
The Tekuma Art Curation System

{stwhite, anyati}@mit.edu

Stephen L. White and Afika Nyati

May 3, 2017

1 OVERVIEW

There has been approximately one year to date of active, ongoing technologies development at Tekuma, Inc. When development began in June 2016, very little web infrastructure existed besides the tekuma homepage¹. The first goal realized was to build a system for onboarding, cataloging, and accessing artworks in an effort to streamline the art curation process happening at Tekuma, and provide a more scalable business model. One year later, much of the infrastructure for this goal has been implemented, and is ready for innovative and more marketable applications to be built on top of this curation infrastructure. Over 100,000 lines of code exist in Tekuma's repositories, accounting for multiple web applications and server back-ends. The following details these components.

2 COMPONENTS

The infrastructure which would most help Tekumas curation process was not clear at the time development began. Design decisions were made as they arose, as Tekuma's needs and use cases were evolving with the development. Different prototypes were being tested to analyze different possible web technologies that could be used. After two weeks of prototyping, it became apparent that some preliminary specification needed to be defined in order to structure development of the

¹<http://tekuma.io>. A WordPress static website

	Name	Targeted Users	URL	Repository
A	The Artist Portal	Artists	https://artist.tekuma.io	artist-portal
B	The Business Portal	Property Owners	https://business.tekuma.io *	n/a
C	The Curator Portal	Curators	https://curator.tekuma.io	curator-portal
D	The Discover Portal	Consumers	https://discover.tekuma.io	n/a

Figure 2.1: the ABCD User Abstraction

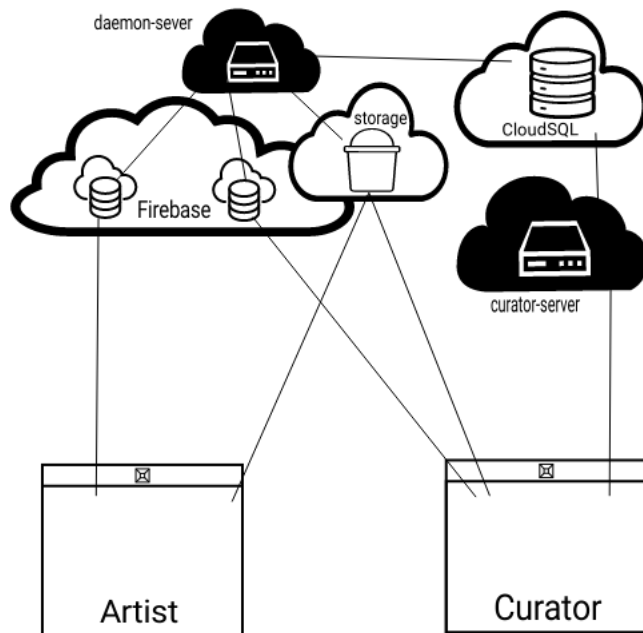


Figure 2.2: Arist-Curator System Diagram: (1) The daemon-server listens to the Firebase DBs and can insert into the SQL DB. (2) Firebase Databases hold all interface-state related data for both portals. (3) The storage bucket holds all of Tekuma’s images, and acts as a CDN to the portals. (4) The curator-server mediates queries from the curator to the SQL DB.

core infrastructure and to drive production forward. In June 2016, Stephen decided that a four-sided approach would best meet the needs of Tekuma’s complex business model, and multiple targeted users. Specifically, an underlying cloud-based system serves 4 web apps to act as portals into different parts of Tekuma’s curation system. These web apps are referred to as the ABCD user abstraction. This model is described below and summarized in figure (2.1), where each entry represents an online hosted application. All code connected to the curation system is stored in Tekuma’s GitHub. ²

²<https://github.com/tekuma>

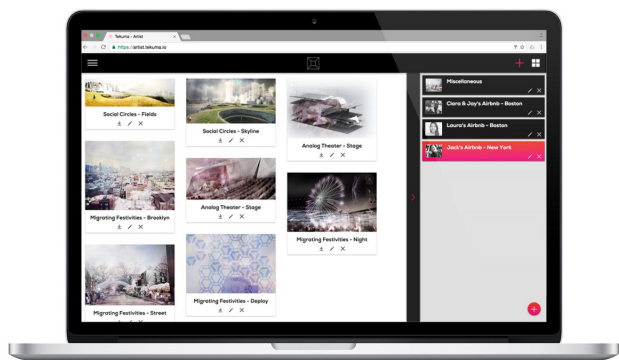


Figure 2.3: The Artist Portal

2.1 BOILERPLATE

Before creating the web applications, a general boilerplate was designed for online user interfaces. The boilerplate makes up Tekumas online graphic and visual identity. The UI primarily defaults to customs and principles dictated by Google's Material Design³ in line with the current standards in User Interface design. Also included with the boilerplate is Tekumas general CSS stylesheet, and a set of interface icons created by Afika Nyati. The style sheet builds upon the material design guidelines by customizing it to Tekumas aesthetics. More technically, the interface is written in JavaScript ES6+ using ReactJS, transpiled using webpack and BabelJS. State in the interfaces, user account creation, and file uploads are all handled by Google Firebase and managed in the Firebase Console. For simplicity, all code in the system is written in JavaScript and is either executed in-browser, or with NodeJS (v6.4.0 LTS). Together, the stylesheet, ReactJS interface components, logo, color palette and gradient, type choices (Nexa font, utf-8) makeup Tekumas online visual identity, an extrapolation of Tekumas brand identity developed by Kwaku Opoku in collaboration with Tekumas founders. (do we hold trademarks on the font usage?)

2.2 ARTIST

The first component designed and implemented was the beta version of the Artist portal. The Artist portal is powered primarily by Google Firebase, and was originally designed to function without server-side code as a statically served interface. All communications from the portal to the rest of the curation system are handled through the Firebase Realtime Database. The Artist Portal provides two main functionalities to its users, as well as a standard interface for editing and supplying personal information about themselves and their artistic style.

³<https://material.io/guidelines/>

2.2.1 ARTIST STUDIO

The primary goal of the portal is to handle uploaded and cataloging of artworks. All uploaded files are saved into Google Cloud Storage buckets, located at in the Cloud Console⁴, and all data cataloged in hierarchical (non-structural) format in the Firebase Database. The organization of these files within the storage buckets is detailed further in url-schema.md in the Documentation repository⁵. In the Studio, artists can organize their artworks into albums, edit the name and creator of the artwork, add or remove tag words, add a date of creation, as well as leave a description of the artwork. Whenever an artist is satisfied with the information supplied with the artwork, they can submit button (an up arrow) on the artwork tile. This will appear in the Curator app, explained in (X.x.x).

2.2.2 ARTIST GALLERY

The artist Gallery provides a list of all artworks which they have submitted to Tekuma. When one is clicked, more information appears, such as the review status (pending, approved, declined, or held), and any comments from the reviewing staff. Information such as sales data for that artwork can also be displayed here.

2.2.3 ARTIST ADMIN MODE

The Artist Portal also has an admin mode. This was explicitly created so that Tekuma Administrators can impersonate artists and update/submit/change artwork details on their behalf. Which users are admins is explicitly stated in the source code, and Firebase Database rules.

2.2.4 ARTIST DAEMON

Later, functionality was added to do extra processing on uploaded artworks to provide more labeled data. This functionality exceeded the scope of client-side execution, and integrating HTTP communication to a server would incur a large overhead. Instead, a daemon running on a Google Compute Engine (GCE) cloud-hosted server handles this extra processing. Communication to the daemon is facilitated by Google Firebases real-time database where a first-in-first-out (FIFO) stack is used to keep track of jobs for the daemon to execute. Jobs added to the stack have different task fields for things like resizing an image, submitting an artwork, and tagging an image. The extra labeling includes gathering text tags and color palettes from the uploaded images using machine-vision technology via the Clarif.ai API⁶. The Artist Daemon also serves another purpose. Data

⁴<https://console.cloud.google.com/storage/browser/art-uploads/?project=artist-tekuma-4a697>

⁵<https://github.com/tekuma/documentation>

⁶<https://www.clarifai.com/>

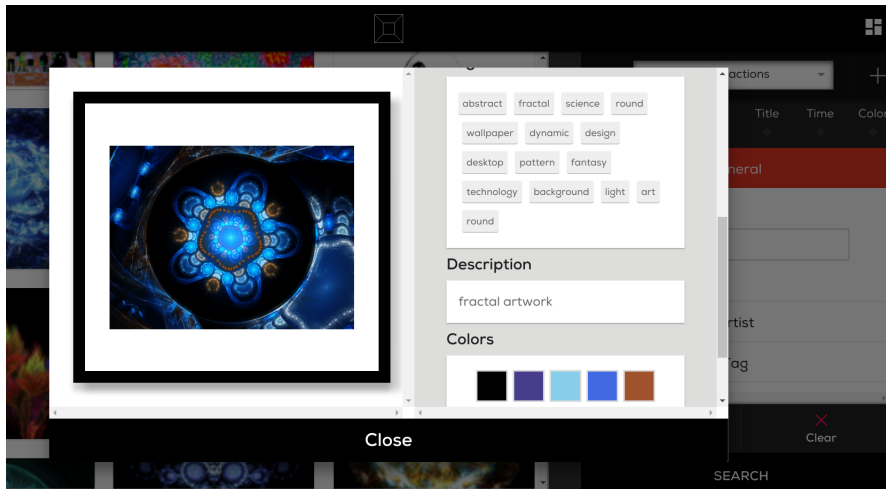


Figure 2.4: Screenshot from the Curator Portal

about submitted/approved artworks is stored in the Curator's Firebase database. This prevents any artist from having write permission to their submissions. But since the artist can not write in order to submit an artwork, it sends a job to the Artist Daemon, and the daemon submits into the Curator's Firebase DB.

2.3 BUSINESS

The Tekuma Business portal has not yet been implemented.

2.4 CURATOR

The curator portal serves as a place where Tekumas proprietary data (from the Artist Portal, or from partnerships) is stored. The curator portal also serves as an administrative portal for artworks submitted from the artist portal. The curator portal also has infrastructure to serve as an administrative system over the artist portal, by deciding what data is moved from the artist-portal database, into curator-portals database. The curator-portals database serves as Tekumas proprietary listing of curatable artwork, and therefore has more isolation than the artist-portals database. Additionally, the curator database is an SQL database, served from a Google Cloud SQL instance running MySQL. The SQL database is more efficient for searches than the NoSQL structure used by Firebase.

2.4.1 CURATOR SEARCH

The curator portal provides a very rich interface for queries which accepts words, colors, tag words, dates, and details about artists. In the figure on the right, full details about a search result are displayed. Notice the tag words, description field, and color palette associated with image. Search results can then be added to a Project, in similar fashion to how you add items to a cart in an e-commerce user experience. Queries are sent from the interface to the curator-server⁷ via HTTPS (AJAX). The server then communicates the queries to the MySQL instance, and returns the result to the interface. The server back-end uses an open-source solution (Node.js) and is served with nginx⁸.

2.4.2 CURATOR MANAGE

These projects have their own interface, where curators can: share the project with other curators, leave notes and comments on the project, download images from the project, and export all data from the project in CSV format.

2.4.3 CURATOR REVIEW

Curator's review interface acts as an administrative interface over artworks submitted from the artist portal. Artworks enter as Submitted, and can be moved to: Declined, Approved, or Held. Approved and Declined are terminal states, once the artwork is moved there it is either inserted into the SQL database, or deleted. Items moved to held allow the Artist to make changes to the information about the artwork, and re-submit it. Upon re-submitting, the artwork is removed from the held branch and added to the Submitted branch again.

2.4.4 CURATOR DAEMON

The Curator Daemon serves multiple functions. It acts as a user with access to both the Artist and Curator Firebase Database. First, when a Curator (in Review) moves an artwork into the 'held' branch, the daemon unlocks that artwork. This is because when an artist submits an artwork, it becomes locked since curators have been notified and the data needs to remain constant until the review. The Curator Daemon is also responsible for inserting newly approved artworks into the SQL database. To do this, it establishes a connection to the MySQL server in Google Cloud, and forms a transaction for all of the artwork's relevant data, and executes the transaction. If the transaction was successful, the respective (with same artwork uid) artwork object in the Approved branch of the Curator Firebase DB will have a field of 'sql: true'.

⁷<https://console.cloud.google.com/compute/instancesDetail/zones/us-east1-c/instances/curator-tekuma-1>

⁸<https://nginx.org/en/>

2.5 DISCOVER

The Discover Portal serves as the consumer-facing e-commerce store where arbitrary consumers can buy individual prints. The website is currently managed with a WordPress backend, using Shopify for handling e-commerce technologies. There is currently no proprietary infrastructure involved with the Discover Portal.

3 UNFINISHED COMPONENTS

3.1 BUSINESS

No work on the business portal has been done. Conceptually, the business portal would use the same interface boilerplate as the artist and curator portals, and would allow for business users to have a dashboard of the properties with information the curation that is happening in these properties. Currently, the functionality of the business portal is handled by sales, and automation is not crucial.

3.2 DISCOVERY

Eventually, data on the Discover portal from sales and product cataloging from Shopify (accessible via the Shopify API) should be connected a table in the SQL database. Currently, this data is kept separate from the existing infrastructure and is only accessible through scripts or user interfaces. Linking this data together would allow for more automation, and would allow sales figures to be reflected in the Artist Portal. Additionally, data should be collected on the Discover portal about which artworks each user likes, for later machine learning analysis.

4 DATABASES

4.1 ARIST DATABASE

Generally, the Firebase Databases are used to handle interface state. In the Artist Portal, it stores all the data for each user, including their information and uploaded artworks. To directly interact with the database, go to: <https://console.firebase.google.com/project/artist-tekuma-4a697/database/data>

The database has 3 branches from root, `_private`, `jobs`, and `public`. `_private` and `public` then both have an `onboarders` branch, keyed by all artist UIDs. The `jobs` branch acts as the stack for the artist daemon.

4.2 CURATOR DATABASE

The Curator Database holds all of the Curator's state data (like users, projects, etc). The Curator Firebase database also holds the 4 review branches submissions, held, approved, and declined. The Curator Firebase database can be accessed at :

<https://console.firebase.google.com/project/curator-tekuma/database/data>

4.3 TEKUMA ARTWORK DB

The SQL database contains 6 tables, (artworks, artist, associations, locations, collections, labels). Each artwork has an entry in the artworks table, keyed by the artwork UID. Each row contains:

- uid (char)
- title (char)
- description (text)
- artist_uid (char)
- date_of_addition (DATETIME)
- date_of_creation (DATETIME)
- thumbnail_url (char)
- origin (char)
- reverse_lookup (char)
- META (blob)

4.4 ART.COM DATABASE

The Art.com Database, which is roughly 10MB, contains **2,311,390** rows of data, in structured SQL format. Examples of these rows are below.

```
columns = [ 'SKU', 'ITEM_TITLE', 'ITEM_LONG_DESC', 'ARTISTNAME',  
'ITEM_TYPE', 'IMAGE_URL', 'PRODUCT_URL', 'PRODUCT_TAXONOMY' ]  
actual_row = [ '30742196502A',  
               'Spurzheim_Bust',
```


'JOHANN_KASPAR_SPURZHEIM_German_phrenologist_with_his_autograph',
 'H_Corbould',
 'Photographic_Print',
 'http://cache2.allpostersimages.com/MED/\\84\\8488\\5WQK300Z.jpg',
 'http://www.art.com/products/p30742196502-sa-i9032642/.htm?RFID=197560',
 'World_Culture>European_Cultures>German_Culture>>>>>>']

5 RELATED WORK

1. **ArtFlip** (<https://www.artflip.com/product>) A website very similar in UI and UX to the curator portal.
2. **Artsy** (<https://www.artsy.net/collect>)
3. **Canvis** (<http://canviz.co/contact/>) A mass producible digital picture frame, including 3D printer files.
4. **StitchFix** (<https://www.stitchfix.com/>) a clothing company which uses a mixture of expert human stylists and artificial intelligence to pick out outfits for customers on a subscription plan.
5. **Turning Art** (<https://www.turningart.com/gallery>) A curator company with a similar business model of directed towards office, commercial, and multi-family real estate.
6. **Large-scale Classification of Fine-Art Paintings** (<https://arxiv.org/pdf/1505.00855v1.pdf>) Details large scale classification algorithms for fine art paintings.
7. **A Neural Algorithm of Artistic Style** (<https://arxiv.org/pdf/1508.06576.pdf>) Describes methods of transferring artist styles between artworks use Convolutional Neural Nets.
8. **Inceptionism: Going Deeper into Neural Networks** (<https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>) Discussion of Google Deepdream

6 POTENTIAL INNOVATIVE DEVELOPMENT PROJECTS

6.1 COLLABORATIVE ART RECOMMENDER SYSTEM

This section details methods which are *agnostic* to the content. In other words, the AI is unaware that it is filtering artworks specifically. Collaborative filtering is currently a ubiquitous feature of internet based creative content platforms such as Spotify and Netflix. Collaborative filtering is commonly implemented with either data clustering, or low-rank matrix multiplication (better for sparse data), and relies on having a active user base. The underlying assumption of collaborative filtering is that if user (A) likes artworks a,b,c,d and user (B) likes artworks a,b,c, then it is likely he will also like artwork d as well. The strength of this method is that it is agnostic to the content, which is especially powerful for art and media, as it is inherently abstract and hard to break it down into concrete features.

6.2 TEKUMAI, CURATED RECOMMENDER SYSTEM

This section details methods which are *content-based*, meaning that the features of the artwork are directly considered by the system. This relies on the assumption that all artworks in the system have a concrete list of their features, known as a feature vector. Features in this vector have a numerical representation, and are chosen such that they partition the artworks into separate sets which people might conceivably assign different preferences to. Then, each use is assigned a vector, θ , which is continually updated to reflect the user's preferences. When this vector is initialized (usually with random values), it is not very powerful. This is known as the cold-start problem. Methods for circumventing cold start can "borrow" data from other places. Specifically, data could be borrowed from similar users, such as descbried in the previous section with collaborative filtering. The user could also be presented with a survey of his art tastes, which could be used to initialize the vector to something more accurate than if it was just set to random values. Given a feature vector $x^{(i)}$, we can predict a rating \hat{y} by:

$$\hat{y}^{(i)} = \theta \cdot x^{(i)} = \sum_{j=1}^d \theta_j x_j^{(i)}$$

It is important to note that good features are those that partition the artworks into sets that people might conceivably assign different preferences to.