NodejsSDK

Rivalz Nodejs SDK for developers

https://www.npmjs.com/package/rivalz-client

Rivalz client Node.js SDK

This is a TypeScript library that provides functionalities for Rivazl AI

Features

- **Upload Files**: Upload any file to the Rivalz platform and get an IPFS hash.
- **Upload Passport Images**: Upload passport images to the Rivalz platform.
- **Download Files**: Download files from the Rivalz platform using an IPFS hash.v
- **Delete Files**: Delete files from the Rivalz platform using an IPFS hash.
- **Vectorize Documents**: Vectorize documents to create a RAG (Retrieval-Augmented Generation) based on the document uploaded.
- **Create conversations**: Create conversations based on the document uploaded.

Installation

Before getting started, ensure that you have both **Node.js** and either **npm** or **yarn** installed. These are essential for managing the Rivalz client dependencies.

To install the Rivalz client, run one of the following commands:

```
# Using npm
npm install rivalz-client

# Or, using yarn
yarn add rivalz-client
```

Usage

1. After installing the package, proceed to the Rivalz Dashboard to generate your **encryption key** and **secret key**:

- **Encryption Key**: Used for encrypting files to ensure data security.
- **Secret Key**: Required for authenticating API requests to access Rivalz services.

2. Import and use the RivalzClient class in your TypeScript/JavaScript code:

```
import RivalzClient from 'rivalz-client';
const rivalzClient = new RivalzClient('your-secret-key');
```

API

---

1. Upload File

```
rivalzClient.uploadFile(file, fileName)
```

- file: A readable stream of the file to be uploaded.
- Returns a promise that resolves to the IPFS hash of the uploaded file.

2. Upload passport file

```
rivalzClient.uploadPassport(file)
```

- file: A readable stream of the file to be uploaded.
- Returns a promise that resolves to the IPFS hash of the uploaded file.

3. Download File and save it to the local file system (Node.js only)

```
rivalzClient.downloadFile(ipfsHash, savePath)
```

- ipfsHash: The IPFS hash of the file to be downloaded.
- savePath: The path where the downloaded file will be saved.
- Returns a promise that resolves to the path of the saved file.

4. Download the File and return it as a buffer

```
rivalzClient.download(ipfsHash)
```

- ipfsHash: The IPFS hash of the file to be downloaded.
- Returns a promise that resolves to a buffer containing the downloaded file.

5. Delete File

```
rivalzClient.deleteFile(ipfsHash)
```

- ipfsHash: The IPFS hash of the file to be deleted.
- Returns a promise that resolves to the IPFS hash of the deleted file.

Please replace `your-secret`, `file`, `passport`, `ipfsHash`, and `savePath` with actual values when using the `RivalzClient` class.

~~~
```javascript
const RivalzClient = require('rivalz-client');

const fs = require('node:fs');

const rivalzClient = new RivalzClient('your-secret-key');

//Upload File
async function uploadFile() {
    const filePath = 'file_path';
    const buffer = fs.readFileSync(filePath)
    const fileName = filePath.split('/').pop();
    try {
        const filelog = await rivalzClient.uploadFile(buffer,fileName);
        console.log(filelog);
    } catch (error) {
        console.error('Error uploading file:', error);
    }
}
```
~~~

6. RAG (Retrieval-Augmented Generation) API

Prerequisites

Before using the RAG API, you need api key and some rivalz credits. Claim for free now here

Creating a knowledge base from a document

To vectorize a document and create a knowledge base for Retrieval-Augmented Generation (RAG), use the `createRagKnowledgeBase` method, which takes the document's file path as input. This method generates a vectorized embedding of the document, assigns it a **knowledge base ID**, and stores it for future use in RAG-based conversations. Currently, this process supports only PDF files.

Click here to learn How to create a knowledge base

```
const response = await client.createRagKnowledgeBase('path/to/your/docume
console.log(response)
// {'id': '66fa5bf022e73c17073768f0', 'name': 'test', 'files': '172768356
```

Adding documents to an existing knowledge base

To add a document to an existing knowledge base, use the `addDocumentToKnowledgeBase` method with the knowledge base id and the path to the document.

```
const response = await client.addDocumentToKnowledgeBase('path/to/your/do
console.log(response)
```

Deleting documents from an existing knowledge base

To delete a document from an existing knowledge base, use the `deleteDocumentFromKnowledgeBase` method with the knowledge base id and the document name.

```
const response = await client.deleteDocumentFromKnowledgeBase('document_i
console.log(response)
```

Getting all knowledge bases

To get all knowledge bases, use the `getKnowledgeBases` method.

```
const response = await client.getKnowledgeBases()
console.log(response)
```

Getting details of a knowledge base

To get details of a knowledge base, use the `getKnowledgeBase` method with the knowledge base id.

```
const response = await client.getKnowledgeBase('knowledge_base_id')
console.log(response)
```

7. Conversations

To initiate a conversation in the RAG (Retrieval Augmented Generation) system, use the `createChatSession` method. This method requires the **knowledge base ID** (from your existing knowledge base) and the **question** you want to ask. The AI will return a response based on the context provided by the knowledge base, along with a **chat session ID** to continue the conversation if needed.

```
const response = await client.createChatSession('knowledge_base_id', 'que
console.log(response)
// {'answer': 'Hello! How can I help you today? \n', 'session_id': '66fa6
```

Adding a message to a conversation

To add a message to a conversation, use the same method `createChatSession` with the chat session id and the message.

```
const response = await client.createChatSession('knowledge_base_id', 'mes
console.log(response)
```

Getting all conversations

To get all conversations, use the `getChatSessions` method.

```
const response = await client.getChatSessions()
console.log(response)
```

Getting details of a conversation

To get details of a conversation (which contains chat history for this conversation), use the `getChatSession` method with the chat session id.

```
const response = client.getChatSession('chat_session_id')
console.log(response)
```

Get uploaded documents

To get all uploaded documents, use the `getUploadedDocuments` method.

```
const response = await client.getUploadedDocuments()
console.log(response)
```

Examples

Here is a complete example demonstrating how to use the `rivalz-client` to create a simple RAG conversation based on a PDF document:

```
/*
main.ts
 */

import RivalzClient from 'rivalz-client';

const main = async () => {
  // Initialize the RivalzClient with the secret token
  const client = new RivalzClient('your-secret-key');
  // create knowledge base
  const knowledgeBase = await client.createRagKnowledgeBase('sample.pdf',
  const knowledgeBaseId = knowledgeBase.id;
  if(knowledgeBase.status !== 'ready') {
    console.log('Knowledge base is still processing');
    // wait for 5 seconds to allow the process to finish
    await new Promise(resolve => setTimeout(resolve, 5000));
  }
  // create conversation
  let conversation = await client.createChatSession(knowledgeBaseId, 'wha
  const conversationId = conversation.session_id;
  // add message to conversation
  conversation = await client.createChatSession(knowledgeBaseId, 'What is
  console.log(conversation.answer);

}
main()
```