Generate Metadata Script

`generateMetadata.js` is a Node.js script designed to automate the creation of metadata JSON files for NFTs. It processes a list of uploaded image links, optionally incorporates custom descriptions, and generates structured metadata compliant with NFT standards.

Table of Contents

- Features
- Prerequisites
- Installation
- Configuration
  - Input Files
  - Directory Structure
- Usage
- Output
- Customization
- Troubleshooting
- Contributing
- License

Features

- **Automated Metadata Generation**: Converts uploaded image links into structured JSON metadata files.
- **Custom Descriptions**: Supports optional custom descriptions via a `descriptions.json` file.
- **IPFS Integration**: References images using IPFS URIs.
- **Scalable**: Easily handles large numbers of NFT entries.
- **Extensible Attributes**: Placeholder for adding specific NFT attributes.

Prerequisites

- **Node.js**: Ensure you have Node.js installed. You can download it from nodejs.org.
- **npm**: Node.js typically comes with npm. Verify installation with:

```
npm -v
```

Installation

1. **Clone the Repository**

```
git clone https://github.com/yourusername/your-repo.git
cd your-repo
```

2. **Install Dependencies**

This script uses Node.js's built-in `fs` and `path` modules, so no additional dependencies are required. However, ensure your Node.js version is up-to-date.

```
npm install
```

Configuration

Input Files

1. **Uploaded Links File (`uploaded_links.txt`)**
   - **Path**: Located in the root directory (`./uploaded_links.txt`).
   - **Format**: Each line should follow the pattern:

     ```
     Uploaded art_XXXX.png: <IPFS_HASH>
     ```

     Example:

     ```
     Uploaded art_0001.png: QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco
     Uploaded art_0002.png: QmYwAPJzv5CZsnAzt8auV2g9d2XQjfz3jB6u1a9UKHk1rL
     ```

2. **Descriptions File (`descriptions.json`)** *(Optional)*
   - **Path**: Located in the root directory (`./descriptions.json`).
   - **Format**: JSON object mapping NFT IDs to descriptions.

     Example:

     ```
     {
       "0001": "This is a unique piece from the TONS Universe collection.",
       "0002": "Exclusive artwork representing the TONS Universe."
     }
     ```

   - **Note**: If `descriptions.json` is not provided or a description for a specific NFT ID is missing, the script will use a default description.

Directory Structure

Ensure your project directory has the following structure:

```
your-repo/
├── generateMetadata.js
├── uploaded_links.txt
├── descriptions.json (optional)
├── metadata/ (will be created by the script)
└── package.json
```

Usage

1. **Prepare Input Files**

- Populate `uploaded_links.txt` with your uploaded image links following the specified format.
- (Optional) Create `descriptions.json` with custom descriptions for your NFTs.

2. **Run the Script**

   Execute the script using Node.js:

   ```
   node generateMetadata.js
   ```

   Output:

   - The script will create a `metadata` directory (if it doesn't already exist).
   - For each valid entry in `uploaded_links.txt`, a corresponding JSON file (e.g., `0001.json`) will be generated in the `metadata` directory.

3. **Sample Output**

   ```
   {
       "name": "TONS Universe NFT #0001",
       "description": "This is a unique piece from the TONS Universe
   collection.",
       "image": "ipfs://QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco",
       "attributes": [
           // Add any specific attributes here
           // Example:
           // { "trait_type": "Background", "value": "Blue" }
       ]
   }
   ```

Output

- **Metadata Files**: Located in the `metadata/` directory, each JSON file corresponds to an NFT and contains the following fields:
  - `name`: The name of the NFT.
  - `description`: A description of the NFT.
  - `image`: The IPFS URI of the NFT image.
  - `attributes`: An array for additional NFT attributes (currently empty; customize as needed).

Customization

Adding Attributes

To include specific attributes for each NFT, modify the `attributes` array in the generated JSON files. You can automate this by updating the script to include attribute data based on your requirements.

Example:

```
"attributes": [
    { "trait_type": "Background", "value": "Blue" },
```

```
    { "trait_type": "Eyes", "value": "Green" },
    { "trait_type": "Accessory", "value": "Hat" }
]
```

Extending the Script

You can enhance the script to include more features, such as:

- **Validation**: Ensure IPFS hashes are valid.
- **Logging**: Implement more robust logging mechanisms.
- **Error Handling**: Improve error messages and recovery options.
- **Integration**: Connect with other services or APIs for dynamic data.

Troubleshooting

- **Metadata Directory Not Created**

  Ensure the script has the necessary permissions to create directories in the project path.

- **Error Parsing `descriptions.json`**

  Verify that `descriptions.json` is valid JSON. Use a JSON validator to check for syntax errors.

- **Incorrect Line Format in `uploaded_links.txt`**

  Ensure each line follows the specified format:

  ```
  Uploaded art_XXXX.png: <IPFS_HASH>
  ```

  Lines not matching this pattern will be skipped with a warning.

- **Missing IPFS Hashes**

  Ensure that all uploaded links in `uploaded_links.txt` include valid IPFS hashes.

Contributing

Contributions are welcome! Please follow these steps:

1. **Fork the Repository**
2. **Create a Feature Branch**

   ```
   git checkout -b feature/YourFeature
   ```

3. **Commit Your Changes**

   ```
   git commit -m "Add Your Feature"
   ```

4. **Push to the Branch**

   ```
   git push origin feature/YourFeature
   ```

5. **Open a Pull Request**

Describe your changes and submit the pull request for review.

License

This project is licensed under the MIT License.

Happy NFT Creating!

Complete `generateMetadata.js` Code

Below is the complete `generateMetadata.js` script. You can copy and paste it into your project.

```
const fs = require('fs');
const path = require('path');

// Path to the uploaded links file
const uploadedLinksPath = path.join(__dirname, 'uploaded_links.txt');

// Directory to save metadata JSON files
const metadataDir = path.join(__dirname, 'metadata');

// Base URL for IPFS
const ipfsBaseURL = 'ipfs://';

// Path to the descriptions file (optional)
const descriptionsPath = path.join(__dirname, 'descriptions.json');

// Ensure the metadata directory exists
if (!fs.existsSync(metadataDir)) {
    fs.mkdirSync(metadataDir);
    console.log(`Created metadata directory at ${metadataDir}`);
}

// Read custom descriptions if available
let descriptions = {};
if (fs.existsSync(descriptionsPath)) {
    const descriptionsData = fs.readFileSync(descriptionsPath, 'utf8');
    try {
        descriptions = JSON.parse(descriptionsData);
        console.log(`Loaded custom descriptions from ${descriptionsPath}`);
    } catch (parseErr) {
        console.error(`Error parsing ${descriptionsPath}:`, parseErr);
        console.warn(`Using default descriptions.`);
        descriptions = {};
    }
} else {
    console.warn(`No descriptions.json found. Using default descriptions.`);
}

// Read the uploaded_links.txt file
```

```
fs.readFile(uploadedLinksPath, 'utf8', (err, data) => {
    if (err) {
        console.error(`Error reading ${uploadedLinksPath}:`, err);
        return;
    }

    // Split the file content into lines
    const lines = data.split('\n').filter(line => line.trim() !== '');

    lines.forEach((line, index) => {
        // Example line: "Uploaded art_0004.png:
QmcawuGxRQteftLuJPRLAYtUaoq5kQd8VSL5mHdCSdhL66"
        const regex = /Uploaded\s+art_(\d+)\.png:\s+([A-Za-z0-9]+)/;
        const match = line.match(regex);

        if (match) {
            const nftId = match[1]; // e.g., "0004"
            const ipfsHash = match[2];  // e.g.,
"QmcawuGxRQteftLuJPRLAYtUaoq5kQd8VSL5mHdCSdhL66"

            // Generate unique description
            const description = descriptions[nftId] || `TONS Universe NFT #${nftId} -
This unique piece is part of the exclusive TONS Universe collection.`;

            // Create metadata object
            const metadata = {
                name: `TONS Universe NFT #${nftId}`,
                description: description,
                image: `${ipfsBaseURL}${ipfsHash}`,
                attributes: [
                    // Add any specific attributes here
                    // Example:
                    // { "trait_type": "Background", "value": "Blue" },
                ]
            };

            // Define the metadata file path
            const metadataFilePath = path.join(metadataDir, `${nftId}.json`);

            // Write metadata to JSON file
            fs.writeFile(metadataFilePath, JSON.stringify(metadata, null, 4), (err) =>
{
                if (err) {
                    console.error(`Error writing metadata for NFT #${nftId}:`, err);
                } else {
                    console.log(`Metadata for NFT #${nftId} created successfully.`);
                }
            });
        } else {
            console.warn(`Line format is incorrect and was skipped: "${line}"`);
        }
```

```
    });
});
```

Example `uploaded_links.txt`

Ensure your `uploaded_links.txt` follows the specified format. Below is an example:

```
Uploaded art_0001.png: QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco
Uploaded art_0002.png: QmYwAPJzv5CZsnAzt8auV2g9d2XQjfz3jB6u1a9UKHk1rL
Uploaded art_0003.png: QmZ1234567890abcdefghij1234567890abcdefghi
```

Example `descriptions.json`

If you choose to use a `descriptions.json` for custom descriptions, ensure it is properly formatted. Below is an example:

```
{
  "0001": "This is a unique piece from the TONS Universe collection.",
  "0002": "Exclusive artwork representing the TONS Universe.",
  "0003": "A rare NFT showcasing the beauty of the TONS Universe."
}
```

Converting Markdown to PDF with Dark Theme

To convert this Markdown document to a PDF with a dark theme, follow these steps:

1. **Install a Markdown to PDF Converter**

   You can use Pandoc or the Markdown PDF extension for VS Code.

2. **Using Pandoc**

   ```
   pandoc -s your-document.md -o your-document.pdf
   ```

   To apply a dark theme, you may need to customize the CSS or use a pre-defined dark template.

3. **Using VS Code Extension**

   - Install the **Markdown PDF** extension.
   - Open your Markdown file in VS Code.
   - Press `Ctrl+Shift+P` and select `Markdown PDF: Export (pdf)`.
   - *Note:* The default export may not have a dark theme. To apply a dark theme, customize the CSS as per the extension's documentation.

Additional Resources

- Node.js Documentation
- IPFS Documentation

- GitHub Markdown Guide

For any further assistance, feel free to open an issue or contact the maintainer.