REQUEST FOR COMMENTS

eCreation Design Toolkit

Problem & Context

The full longer-term RFC for this includes Themes and a more detailed RFC can be found here. This RFC will specifically focus on the requirements of the design toolkit so that the mStudio migration can be completed successfully.

Design data issues

The current implementation of design data poses a number of challenges to product innovation, maintenance effort and performance.

1. **Data Immutability -** currently there is no way to enforce immutability on design assets which can not change due to be used in existing creations. The assets need to be first-class citizens and should be addressable via a URI.
2. **Design Visualisation** - it is not easy to visualise what you're editing. For example, if you look at a layout defined in JSON it is very difficult to visualise what that layout looks like on a page.
3. **Data Duplication and Validation** - it is very hard to validate that data being added or modified does not already exist without searching different UUIDs and manually comparing JSON objects.
4. **Data Payload Size** - Currently when the editors (both app and web) load the design data, it has to load it in its entirety, even if there are layouts, colours and assets which will not be used for the creation being made. The design data JSON is ~1.5MB (at the time of writing).
5. **Cumbersome Process** - Manually updating JSON files is very tedious and requires time from an Engineer, even for the simplest of changes to the dataset.
6. **Filtering Designs by Products/Variants** - The ability to have designs for only for a subset of products/variants is not something that is easily achieved with the existing schema.

Considered, but out of Scope

Everything left out of the main RFC for Themes will be left out of scope with a focus being for now just on what is required for migration away from mStudio.

The full RFC can be found here:

https://docs.google.com/document/d/1bmEMsyPkr06AD4EnkVayrude7Wc9zUsJSnPdbM24eYc

Solution

High Level Architecture

The proposed architecture will be similar to what is described in the parent RFC. However, as part of the mStudio migration, this work is happening in conjunction with migration of the product data and as such the two systems should follow the same architectural patterns. It is likely that the architecture for the product data will be sorted out first and as such the design toolkit can simply replicate.

Considerations and Requirements

- Themes and Templates will need to be shared between different ranges, products, variants, planboards and ratios.
- We need to ensure users upload assets which have been optimised for web and mobile use. (lower priority).
- The generated JSON needs to be validated against the design assets data model defined in ecom-creation-data-model.
- Certain templates will have localised text labels, these need to be generated in the blank creation service and the content needs to be pulled from somewhere. Most likely ContentStack for translations.

- All data updates should be versioned to allow previews of new data in production. When approved, the current live version can be updated to the latest version. This also allows Apps to target specific design versions for specific release so updates do not break existing builds. This logic also ties directly into the publishing workflow. The new versioned data can be previewed into production before it is made visible to users. We need to think how the system can group together "change sets" to be published.

- Themes need to be classified in such a way that we are able to only match them with products that they are compatible with.

- Assets need to be loaded via an addressable URI, once a theme has been applied the assets should work if saved to that template where the theme is changed / removed.

- Old assets should never be removed as they need to be available for older creations but we should deprecate assets we no longer wish to use.

- There should be two separate mechanisms for saving changes and publishing/releasing them.

Dependencies & Integration

The planboards currently reference specific templates. That aspect needs to be moved out of the planboard or made more general, so that the planboard can work with any number of themes without requiring detailed knowledge of them.

Both the App and Web editors will need to have some updates to pull the new data structure for collections and themes. For the first iteration I recommend we still keep the default design as it exists now so the editors can move over to the new format in a timely manner.

Another reason to keep the existing default design is that existing saved creations depend on that data. Finally, creations can be started from a theme or from blank, so we need to load different sets of data based on the selected journey.

Interfaces

The API interface is still to be defined and developed but the below outlines the current plan of thinking.

Management of design data via a GUI will be controlled through a GraphQL API. There will be a similar set of standard queries and mutations for each entity to perform basic CRUD operations (remove mutations tbc). Below are a few examples for a sample of entities.

Entity	Queries	Mutations
Collections	collections, collection	add, update
Themes	themes, theme	add, update, addtoCollection
Assets (layouts, backgrounds, palettes, colours, fonts etc..)	assets, asset	add, update, addToTheme

The design data service will be a REST api with GET methods only, much as it is now, it will include some additional parameters and filters:

Endpoint	Headers	Parameters
/collections	brand/locale	variantId
/collection/id	brand/locale	variantId, collectionId
/themes	brand/locale	variantId
/theme/id	brand/locale	variantId, themeId

Infrastructure

Base Setup

Both parts of the solution will use our standard approach, making use of serverless with API Gateway and Lambda. It will also use Cognito for Authentication.

Database

Based on the data model AWS DocumentDB will be used as the Managed Database Service.

This was also a recommendation from the DBOps team on the parent RFC.

CI/CD

GitHub Actions will be used for the pipeline.

Scale & Performance

The design data portal is strictly an internal tool and as such there are no real performance concerns on that side. That said, the solution will still use serverless technology so we can be confident it will scale nicely to the limits of Lambda and the throughput limits of the chosen database solution.

The design data service which is consumed by the clients will scale alongside Lambda using serverless architecture. On top of that, the clients go via Orchestration which caches the data for a short period of time in order to handle heavy traffic. The Web editor uses persisted queries in Apollo to fetch the data and the App editor will soon do the same, once enabled in Apps Orchestration. Improving both security and performance.

Reliability

For similar reasons listed above around scalability and performance we expect reliability to be extremely high.

Should the database go down it is not client facing as the data is published to static JSON files served directly from S3.

The API uptime will be tied to the uptime and reliability of Cloudfront, API Gateway, Lambda and S3.

Redundancy

AWS DocumentDB Backups - DocumentDB continuously backs up to S3 for 1–35 days, allowing us to quickly restore to any point within the backup retention period. DocumentDB also takes automatic snapshots of the data as part of this continuous backup process.

Along with general backups of the database we will also keep track of data changes within the data so we can track what was changed when, by whom and, if needed, revert back to an older version of the data. This is being covered by the DocumentDB RFC but it will look at how data is versioned and what what limit length of time, e.g. nothing older than 90 days.

For the customer facing design service we can also look at turning on versioning for the S3 bucket so that we can very easily and quickly recover from unintended user actions if we need to fix something in a pinch.

Monitoring & Instrumentation

This is largely non-customer facing so we will just use CloudWatch alerts and logs to monitor any issues.

It would be useful to track and know how many customers are using different themes, especially with the seasonal/licensed ones, but this tracking would need to be done in the editors.

Failure Scenarios

The worst case scenario would be that some incorrect or broken data is published to production, at the same time the database falls over meaning that we can not correct the issue. In this scenario we would still have the ability to find the data on S3 and manually fix it.

Cached and persisted queries will also protect against Lambda throttling or API Gateway temporarily being unavailable.

Security

<u>Cognito</u>

Access to the design data portal will be secured by Cognito using Okta as an Identity Provider. Users logging in via Okta will only be able to modify data if they've been granted that level of access. The system will manage user permissions using JWT scopes.

<u>WAF-as-a-Service</u>

The public facing API will be secured behind the new WAF-as-a-Service to protect against common web exploits and DDoS attacks.

Future Directions

There are a number of features mentioned in the out of scope section that we can look to add moving forward. These would all be useful additions and should be taken into consideration when building out the foundations.

The future requirements and improvements to the publishing workflow and process will be important to keep in mind when building the MVP, as to avoid a major redesign of the process. The initial MVP will have a way to first preview and then publish the data in production. This needs to be done in a way that we can improve the workflow further by:

1. Allowing production data to be synced back to development for debugging
2. Allowing branches/changesets of data updates so users are not working on top of one another

Rollout

This new version of the design data will be deployed alongside the existing design data so that existing creations do not break from design data changes. Doing it this way also means we can use feature flags to slowly rollout, test and get feedback on

Themes and the different user journeys customers are following to make use of them.

There is a fair amount to do before we get to a point where we're ready to roll something out, and whilst this is still being looked at from a top level and any estimates on level of effort will be broad, it's reasonable to think of this piece of work taking place over a quarter.

Risks & Open Questions

The biggest risk we have is producing a new data structure that does not work within the editors without big refactors. We need to make sure there is ample testing on all different types of products and design combinations. By doing a feature flag rollout it will also allow use to switch the new approach off quickly if it is causing mayhem. We should be looking to keep the published JSON schema as similar as possible to avoid editor code changes.

Alternative Approaches

An argument could be made that the piece of work to get the concept of Collections and Themes into the data could be done within the existing data structure, without the use of a database and GUI, in order to move faster with having Themes ready for the end user.

However, thinking about the level of effort it takes to maintain JSON files manually and that this effort will only increase as more data is added, it is important to think of the introduction of a database and GUI as a longer term investment to help us move even faster by providing a self service approach for teams like Product Innovation to be able to review and manage designs themselves.

Next Steps

Design and refinement sessions, initial implementation, testing and a follow up RFC around versioning and releases.

The biggest challenge will be around how we look at versioning assets and themes, certain data needs to be immutable so as not to break existing creations, whilst other parts need to be mutable to allow us to evolve quickly over time.

The will tie in with how we look at previewing releases of designs and how seasonal assets become available for only short periods of time.

The next steps involve a design session around this part of the project and some initial implementation. The solution(s) around this will be put into another (smaller) RFC to be presented.