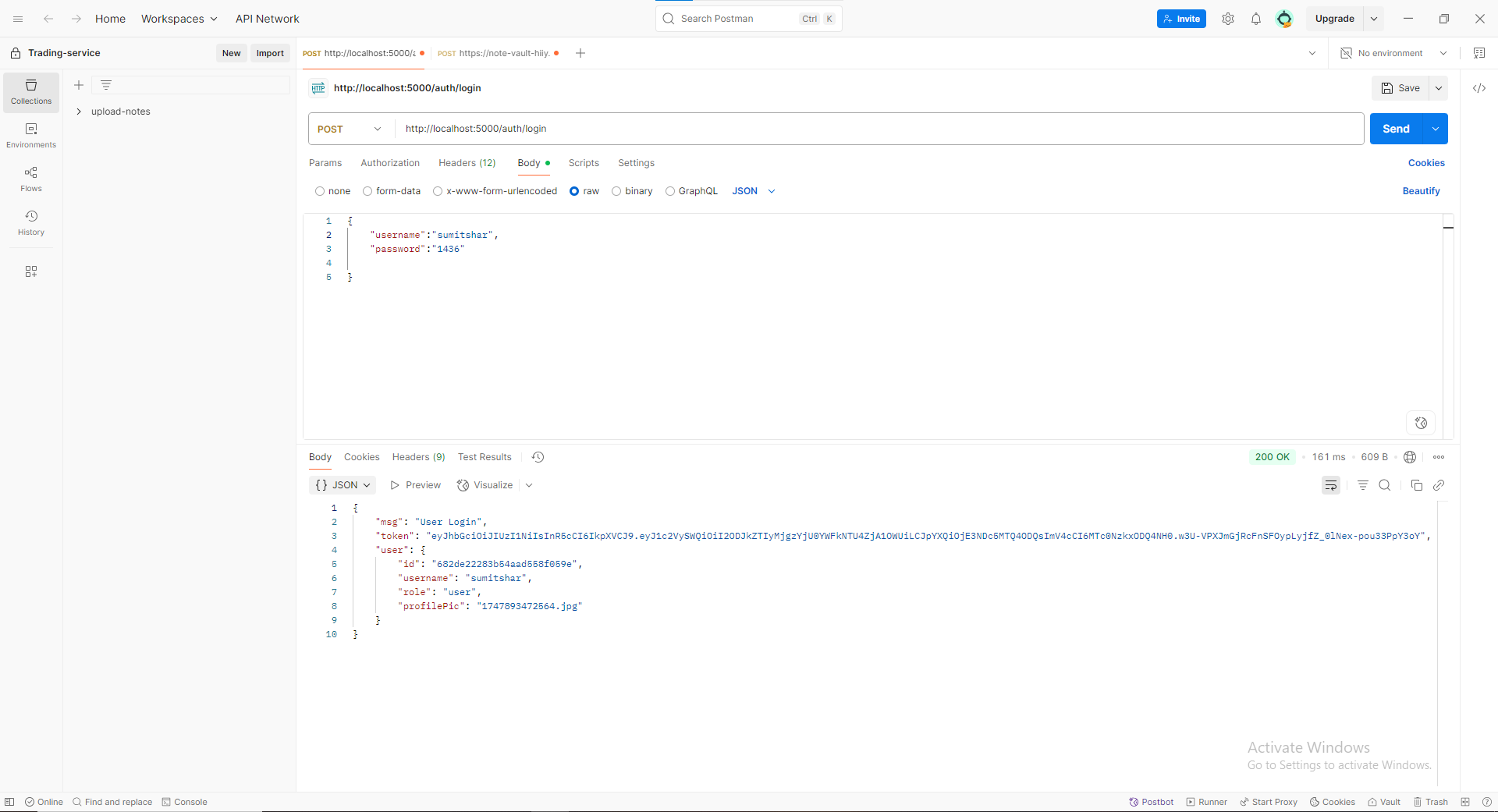


Admin All User upload notes List

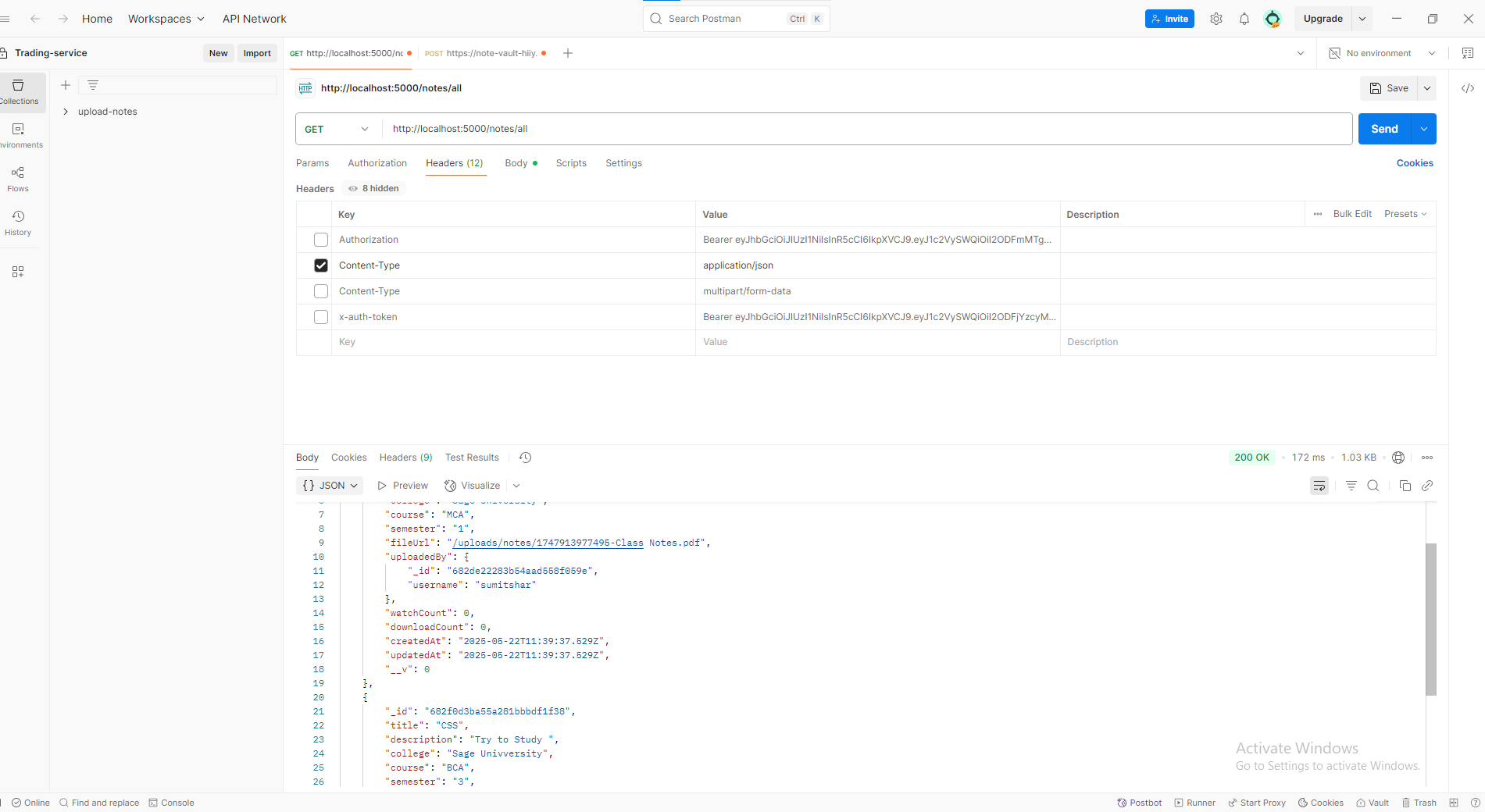


Test Admin Login API in Postman

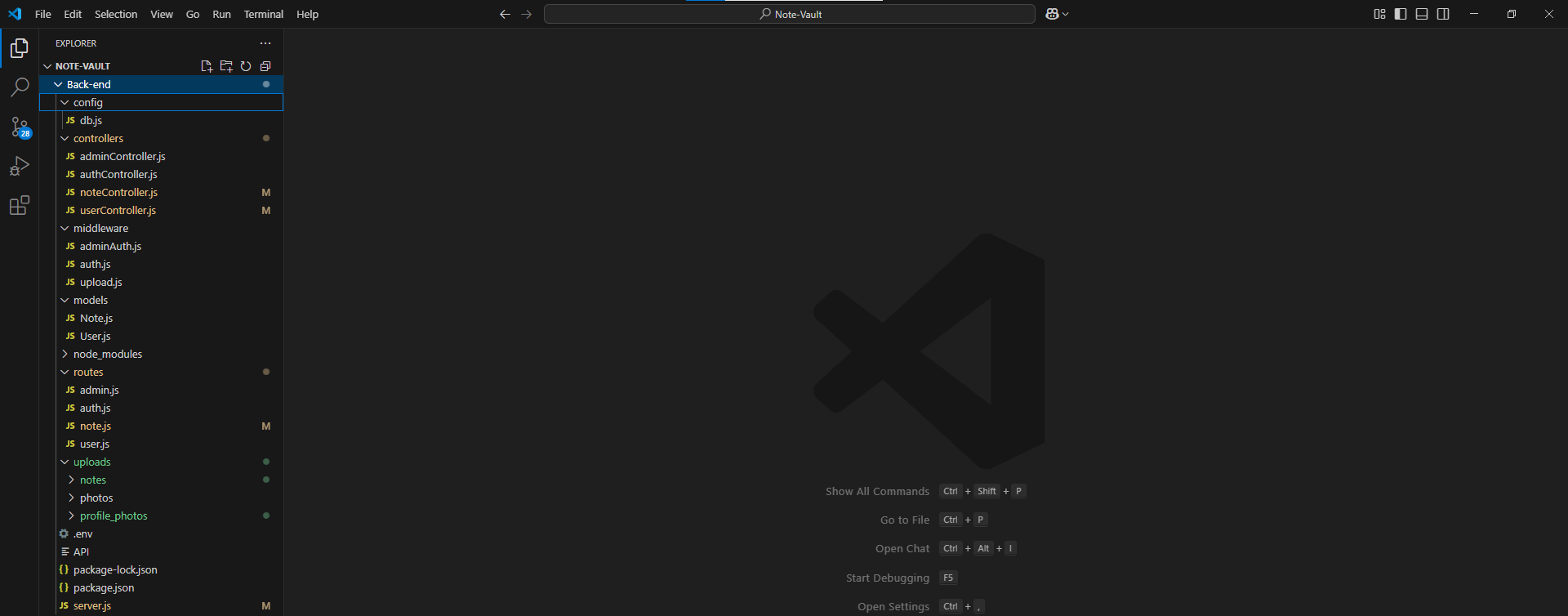


Test User Login API in Postman   


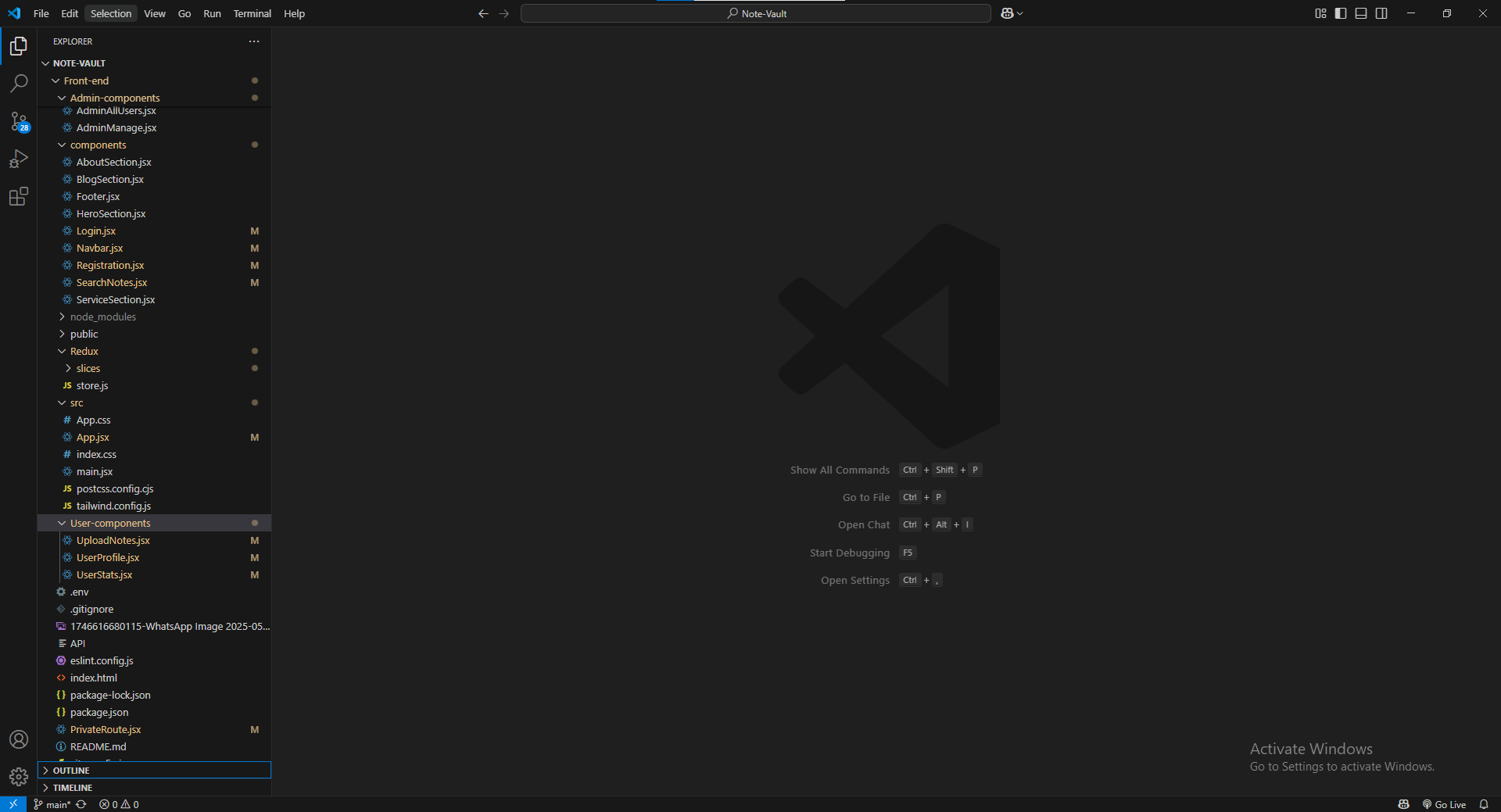
Test All Notes API in Postman

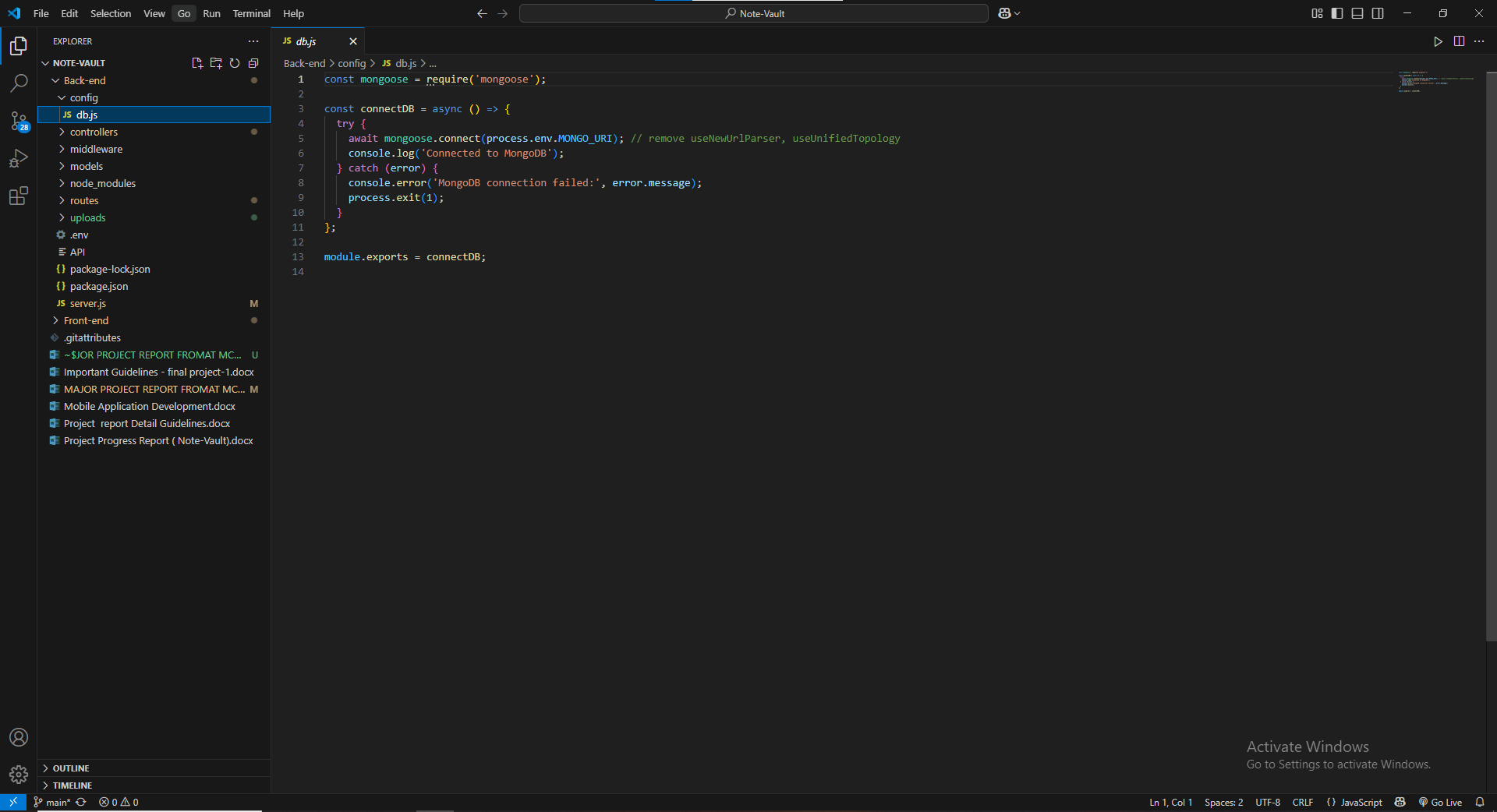


Back-end Folder Structure

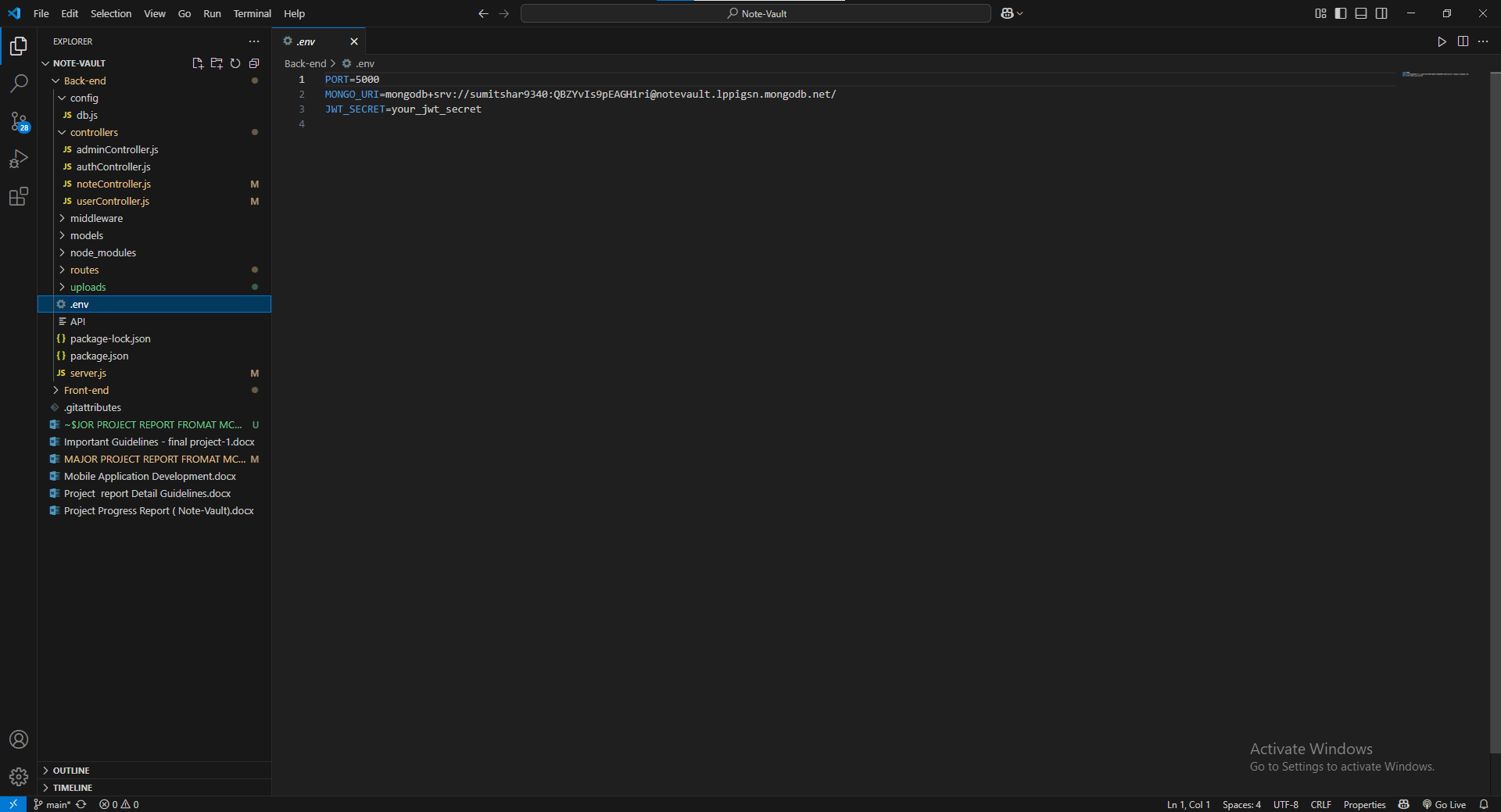


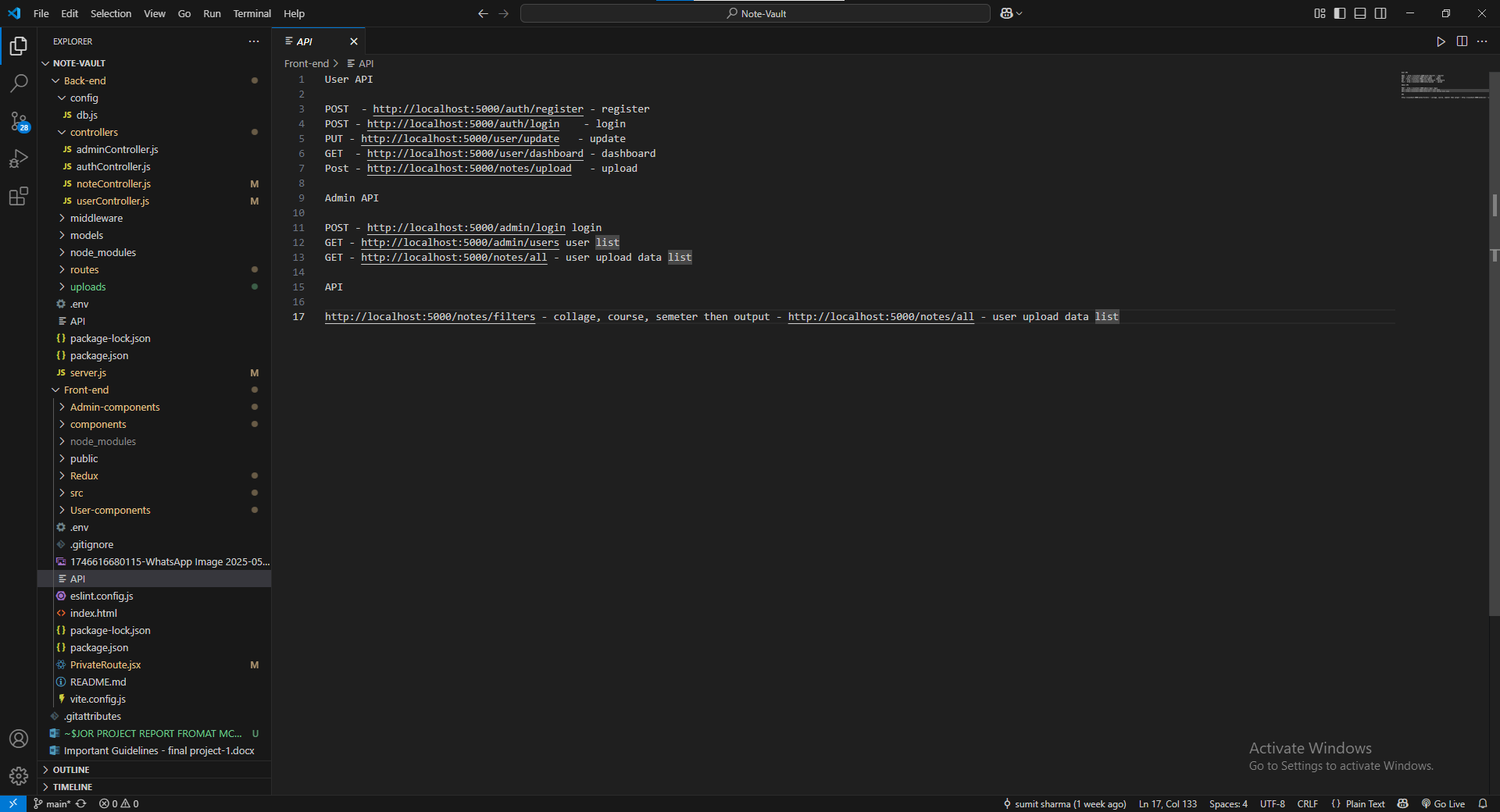
Front-end Folder Structure

MongoDB Connect

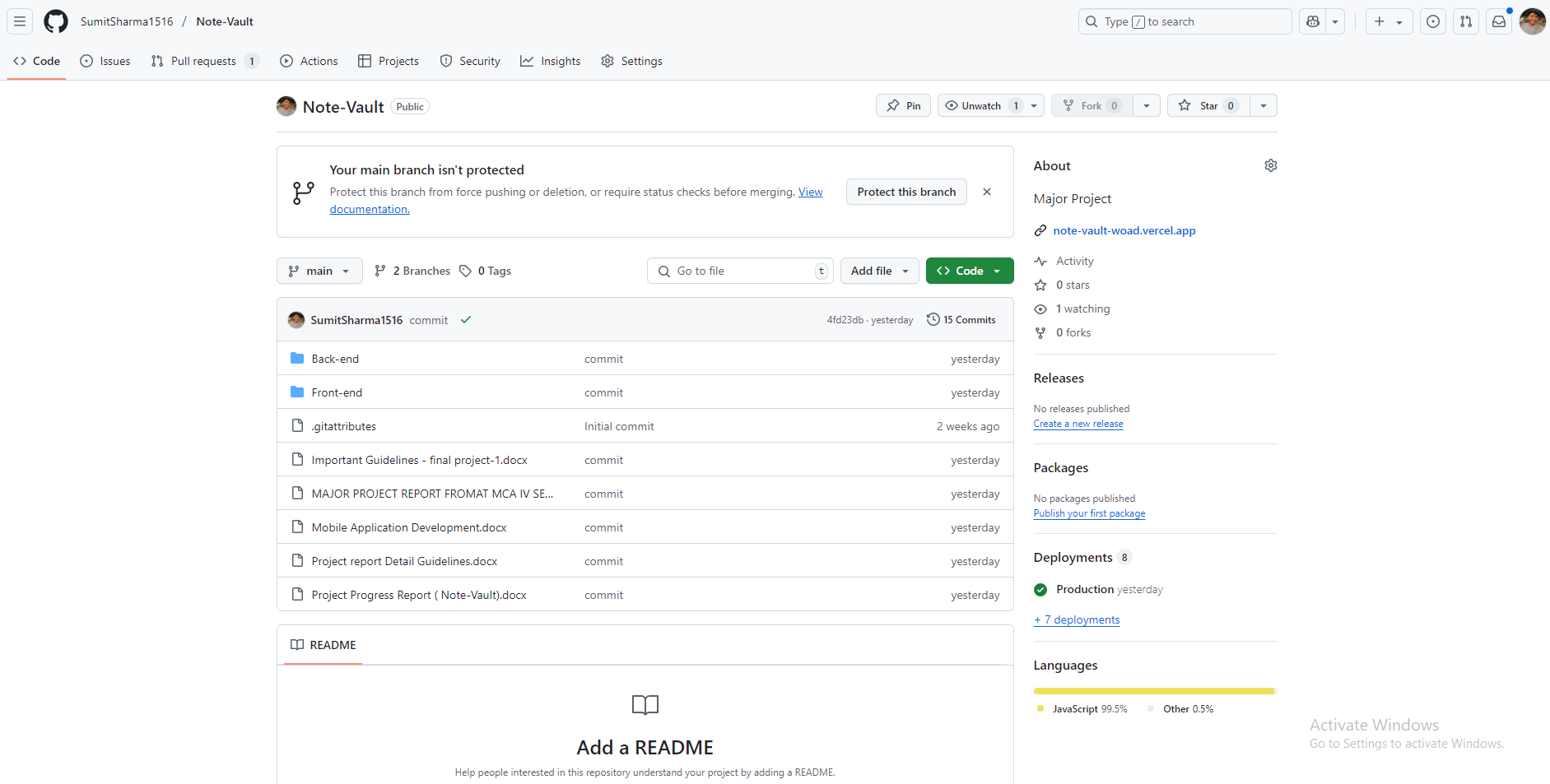


Back-end run Port

Total API Use



Github repositories



Github Commit

