An analysis of everything returned by the PS API

- id is **required**
- PatchStorage, as far as I can tell, uses a database trigger to set the ID of the most recently uploaded patch to the id of the previous patch + 1. Of course, this accounts for all patches on PatchStorage, not just ZOIA patches.
- It seems the first patch started at id 105188 (for ZOIA), but 100000 was most likely the first patch id used on the site. The most recent ZOIA patch takes on the ID 124614, which means there have been 24614 patches uploaded to PatchStorage. Given the current number of patches currently on PatchStorage, this does not make sense. My theory is that many patches are uploaded, removed, modified, etc. Updates seem to cause the database trigger to fire (which seems odd, since it could just take on the previously assigned id).
    - A 6-digit unique identifier used to distinguish patches stored on PatchStorage
    - Min: 6, Max: 6
    - Type: Integer
    - Required: true
- self can be inferred and does not need to be apart of the metadata schema.
    - A link used to retrieve the API patch object. Takes the form of: "https://patchstorage.com/api/alpha/patches/{patchID}"
    - Type: String
    - Required: false
- link can also be inferred and would not serve a purpose within the LibraryApp unless we want to provide users the ability to view a patch on PatchStorage. It does not hurt to have it.
    - A link used to view a patch object as a page on the PatchStorage website. Takes the form of: "https://patchstorage.com/{patchName}"
    - Type: String
    - Required: false
- created_at is information that the user may be interested in. We can make it required as we can force new patches (those created and imported from a ZOIA) to take on the current time stamp at the point the system detects it to conform with the API
    - A timestamp used to represent the time at which the patch was first created. Follows the format: "YYYY-MM-DDTHH:MM:SS+00:00"
        - +00:00 seems like a time zone difference but it seems to always be retrieved as 00:00, need to investigate it a bit more.
        - Time is represented in 24-hour time, i.e. 1 PM is 13:00
    - Type: String
    - Required: true
- updated_at is information that could be displayed to the user. A patch is updated once it is first created, so this can also be made required.
    - A timestamp used to represent the time at which the patch was updated. Follows the format: "YYYY-MM-DDTHH:MM:SS+00:00"

        - +00:00 seems like a time zone difference but it seems to always be retrieved as 00:00, need to investigate it a bit more.
        - Time is represented in 24-hour time, i.e. 1 PM is 13:00
    - Type: String
    - Required: true
- slug is a non-unique name for a patch. As information is lost when the conversion from title to slug takes place, the title should be used as the main required identifier.
    - A non-unique string used to identify a patch by a name, rather than just its patch id. If the name is “Patch Name Very Creative”, the slug will be “patch-name-very-creative”. Notice whitespace is replaced by a hyphen and everything is converted to lowercase characters.
    - Type: String
    - Required: false
- title is a non-unique identifier used to refence a patch object. It serves as the source of the slug and as such, is required (I am not sure if a patch needs a name, will investigate once I get my hands on a ZOIA)
    - A non-unique string used to identify a patch object by a name, rather than just its patch id.
    - Type: String
    - Required: true
- excerpt should be dropped completely. We do not need it and have no reason to display it over the content
    - A short excerpt of the content for a given patch.
    - Type: String
    - Required: false
- content represents the patch notes a user has written for a patch. We cannot require this since patch notes cannot be written for patches that originate off a ZOIA. Users have expressed a lot of interest in being able to read patch notes so I think it would be irresponsible to completely omit it.
    - A description/patch notes for a given patch object.
    - Type: String
    - Required: false
- code is something. It has never returned anything, so I am not actually sure what it is supposed to be.
    - Returns the code associated with a patch object.
    - Type: String
    - Required: false

- files is an array of File objects. A file object contains a unique patch id, a url to the location where the file is stored, and the expected file size and name. We should use the file size and name to verify the content retrieved from PS, and the id can be used to create new metadata for any files that do not have the same id as the id schema mentioned above. Cannot be required since patches that originate from a ZOIA will not have a url.
  - An array of File objects. These file objects contain a unique patch id that mimics the id schema, a url that specifies the location of the patch, the file size and the file name.
  - Type: File
    - id: Integer (min: 6, max: 6)
    - url: String
    - filesize: Integer
    - filename: String
  - Required: false
- artwork is not something that is necessary to display within the Library App, but we can display it if the users would prefer that. We do not need to display the thumbnails users upload. We also cannot make it required since patches originating from a ZOIA will not have artwork. The way this is done on PatchStorage currently is unnecessary. They use an Artwork object that contains a url item that is a string. It could just be an artwork item that is a string, no nesting required.
  - A url that links to the uploaded artwork for the patch object.
  - Type: String
  - Required: false
- preview_url contains a link to a preview of the patch. Cannot be required since patches are not required to have a preview.
  - A link to a preview for the patch object. Typically, an audio or visual demonstration.
  - Type: String
  - Required: false
- source_code_url always returns nothing as well. Very strange. It is not needed and should be omitted.
  - A url that links to the source code for a patch object.
  - Type: String
  - Required: false
- revision exists as soon as the patch does. It would also help with version control.
  - A version number that indicates the current revision of the patch object
  - Type: String
  - Required: false

- The next few “count” attributes should be collapsed under a count object, rather than existing as their own attributes.
- comment_count is unnecessary information since we have no way of displaying comments in the Library App.
    - The total number of comments for the patch object.
    - Type: Integer
    - Required: false
- view_count is data that could be displayed and may help users to determine which patches are worthwhile.
    - The total number of views for a given patch object.
    - Type: Integer
    - Required: false
- like_count is data that could be displayed and may help users to determine which patches are worthwhile.
    - The total number of likes for a given patch object.
    - Type: Integer
    - Required: false
- download_count is data that could be displayed and may help users to determine which patches are worthwhile.
    - The total number of downloads for a given patch object.
    - Type: Integer
    - Required: false
- author is data that should be displayed to ensure patch creators are given proper credit. We cannot make it required because patches that originate from a ZOIA are “authorless”. The concept of an offer only exists on patch storage.
    - An Author object that contains the information surrounding the creation of the patch object.
    - Type: Author
        - id (unique): Integer
        - slug: String
        - name: String
    - Required: false
    - Need to find out if a patch can have multiple authors
        - Update: Only one author is permitted on PatchStorage.
- categories are data that should be displayed to the user. We cannot make it mandatory as patches that originate from a ZOIA will not initially have categories.
    - An array of Taxonomy objects. These Taxonomy objects contain the information about a category, including the id, slug, and name.
    - Type: Taxonomy
        - id (unique): Integer
        - name: String
        - slug: String
    - Required: false

- tags are data that should be displayed to the user. We cannot make it mandatory as patches that originate from a ZOIA will not initially have tags.
    - An array of Taxonomy objects. These Taxonomy objects contain the information about a tag, including the id, slug, and name.
    - Type: Taxonomy
        - id (unique): Integer
        - name: String
        - slug: String
    - Required: false
- platform is not data that we need to maintain, since we will only be checking if it is a ZOIA patch when we retrieve them from PatchStorage. We cannot make it mandatory as patches that originate from a ZOIA will not initially have a platform.
    - A Taxonomy object. This Taxonomy object contains the information about a platform, including the id, slug, and name.
    - Type: Taxonomy
        - id (unique): Integer
        - name: String
        - slug: String
    - Required: false
- state is information we could display to the user. We could offer searching and sorting options based on the state of a patch object as well. We cannot make it mandatory as patches that originate from a ZOIA will not initially have a state.
    - A Taxonomy object. This Taxonomy object contains the information regarding the state of a patch object, including the id, slug, and name.
    - Type: Taxonomy
        - id (unique): Integer
        - name: String
        - slug: String
    - Required: false
- license is information that we must display if it exists. We cannot make it mandatory as patches that originate from a ZOIA will not initially have a license.
    - A Taxonomy object. This Taxonomy object contains the information regarding the license of a patch object, including the id, slug, and name.
    - Type: Taxonomy
        - id (unique): Integer
        - name: String
        - slug: String
    - Required: false
- custom_license_text is information that we must display if it exists. We cannot make it mandatory as a patch without a custom license or patches that originate from the ZOIA will not have any custom license text.
    - The custom licensing agreement for a patch object.
    - Type: String
    - Required: false

Proposed schema for patches that are displayed once the user starts up the application and is connected to the internet. (The is the case where all ZOIA patches are retrieved from PatchStorage and displayed for the user to browse and download).

- Located in BaseSchema.json

- This schema omits anything that is not necessary to display. We should not be storing this information as a json file, but rather passing it from the model to the view as a json object. This ensures that the only json files the user has on their system related to the Library App deal with patches they have chosen to download. Of course, since we are passing data around, we need to keep the schema as concise as possible to maintain efficiency. Attributes such as artwork, platform, comment_count, source_code_url, code, and revision are not included as the frontend does not need to be aware of such attributes. The attributes self, link, and slug can all be inferred by other attributes and were also omitted. Files is included because the reference to where to retrieve the file should be maintained and unfortunately, the id for a file can differ from the id for a patch object.

- This schema is meant for validation purposes and not for the creation of json files.

Proposed schema for patches that are downloaded from PatchStorage, imported via a filesystem, or transferred from an SD card to the Library App.

- Located in MetaSchema.json

- I am very unsure on this schema. Ideally it should be kept close to the actual json information retrieved from PS, but we also do not need all the information returned. For now, I am continuing to omit artwork, platform, comment_count, source_code_url, code, slug but including files, revision, self, and link (all of which are not required except for files). Although a few of those attributes can be inferred, it is better to include them in the metadata in case the files will be accessed multiple times and the inference needs to be done more than once (which should not be the case with the BaseSchema).

- This schema is meant for validation and the creation of json files.

Proposed schema for patch banks.

- Located in BankSchema.json

- Keeps things as simple as possible, associates a slot with a patch file, all stored with the LibraryApp directory.

- This schema is meant for validation and the creation of json files.