API Documentation

StudyHacks

October 8, 2023

Contents

1 Introduction

1.1 Overview

This documentation provides detailed information about the StudyHacks API, which is designed to facilitate document management, chat processing, and document summarization for educational and research purposes. The API allows users to upload, process, and retrieve documents, create and manage chats related to documents, and generate document summaries.

1.2 Purpose

The StudyHacks API aims to streamline the document management and information retrieval process for students, researchers, and educators. It offers functionalities to upload various types of documents, including PDFs, images, and plain text, and extract textual content from them. Users can also create chat conversations to ask questions related to documents and receive answers generated using AI. Additionally, the API provides document summarization capabilities to condense lengthy texts into concise summaries.

1.3 Authentication

The API utilizes JSON Web Tokens (JWT) for authentication. To access protected endpoints and perform actions such as document uploads, chat creation, and document summarization, users must obtain a valid JWT by logging in with their credentials. The JWT should be included in the request headers for secured access.

1.4 Base URL

All endpoints in this API are relative to the base URL:

```
https://api-docs-studyhacks.onrender.com
```

1.5 Response Format

The API primarily returns responses in JSON format, making it easy for clients to parse and process the data. Successful responses include a status code and the relevant data, while error responses provide details about the encountered issues.

1.6 API Versioning

The API currently operates under version 1 (v1). The version number is included in the base URL as follows:

```
https://api-docs-studyhacks.onrender.com
and verson 1 will be https://api-docs-studyhacks.onrender.com/v1
```

1.7 Rate Limiting

To maintain service quality and prevent abuse, the API enforces rate limiting. Users are limited to a certain number of requests per minute. Exceeding the rate limit may result in temporary access restrictions.

1.8 Getting Started

To begin using the StudyHacks API, users should obtain an API key, which is used for authentication. Detailed instructions on obtaining an API key and making requests to the API can be found in the following sections.

1.9 Contact Information

For inquiries, support, or assistance related to the StudyHacks API, please contact our support team at:

`Samson Mhango  +265 995 22 42 38`
`Patrick  Phandera  +265 885 09 21 48`

Now that you have an overview of the StudyHacks API, let's explore its various endpoints and functionalities in the following sections.

2 User Management Endpoints

2.1 Register User

2.1.1 Endpoint

`POST /register`

2.1.2 Parameters

- `email` (string): User's email address.
- `name` (string): User's name.
- `role` (string): User's role.
- `profile_complete` (boolean): Indicates if the user's profile is complete.
- `profile_picture` (string): URL of the user's profile picture.
- `password` (string): User's password.
- `country` (string): User's country.
- `gender` (string): User's gender.
- `institution` (string): User's institution.
- `mobile` (string): User's mobile number.

2.1.3 Description

Registers a new user in the system.

2.1.4 Example Request

```
POST /register
{
    "email": "user@example.com",
    "name": "John Doe",
    "role": "user",
    "profile_complete": true,
    "profile_picture": "https://example.com/profile.jpg",
    "password": "securepassword",
    "country": "USA",
    "gender": "Male",
    "institution": "XYZ University",
    "mobile": "+1234567890"
}
```

2.1.5 Example Response

```
{
    "msg": "User registered successfully"
}
```

2.2 Login

2.2.1 Endpoint

`POST /login`

2.2.2 Parameters

- `email` (string): User's email address.
- `password` (string): User's password.

2.2.3 Description

Authenticates a user by checking their email and password. Upon successful authentication, a JSON Web Token (JWT) is provided for further access.

2.2.4 Example Request

```
POST /login
{
    "email": "user@example.com",
    "password": "securepassword"
}
```

2.2.5 Example Response

```
{
    "msg": "Login successful",
    "access_token": "your-access-token",
    "user": {
        "_id": "user-id",
        "name": "John Doe",
        "email": "user@example.com",
        "role": "user",
        "profile_complete": true,
        "profile_picture": "https://example.com/profile.jpg",
        "country": "USA",
        "gender": "Male",
        "institution": "XYZ University",
        "mobile": "+1234567890"
    }
}
```

2.3 Get All Users

2.3.1 Endpoint

`GET /users`

2.3.2 Description

Retrieves a list of all users in the system. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

2.3.3 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

2.3.4 Response

A list of user objects, each containing user information.

2.3.5 Example Request

```
GET /users
Authorization: Bearer your-access-token
```

2.3.6 Example Response

```
[
    {
        "_id": "user-id-1",
        "name": "John Doe",
        "email": "john@example.com",
        "role": "user",
        "profile_complete": true,
        "profile_picture": "https://example.com/john.jpg",
        "country": "USA",
        "gender": "Male",
        "institution": "XYZ University",
        "mobile": "+1234567890"
    },
    {
        "_id": "user-id-2",
        "name": "Jane Smith",
        "email": "jane@example.com",
        "role": "admin",
        "profile_complete": false,
        "profile_picture": "",
        "country": "Canada",
        "gender": "Female",
        "institution": "ABC College",
        "mobile": "+9876543210"
    }
]
```

2.4 Get Single User

2.4.1 Endpoint

`GET /users/{user_id}`

2.4.2 Description

Retrieves information about a single user specified by their unique `user_id`. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

2.4.3 Request Parameters

- `user_id` (string): The unique identifier of the user to retrieve.

2.4.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

2.4.5 Response

A JSON object containing user information for the specified user.

2.4.6 Example Request

```
GET /users/user-id-1
Authorization: Bearer your-access-token
```

2.4.7 Example Response

```
{
    "_id": "user-id-1",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user",
    "profile_complete": true,
    "profile_picture": "https://example.com/john.jpg",
    "country": "USA",
    "gender": "Male",
    "institution": "XYZ University",
    "mobile": "+1234567890"
}
```

3 File Processing Endpoints

This section provides information about the endpoints related to processing various types of files, including PDFs, images, and text.

3.1 Upload PDF

3.1.1 Endpoint

`POST /pdfs`

3.1.2 Description

Allows users to upload a PDF file to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.1.3 Request Parameters

- `file` (file): The PDF file to be uploaded.

3.1.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

3.1.5 Response

A JSON object with a success message indicating that the document has been processed successfully.

3.1.6 Example Request

```
POST /pdfs
Authorization: Bearer your-access-token
Content-Type: multipart/form-data

File: Your PDF file
```

3.1.7 Example Response

```
{
    "msg": "Document processed successfully"
}
```

3.2 Upload Image

3.2.1 Endpoint

`POST /images`

3.2.2 Description

Allows users to upload an image file to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.2.3 Request Parameters

- `file` (file): The image file to be uploaded.

3.2.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

3.2.5 Response

A JSON object with a success message indicating that the image has been processed successfully.

3.2.6 Example Request

```
POST /images
Authorization: Bearer your-access-token
Content-Type: multipart/form-data

File: Your image file
```

3.2.7 Example Response

```
{
    "msg": "Image processed successfully"
}
```

3.3 Upload Text

3.3.1 Endpoint

`POST /text`

3.3.2 Description

Allows users to upload text data to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.3.3 Request Parameters

- `text` (string): The text data to be uploaded.

3.3.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

3.3.5 Response

A JSON object with a success message indicating that the text has been processed successfully.

3.3.6 Example Request

```
POST /text
Authorization: Bearer your-access-token
Content-Type: application/json

{
    "text": "This is a sample text."
}
```

3.3.7 Example Response

```
{
    "msg": "Text processed successfully"
}
```

3.4 Get Text Files

3.4.1 Endpoint

`GET /text`

3.4.2 Description

Retrieves a list of all text files that have been uploaded to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.4.3 Response

A JSON array containing information about the text files, including their IDs, content, and timestamps.

3.4.4 Example Request

```
GET /text
Authorization: Bearer your-access-token
```

3.4.5 Example Response

```
[
    {
        "_id": "text-file-id-1",
        "type_": "text",
        "extracted_text": "This is the extracted text from a text
   file.",
        "name": "",
        "chat_ids": [],
        "path": "",
        "timestamp": "2023-10-08 12:00:00"
    },
    {
        "_id": "text-file-id-2",
        "type_": "text",
        "extracted_text": "Another text file content.",
        "name": "",
        "chat_ids": [],
        "path": "",
        "timestamp": "2023-10-08 13:00:00"
    }
]
```

3.5 Get PDF Files

3.5.1 Endpoint

`GET /pdfs`

3.5.2 Description

Retrieves a list of all PDF files that have been uploaded to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.5.3 Response

A JSON array containing information about the PDF files, including their IDs, content, names, chat IDs, paths, and timestamps.

3.5.4 Example Request

```
GET /pdfs
Authorization: Bearer your-access-token
```

3.5.5 Example Response

```
[
    {
        "_id": "pdf-file-id-1",
        "type_": "pdf",
        "extracted_text": "This is the extracted text from a PDF
   file.",
        "name": "sample.pdf",
        "chat_ids": ["chat-id-1"],
        "path": "https://example.com/files/pdf-file-id-1",
        "timestamp": "2023-10-08 14:00:00"
    },
    {
        "_id": "pdf-file-id-2",
        "type_": "pdf",
        "extracted_text": "Another PDF file content.",
        "name": "document.pdf",
        "chat_ids": ["chat-id-2"],
        "path": "https://example.com/files/pdf-file-id-2",
        "timestamp": "2023-10-08 15:00:00"
    }
]
```

3.6 Get Image Files

3.6.1 Endpoint

`GET /images`

3.6.2 Description

Retrieves a list of all image files that have been uploaded to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.6.3 Response

A JSON array containing information about the image files, including their IDs, content, names, chat IDs, paths, and timestamps.

3.6.4 Example Request

```
GET /images
Authorization: Bearer your-access-token
```

3.6.5 Example Response

```
[
    {
        "_id": "image-file-id-1",
        "type_": "image",
        "extracted_text": "This is the extracted text from an image.
    ",
        "name": "sample.jpg",
        "chat_ids": ["chat-id-3"],
        "path": "https://example.com/files/image-file-id-1",
        "timestamp": "2023-10-08 16:00:00"
    },
    {
        "_id": "image-file-id-2",
        "type_": "image",
        "extracted_text": "Another image content.",
        "name": "photo.png",
        "chat_ids": ["chat-id-4"],
        "path": "https://example.com/files/image-file-id-2",
        "timestamp": "2023-10-08 17:00:00"
    }
]
```

3.7 Get All Contents - PDFs, images, and text

3.7.1 Endpoint

`GET /pdf_images_text`

3.7.2 Description

Retrieves a list of all documents, including PDFs, images, and text files, that have been uploaded to the server. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

3.7.3 Response

A JSON array containing information about all documents, including their IDs, types, extracted text, names, chat IDs, paths, and timestamps.

3.7.4 Example Request

```
GET /pdf_images_text
Authorization: Bearer your-access-token
```

3.7.5 Example Response

```
[
    {
        "_id": "document-id-1",
        "type_": "pdf",
        "extracted_text": "This is the extracted text from a PDF file.",
        "name": "sample.pdf",
        "chat_ids": ["chat-id-1"],
        "path": "https://example.com/files/document-id-1",
        "timestamp": "2023-10-08 14:00:00"
    },
    {
        "_id": "document-id-2",
        "type_": "image",
        "extracted_text": "This is the extracted text from an image.",
        "name": "sample.jpg",
        "chat_ids": ["chat-id-3"],
        "path": "https://example.com/files/document-id-2",
        "timestamp": "2023-10-08 16:00:00"
    },
    {
        "_id": "document-id-3",
        "type_": "text",
        "extracted_text": "This is the extracted text from a text file.",
        "name": "",
        "chat_ids": [],
        "path": "",
        "timestamp": "2023-10-08 12:00:00"
    }
]
```

3.8 Download File

3.8.1 Endpoint

`GET /files/download/{document_id}`

3.8.2 Description

Allows users to download a specific file by specifying its unique `document_id`. No authentication is required for this endpoint.

3.8.3 Request Parameters

- `document_id` (string): The unique identifier of the document to download.

3.8.4 Response

The file to be downloaded (PDF, image, or text).

3.8.5 Example Request

```
GET /files/download/document-id-1
```

3.8.6 Example Response

The specified file is downloaded.

4 Chat Processing Endpoints

This section provides information about the endpoints related to chat processing and document summarization.

4.1 Create Chat

4.1.1 Endpoint

`POST /chats/{pdf_id}`

4.1.2 Description

Allows users to create a chat associated with a specific PDF document by providing a question. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.1.3 Request Parameters

- `pdf_id` (string): The unique identifier of the PDF document to associate the chat with.

4.1.4 Request Body

- `question` (string): The question to be asked in the chat.

4.1.5 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

4.1.6 Response

A JSON object containing a success message and details of the created chat.

4.1.7 Example Request

```
POST /chats/pdf-file-id-1
Authorization: Bearer your-access-token
Content-Type: application/json

{
    "question": "What is the main idea of this PDF document?"
}
```

4.1.8 Example Response

```
{
    "msg": "Chat created successfully",
    "chat": {
        "_id": "chat-id-1",
        "user_id": "user-id-1",
        "pdf_id": "pdf-file-id-1",
        "question": "What is the main idea of this PDF document?",
        "answer": "The main idea is..."
    }
}
```

4.2 Create Summary from PDF

4.2.1 Endpoint

`GET /summary/{pdf_id}`

4.2.2 Description

Generates a summary for a specific PDF document by providing its unique `pdf_id`. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.2.3 Request Parameters

- `pdf_id` (string): The unique identifier of the PDF document for which to generate a summary.

4.2.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

4.2.5 Response

A JSON object containing a success message and the generated document summary.

4.2.6 Example Request

```
GET /summary/pdf-file-id-1
Authorization: Bearer your-access-token
```

4.2.7 Example Response

```
{
    "msg": "Summary created successfully",
    "summary": "This is the summary of the PDF document..."
}
```

4.3 Create Summary from Text

4.3.1 Endpoint

`POST /summary`

4.3.2 Description

Generates a summary from provided text data. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.3.3 Request Body

- `text` (string): The text data for which to generate a summary.

4.3.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

4.3.5 Response

A JSON object containing a success message and the generated document summary.

4.3.6 Example Request

```
POST /summary
Authorization: Bearer your-access-token
Content-Type: application/json

{
    "text": "This is the text for which to generate a summary..."
}
```

4.3.7 Example Response

```
{
    "msg": "Summary created successfully",
    "summary": "This is the summary of the provided text..."
}
```

4.4 Get Chats

4.4.1 Endpoint

`GET /chats`

4.4.2 Description

Retrieves a list of all chats created by the authenticated user. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.4.3 Response

A JSON array containing information about the user's chats, including chat IDs, associated PDF IDs, questions, answers, and timestamps.

4.4.4 Example Request

```
GET /chats
Authorization: Bearer your-access-token
```

4.4.5 Example Response

```
[
    {
        "_id": "chat-id-1",
        "user_id": "user-id-1",
        "pdf_id": "pdf-file-id-1",
        "question": "What is the main idea of this PDF document?",
        "answer": "The main idea is...",
        "timestamp": "2023-10-08 18:00:00"
    },
    {
        "_id": "chat-id-2",
        "user_id": "user-id-1",
        "pdf_id": "pdf-file-id-2",
        "question": "Can you explain this PDF's concept?",
        "answer": "Sure, it's about...",
        "timestamp": "2023-10-08 19:00:00"
    }
]
```

4.5 Get Chat by ID

4.5.1 Endpoint

`GET /chats/{chat_id}`

4.5.2 Description

Retrieves a specific chat by providing its unique `chat_id`. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.5.3 Request Parameters

- `chat_id` (string): The unique identifier of the chat to retrieve.

4.5.4 Response

A JSON object containing details of the specified chat, including the chat ID, associated PDF ID, question, answer, and timestamp.

4.5.5 Example Request

```
GET /chats/chat-id-1
Authorization: Bearer your-access-token
```

4.5.6 Example Response

```
{
    "_id": "chat-id-1",
    "user_id": "user-id-1",
    "pdf_id": "pdf-file-id-1",
    "question": "What is the main idea of this PDF document?",
    "answer": "The main idea is...",
    "timestamp": "2023-10-08 18:00:00"
}
```

4.6 Update Chat by ID

4.6.1 Endpoint

`PUT /chats/{chat_id}`

4.6.2 Description

Updates a specific chat by providing its unique `chat_id`. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.6.3 Request Parameters

- `chat_id` (string): The unique identifier of the chat to update.

4.6.4 Request Body

A JSON object containing the fields to be updated in the chat. Fields that can be updated include `question`, `answer`, and any other relevant fields.

4.6.5 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

4.6.6 Response

A JSON object containing a success message if the chat is updated successfully.

4.6.7 Example Request

```
PUT /chats/chat-id-1
Authorization: Bearer your-access-token
Content-Type: application/json

{
    "answer": "The updated answer to the question..."
}
```

4.6.8 Example Response

```
{
    "msg": "Chat updated successfully"
}
```

4.7 Delete Chat by ID

4.7.1 Endpoint

`DELETE /chats/{chat_id}`

4.7.2 Description

Deletes a specific chat by providing its unique `chat_id`. Requires authentication with a valid JSON Web Token (JWT) provided in the request header.

4.7.3 Request Parameters

- `chat_id` (string): The unique identifier of the chat to delete.

4.7.4 Request Header

- `Authorization` (string): Bearer token with the JWT obtained during login.

4.7.5 Response

A JSON object containing a success message if the chat is deleted successfully.

4.7.6 Example Request

```
DELETE /chats/chat-id-1
Authorization: Bearer your-access-token
```

4.7.7 Example Response

```
{
    "msg": "Chat deleted successfully"
}
```

5 Conclusion

5.1 Summary

In this documentation, we have covered the StudyHacks API, a powerful tool designed to enhance document management, chat processing, and document summarization for educational and research purposes. Users can leverage this API to seamlessly upload documents, extract text from various formats, create chat conversations for document-related inquiries, and generate concise document summaries. With robust authentication, rate limiting, and versioning features, the StudyHacks API offers a secure and efficient solution for handling educational and research materials.

5.2 Future Developments

As we continue to improve and expand the StudyHacks platform, the API will also evolve to offer new features and capabilities. Future developments may include enhanced AI-driven document analysis, support for additional document types, and improved chat interactions. We welcome user feedback and suggestions for further enhancements.

5.3 Get Started Today

We invite you to explore the StudyHacks API and integrate its functionalities into your educational or research applications. Whether you are a student, educator, or researcher, this API can help streamline your document-related tasks and improve information retrieval.

5.4 Contact Us

If you have any questions, encounter issues, or need assistance while using the StudyHacks API, please do not hesitate to reach out to our dedicated support team. We are here to assist you in making the most of this powerful tool.

Thank you for choosing StudyHacks to enhance your document management and research endeavors. We look forward to seeing the innovative ways you leverage our API to achieve your goals.

5.5 Stay Informed

Stay informed about the latest updates, announcements, and news related to the StudyHacks API by visiting our website and subscribing to our newsletter. We are committed to providing you with cutting-edge solutions for your educational and research needs.