ChatGPT

Unified File Format Metadata Schema (JSON)

Below is a comprehensive JSON structure that captures the metadata, data fields, and properties of various file types (HTML webpage, image, audio/sound file, markdown document, PDF, CSV, plain text, and JavaScript code). It groups common properties (like file name, size, type, encoding, permissions) which are shared across file types, and also includes file-specific details. The schema is organized hierarchically with nested groups to reflect different layers (from high-level human-readable content down to low-level binary representation), and it highlights relationships between file types. Common *structural metadata* fields such as file size, dimensions, format, creation date, and owner are included for each file type [1]. Each file type also has *descriptive* metadata (e.g. titles, authors, tags) and *technical* metadata specific to its format (e.g. image EXIF tags, audio ID3 tags, PDF info fields like Creator/Producer [2]). In addition, the JSON captures how these files map from human concepts to computer representations — for example, images store color data (related to real-world light), audio files encode sound waves (real-world physics), text and markdown represent natural language (for language processing), etc. We also list cross-type relationships (e.g. HTML embedding images or scripts, Markdown converting to HTML, PDF encapsulating images) to show an interconnected graph of these file formats. The output is one unified JSON structure encompassing all these details.

File Types Metadata and Relationships


```
{
  "file_types": {
    "HTML": {
      "commonProperties": {
        "class": "Markup Document (HTML)",
        "extension": ".html",
        "mimeType": "text/html",
        "encoding": "text (usually UTF-8)",
        "category": "textual"
      },
      "specificProperties": {
        "structure": {
          "doctype": "e.g. HTML5 doctype '<!DOCTYPE html>'",
          "head": {
            "metadataTags": [
              "title",
              "meta",
              "link",
              "script references"
            ],
            "scriptsCount": "number of <script> tags (e.g. 3)",
            "stylesheetsCount": "number of CSS <link> tags (e.g. 2)"
```

```
          },
          "body": {
            "elements": "DOM tree of content elements (paragraphs, images,
links, etc)"
          }
        },
        "metadata": {
          "title": "Document title from <title> tag",
          "language": "language from <html lang> attribute",
          "author": "author name if specified in meta",
          "keywords": "list of keywords if any meta keywords",
          "description": "meta description content"
        },
        "content": {
          "textContent": "Visible text content of the page (strings)",
          "links": "list of hyperlinks (URLs) present",
          "embeddedMedia": "e.g. images, audio or video embedded via tags"
        },
        "fileSystem": {
          "owner": "user or system account owning the file",
          "permissions": "file permissions (e.g. rw-r--r--)",
          "visibility": "e.g. visible or hidden (if filename starts with . on
Unix)"
        },
        "sizeBytes": "size of file in bytes (depends on content length)",
        "variability": "Varies with amount of content (text and embedded
resources references). Generally small to MBs.",
        "specialCharacters": "Uses markup syntax (<, >, & etc. require
escaping). Unicode text allowed.",
        "naming": "Typically lowercase, alphanumeric and hyphens; .html
extension required. Web URLs may percent-encode special chars.",
        "checksum": "e.g. MD5 or SHA256 hash of the file for integrity
verification",
        "environment": {
          "architecture": "Web client-server model (needs HTTP server and
browser to render)",
          "infrastructure": "Delivered over HTTP/HTTPS [3] ; interpreted by
browser engine; can include JS/CSS for dynamic behavior."
        },
        "semantics": "Structured hypertext document. Intended for human reading
(text, images) and interaction (links, forms).",
        "informationTheory":
"Text-based format (highly compressible; redundancy in tags). Structure adds
semantic meaning on top of content data.",
        "realWorldMapping": "Represents formatted information (text, media) as
seen on a webpage. Links form a graph (web topology)."
      }
    },
```

```
    "Image": {
      "commonProperties": {
        "class": "Image File (Raster or Vector)",
        "extensions": [".jpg", ".png", ".gif", ".bmp", ".svg"],
        "mimeType": "image/* (format-specific e.g. image/jpeg)",
        "encoding": "binary (compressed or raw)",
        "category": "media"
      },
      "specificProperties": {
        "format": "Example: JPEG (Joint Photographic Experts Group standard)",
        "structure": {
          "header": "file header with format signature (e.g. 0xFFD8FFE0 for
JPEG, 0x89504E47 for PNG) 4 ",
          "metadataBlocks": "EXIF metadata segment (for JPEG/TIFF), or PNG
chunks for metadata",
          "pixelData": "pixel array data (possibly compressed)",
          "colorModel": "e.g. RGB, CMYK, Grayscale"
        },
        "metadata": {
          "width": "image width in pixels",
          "height": "image height in pixels",
          "colorDepth": "bits per pixel (e.g. 24-bit color)",
          "resolution": "dpi or ppi if available (metadata)",
          "EXIF": {
            "cameraMake": "Camera manufacturer (if photo)",
            "cameraModel": "Camera model",
            "captureTime": "Date/Time image taken",
            "gpsLocation": "GPS coordinates (if available)",
            "orientation": "Orientation/rotation metadata"
          }
        },
        "content": {
          "visualDescription": "Represents visual information (picture/
graphics).",
          "dominantColors": "key colors in image (if analyzed)",
          "realWorldSubject": "description of depicted scene or object (if
known, not stored in file by default)"
        },
        "fileSystem": {
          "owner": "user or app that created/saved the file",
          "permissions": "e.g. rw-r--r--",
          "visibility": "normal or hidden file"
        },
        "sizeBytes": "file size in bytes (depends on dimensions and
compression)",
        "variability": "Highly variable (from a few KB for icons to many MB for
high-res photos).",
        "specialCharacters": "Binary content (not directly human-readable).
```

```
Filename may have no spaces or be URL-encoded for web.",
        "naming": "Uses extension to indicate format (.jpg, .png). Otherwise
free-form name.",
        "checksum": "integrity hash (MD5/SHA) to verify file not corrupted",
        "environment": {
          "architecture":
"Rendered by image viewer or web browser; uses CPU/GPU to decode and display.",
          "infrastructure": "Can be embedded in documents or delivered via web.
Often stored in binary form in file systems or databases."
        },
        "semantics": "Static visual media. Conveys information via images
(colors, shapes) to human vision (relies on real-world light perception).",
        "informationTheory": "Often compressed (to reduce redundancy in pixel
data). High entropy if raw; compression algorithms (lossy) exploit human visual
perception limits [5].",
        "realWorldMapping": "Encodes real-world scenes or graphics as pixel
data. Color values correspond to physical light intensities/colors."
      }
    },
    "Audio": {
      "commonProperties": {
        "class": "Audio File",
        "extensions": [".mp3", ".wav", ".flac", ".aac"],
        "mimeType": "audio/* (e.g. audio/mpeg for MP3)",
        "encoding": "binary (typically compressed for formats like MP3/AAC, or
uncompressed PCM in WAV)",
        "category": "media"
      },
      "specificProperties": {
        "format": "Example: MP3 (MPEG-1 Audio Layer III) or WAV (RIFF Waveform
Audio)",
        "structure": {
          "header": "File header or ID3 tags (e.g. 'ID3' at start of MP3 for
metadata) or RIFF header for WAV",
          "dataChunks": "audio frames or samples data (MP3 frames, WAV 'data'
chunk)",
          "metadataTags": "ID3v2 metadata frames (for MP3) or tags like title,
artist, album"
        },
        "metadata": {
          "duration": "length of audio (seconds)",
          "bitrate": "bits per second (if compressed, e.g. 320 kbps)",
          "sampleRate": "samples per second (Hz, e.g. 44100)",
          "channels": "number of audio channels (e.g. 2 for stereo)",
          "bitDepth": "bits per sample (for PCM, e.g. 16-bit)",
          "ID3": {
            "title": "Song title",
            "artist": "Artist name",
```

```
            "album": "Album name",
            "year": "Year",
            "genre": "Genre",
            "track": "Track number"
          }
        },
        "content": {
          "audioWaveform": "Digital representation of sound waveform (amplitude
over time)",
          "spectralData": "Frequencies present (can be derived via FFT for
analysis)",
          "perceivedContent": "Music, speech, or sound effect that humans hear
when played"
        },
        "fileSystem": {
          "owner": "creator or source (if downloaded, the user who saved it)",
          "permissions": "e.g. rw-r--r--",
          "visibility": "normal/hidden"
        },
        "sizeBytes":
"depends on duration and compression (e.g. ~1 MB per minute at 128 kbps MP3)",
        "variability":
"Varies with length and quality. Uncompressed audio (WAV) is large, compressed
(MP3) is much smaller for same duration.",
        "specialCharacters": "Binary file. No textual content except metadata
tags. Filenames often avoid special chars or use standard charset.",
        "naming": "Uses extension (.mp3, .wav) to indicate type. Filename may
include track info, etc.",
        "checksum": "hash for file integrity (commonly used when storing or
transferring)",
        "environment": {
          "architecture": "Requires audio codec/decoder (software or hardware)
to play. Output via speakers (digital-to-analog conversion).",
          "infrastructure":
"Stored as files or streams; played in media players or embedded on webpages via
<audio> tag."
        },
        "semantics": "Auditory media. Represents sound (music, voice, etc.) for
human ears, corresponding to pressure waves in air (real-world physics of
sound).",
        "informationTheory": "Often compressed with lossy algorithms exploiting
psychoacoustics. Raw audio has high data rate (e.g. 10 MB/min uncompressed);
compression reduces redundancy.",
        "realWorldMapping": "Encodes sound waves digitally. Sample values
correspond to air pressure fluctuations that produce sound when converted via
speakers."
      }
    },
```

```
    "Markdown": {
      "commonProperties": {
        "class": "Markdown Document (Text)",
        "extension": ".md",
        "mimeType": "text/markdown",
        "encoding": "text (UTF-8 typically)",
        "category": "textual"
      },
      "specificProperties": {
        "structure": {
          "syntaxElements": "Plain text with markup syntax (e.g. '#' for
headings, '*' for lists)",
          "sections": "Divided into headings, paragraphs, lists, etc. based on
syntax",
          "frontMatter": "Optional YAML front-matter at top for metadata
(between --- lines)"
        },
        "metadata": {
          "title": "Possible title (if present in front-matter or first
heading)",
          "author": "If provided in front-matter",
          "date": "If provided (in front-matter)",
          "otherMeta": "Any other custom metadata fields in front-matter"
        },
        "content": {
          "rawText": "The markdown text including syntax characters",
          "renderedHTML": "Output HTML when markdown is parsed (not stored in
file, but generated)",
          "links": "links included in markdown (e.g. [text](url))",
          "embeddedResources": "Images or media linked via markdown syntax"
        },
        "fileSystem": {
          "owner": "file owner",
          "permissions": "e.g. rw-r--r--",
          "visibility": "normal/hidden"
        },
        "sizeBytes": "depends on length of text content (usually small as it's
just text)",
        "variability":
"Varies with content length. Markdown files are generally human-written, so
rarely extremely large.",
        "specialCharacters": "Uses special punctuation for formatting (#, *, -,
etc.). These must be escaped if intended literally.",
        "naming": "Typically .md extension. Filename can be any valid text
filename; often README.md or similar convention for documentation.",
        "checksum": "hash for integrity if needed",
        "environment": {
          "architecture":
```

```
"Converted by markdown parsers (software libraries) to other formats like HTML; viewed in text editors or rendered by platforms (e.g. GitHub).",
          "infrastructure": "Often used in code repositories and static site generators. Plain text, so it can be easily stored, versioned, and shared."
        },
        "semantics": "Lightweight markup for text formatting. Meant to be readable as plain text and also renderable to formatted output for human reading.",
        "informationTheory": "Plain text with simple syntax, easily compressible. Adds minimal overhead to content (format symbols).",
        "realWorldMapping": "Represents documents (e.g. README, notes) that can be rendered to HTML/PDF for human consumption. The text content carries natural language information."
      }
    },
    "PDF": {
      "commonProperties": {
        "class": "PDF Document (Portable Document Format)",
        "extension": ".pdf",
        "mimeType": "application/pdf",
        "encoding": "binary (contains compressed text/graphics)",
        "category": "document"
      },
      "specificProperties": {
        "format": "PDF version (e.g. 1.7)",
        "structure": {
          "header": "PDF file header with '%PDF' magic number [4] and version",
          "bodyObjects": "Collection of objects (fonts, images, text streams, pages, etc.)",
          "crossReferenceTable": "Index of object locations in file (or xref stream in modern PDFs)",
          "trailer": "File trailer with pointers and EOF marker '%%EOF'"
        },
        "metadata": {
          "title": "Document title (Title field in PDF info)",
          "author": "Author of document",
          "creator": "Software or origin that created PDF (Creator field) [2]",
          "producer": "PDF generation software (Producer field) [2]",
          "creationDate": "Date created",
          "modDate": "Date last modified",
          "keywords": "Keywords/tags",
          "encrypted": "Whether the PDF is password protected or encrypted"
        },
        "content": {
          "pages": "Array of pages, each with text, images, vector graphics in a fixed layout",
          "textContent": "Text content of pages (extractable if not outlined)",
          "images": "Embedded images or graphics within pages",
```

```
          "annotations": "Comments, form fields, links, or other annotations if
present"
        },
        "fileSystem": {
          "owner": "owner of the .pdf file on disk",
          "permissions": "file system permissions (and PDF can have internal
access restrictions, e.g. no-print)",
          "visibility": "normal/hidden file"
        },
        "sizeBytes": "varies from small (a few KB for text-only) to very large
(hundreds of MB for image-rich or scanned PDFs)",
        "variability": "Highly variable; content can include images (large) or
just text (small). Linearized PDFs optimize incremental web loading.",
        "specialCharacters": "Binary file; mostly not human-readable. Contains
some readable text (object definitions) and binary streams for images/fonts.",
        "naming": "Usually .pdf extension. Filename often indicates document
content. Can use any characters, but avoid reserved filesystem characters.",
        "checksum": "hash value for verifying file integrity (especially when
archiving or transferring)",
        "environment": {
          "architecture": "Platform-independent document format. Requires PDF
reader software (e.g. Adobe Reader) to display; uses system fonts/rendering for
output.",
          "infrastructure": "Used for final-form document exchange. Often
downloaded or emailed. Can be served on the web and viewed in-browser via
plugins or native support."
        },
        "semantics": "Fixed-layout document for faithful viewing/printing.
Represents text and graphics exactly as intended, independent of software or
device.",
        "informationTheory": "Internally compresses content streams (e.g. text
and images with Flate/ZIP compression). Preserves content structure but
optimized to reduce file size.",
        "realWorldMapping": "Digital equivalent of paper documents. Meant to
mirror printed page layout, encapsulating all resources (text, fonts, images)
for consistency."
      }
    },
    "CSV": {
      "commonProperties": {
        "class": "CSV Data File (Comma-Separated Values)",
        "extension": ".csv",
        "mimeType": "text/csv",
        "encoding": "text (often UTF-8 or ASCII)",
        "category": "data"
      },
      "specificProperties": {
        "structure": {
```

```
          "delimiter": "Character separating fields (e.g. comma, semicolon)",
          "textQualifier": "Quote character for fields containing delimiter or
special chars (e.g. \")",
          "newline": "Record separator (e.g. CRLF or LF)",
          "columns": "Number of fields/columns per record",
          "hasHeader":
"Boolean indicating if first line is a header with column names"
        },
        "metadata": {
          "columnNames": "List of column names (if header present)",
          "recordCount": "Number of data rows",
          "fieldTypes": "Optional data types of columns if known (otherwise all
values are treated as text)"
        },
        "content": {
          "dataPreview": "Example records as arrays of values",
          "allText": "Entire CSV content as one text blob (lines separated by
newline, fields by delimiter)"
        },
        "fileSystem": {
          "owner": "owner of the file",
          "permissions": "e.g. rw-r--r--",
          "visibility": "normal/hidden"
        },
        "sizeBytes": "depends on number of rows and columns (text size). Large
datasets can be MBs or GBs.",
        "variability": "Highly variable. Scales with number of records. Very
large CSVs are hard to handle in memory and may need database or streaming.",
        "specialCharacters": "Newlines, delimiter (comma), and quotes are
special in CSV. These must be escaped or quoted if they appear in field data.",
        "naming": "Filename usually indicates content (e.g. data.csv). .csv
extension. Avoid using the delimiter character in the file name.",
        "checksum": "hash for verifying content integrity if needed, especially
for large data files",
        "environment": {
          "architecture": "Opened by spreadsheet software or data processing
tools. Simple format; can be parsed with basic I/O and string processing.",
          "infrastructure": "Widely used for data exchange. Often loaded into
databases or analysis tools. Plain text means easy version control for smaller
files."
        },
        "semantics": "Tabular data in text form. Each line is a record, fields
separated by a delimiter. Carries structured data for analysis or import into
databases/spreadsheets.",
        "informationTheory": "No inherent compression (plaintext). Redundancy
depends on data (often many repeated values, so external compression like zip is
effective).",
        "realWorldMapping":
```

```
"Represents spreadsheet-like or database table data. Real-world entities (e.g.
people, transactions) are listed as rows with attributes in columns."
      }
    },
    "PlainText": {
      "commonProperties": {
        "class": "Plain Text File",
        "extension": ".txt",
        "mimeType": "text/plain",
        "encoding": "text (ASCII, UTF-8 or other)",
        "category": "textual"
      },
      "specificProperties": {
        "structure": {
          "contentType":
"Unstructured sequence of characters (may be divided by lines)",
          "newlineConvention": "Line break format (LF for Unix, CRLF for
Windows, etc.)"
        },
        "metadata": {
          "language": "Natural language of text if known (not stored in file,
implied by content)",
          "characterCount": "Total number of characters in file",
          "lineCount": "Number of lines (if applicable)"
        },
        "content": {
          "textSample": "Sample of text content (e.g. first 100 characters)",
          "fullText": "Entire text content (could be stored externally if huge)"
        },
        "fileSystem": {
          "owner": "owner of the file",
          "permissions": "e.g. rw-r--r--",
          "visibility": "normal/hidden"
        },
        "sizeBytes":
"proportional to number of characters (e.g. 1 byte per char for ASCII; Unicode
chars may use multiple bytes)",
        "variability": "Varies with content length. Plain text can range from
bytes (notes) to many GB (large logs).",
        "specialCharacters": "Only human-readable characters and common
whitespace. Control characters (non-printable) are possible but not typical.",
        "naming": "Any name with .txt extension by convention. Usually simple
names; extension indicates human-readable text content.",
        "checksum": "hash if needed (commonly used to verify large text file
integrity)",
        "environment": {
          "architecture":
"No special software needed; any text editor on any OS can open. Handled via
```

```
standard file I/O in all systems.",
          "infrastructure": "Used universally for notes, logs, configs. Easily
emailed or shared. No specialized infrastructure beyond storage."
        },
        "semantics": "Human-readable text. Could be plain prose, code, config,
etc. Meaning depends on context (natural language or structured data in text
form).",
        "informationTheory":
"Plain encoding of information with no compression. Often redundant (especially
natural language text), so external compression is effective.",
        "realWorldMapping":
"Direct representation of written characters or data in a form humans can read
(analogous to ink on paper, but without formatting)."
      }
    },
    "JavaScript": {
      "commonProperties": {
        "class": "JavaScript Source Code",
        "extension": ".js",
        "mimeType": "text/javascript (or application/javascript)",
        "encoding": "text (UTF-8)",
        "category": "executable code"
      },
      "specificProperties": {
        "structure": {
          "syntax": "Structured code (ECMAScript syntax: variables, functions,
objects, classes, etc.)",
          "moduleType": "Module vs script (ESM with import/export or classic
script without import)",
          "dependencies": "List of imported modules or libraries (from import/
require statements)",
          "AST": "Abstract Syntax Tree representation (nodes for statements,
expressions, etc.)"
        },
        "metadata": {
          "linesOfCode": "Number of lines in the file",
          "hasShebang": "If starts with #! (for executable scripts on Unix)",
          "usesStrict": "Whether 'use strict' mode is enabled",
          "library": "If this file is a known library, its name/version (if
identifiable from comments or file name)"
        },
        "content": {
          "code": "The raw source code as text",
          "comments": "Comments in the code (documentation or annotations)",
          "exportedFunctions": "Functions or variables exported (if module)",
          "sideEffects":
"Summary of what the code does when executed (if known, e.g. modifies DOM,
performs calculations)"
```

```
        },
        "fileSystem": {
          "owner": "owner of the file",
          "permissions": "e.g. rw-r--r-- (executable bit may be set if intended
to run via Node/shebang)",
          "visibility": "normal/hidden"
        },
        "sizeBytes": "Typically small (KBs to few MB for large libraries) as
source text",
        "variability":
"Varies with code complexity. Can be minified (whitespace removed) for
optimization in web delivery.",
        "specialCharacters": "Uses punctuation for syntax ({};() etc.). Unicode
allowed in strings or comments. Avoid non-UTF-8 bytes in source.",
        "naming": "Usually .js extension. ES modules may use .mjs. Filenames
often reflect module purpose (e.g. `helper.js`).",
        "checksum": "hash can verify code integrity (important for caching or
security of third-party scripts)",
        "environment": {
          "architecture": "Executed by JS engines (browser or Node.js). JIT-
compiled to machine code at runtime by the engine (e.g. V8).",
          "infrastructure": "In browsers, loaded via <script> or bundlers; in
Node, loaded from filesystem or via package managers. May rely on specific
runtime APIs (DOM, Node fs, etc.)."
        },
        "semantics": "Executable logic in text form. Defines program behavior/
algorithms that a machine executes. Human-readable to developers, specifying
instructions for computers.",
        "informationTheory": "Source code is structured text; contains patterns
(keywords, repeats of identifiers). Moderate compressibility (minification and
gzip reduce size significantly).",
        "realWorldMapping": "Represents dynamic behavior in software. The code,
when run, produces effects in the real world (calculations, UI changes, network
calls) via the computer."
      }
    }
  },
  "relationships": [
    {
      "type1": "HTML",
      "type2": "Image",
      "relationship": "HTML can embed images (e.g. <img> tags referencing image
files)"
    },
    {
      "type1": "HTML",
      "type2": "Audio",
      "relationship": "HTML can embed or play audio (e.g. <audio> tag
```

```
referencing sound files)"
    },
    {
      "type1": "HTML",
      "type2": "JavaScript",
      "relationship":
"HTML can reference JS files (via <script>) which then can manipulate the HTML
DOM"
    },
    {
      "type1": "Markdown",
      "type2": "HTML",
      "relationship": "Markdown is often converted to HTML for rendering"
    },
    {
      "type1": "Markdown",
      "type2": "Image",
      "relationship": "Markdown can embed images (via ![alt](url) syntax) which
become <img> in HTML"
    },
    {
      "type1": "PDF",
      "type2": "Image",
      "relationship": "PDF can embed images inside pages (for graphics or
scanned content)"
    },
    {
      "type1": "PDF",
      "type2": "PlainText",
      "relationship": "PDF contains text content (in a binary wrapper, text can
be extracted or embedded)"
    },
    {
      "type1": "CSV",
      "type2": "PlainText",
      "relationship": "CSV is a structured plain text format (values separated
by commas and newlines)"
    },
    {
      "type1": "JavaScript",
      "type2": "JavaScript",
      "relationship": "JS files can import/require other JS modules (forming a
dependency graph of code files)"
    },
    {
      "type1": "Image",
      "type2": "PlainText",
      "relationship": "Images can be encoded as text (e.g. Base64 data URI) for
```

```
embedding in text-based files"
    },
    {
      "type1": "Audio",
      "type2": "PlainText",
      "relationship":
"Audio can be represented as text by encoding (e.g. Base64 or CSV of sample
values, though not common for manual use)"
    }
  ],
  "categories": {
    "textual": ["HTML", "Markdown", "PlainText"],
    "media": ["Image", "Audio"],
    "document": ["PDF"],
    "data": ["CSV"],
    "code": ["JavaScript"]
  }
}
```

Explanation: The JSON above is structured to show **common properties** (under `"commonProperties"`) that appear for all files (like `extension`, `mimeType`, `sizeBytes`, `permissions`, etc.) and **specific properties** unique to each format (under `"specificProperties"`). For example, images have an `"EXIF"` metadata section for camera info, audio files have `"ID3"` tags (title, artist, etc.), PDFs have fields like `"creator"` and `"producer"` in their metadata [2], etc. We also include low-level details such as file headers/signatures in hex (e.g. PDF starts with `%PDF` which is `25 50 44 46` in hex [4], and PNG starts with `89 50 4E 47 ...` [4]) to illustrate binary structure, as well as high-level semantic info (like an HTML file's human-readable text or a JavaScript file's code logic). The `"environment"` sections describe how each file type fits into computing systems (for instance, HTML in a web client-server context [3], images requiring display hardware, audio needing a sound decoder, etc.). The `"relationships"` array at the end captures **interconnections** between file types (forming a graph) – for example, HTML pages containing images and scripts, Markdown converting to HTML, PDFs encapsulating images and text, and so on. The `"categories"` grouping shows clusters of file types by their nature (textual, media, document, data, code), indicating how different formats can be conceptually grouped. This unified JSON schema thus spans from human-facing content down to machine-level representation for a variety of file types, highlighting commonalities and unique traits across the spectrum of formats.

---

[1] The Complete Guide to Metadata
https://www.resourcespace.com/complete-guide-to-metadata

[2] Everything you wanted to know about media metadata, but were afraid to ask
https://freedom.press/digisec/blog/everything-you-wanted-know-about-media-metadata-were-afraid-ask/

[3] README.md
https://github.com/spectra-gallery/spectra-715/blob/177a1ee163076d1860bfc0da4b16c275cf551be3/README.md

[4] Simple Reference - List of File Signatures
https://callumsalter.com/SimpleReference/TableFileSignatures.html

5 Lossy Data Compression: JPEG
https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/index.htm