GalleryGuide Technical Report

Motivation

GalleryGuide was created with the purpose of helping ordinary citizens better connect to art. Whether they've already found an artist they like and want to learn more, want to get to know more about a work that strikes a chord in them, or want to get out and explore the galleries their city or state has to offer, GalleryGuide provides information that can enrich the lives of its users.

With GalleryGuide, users can better understand what artists and artworks a particular gallery is best suited for (by looking at a Gallery instance), what effect a particular generation had on the works of art produced (by sorting artists by date of birth/death), and where to view a particular piece of art if it's on display (by looking at an Artwork instance).

User Stories

Phase I

Our Customers' User Stories

"As a business owner, it would be useful to be able to sort artists by their home cities or countries. This way I could easily feature several artists from the same place. Or I could find out if artists from the same area make similar art."

- Marked issue as Phase 3 milestone

"It would be nice to be able to sort on art style as well since it could be used as inspiration for artists wanting to break into or see certain styles of brushing or paint material like acryllic."

- Marked issue as Phase 3 milestone

"As a business owner, I love learning more about the ways in which different artists interacted with one another and displaying that information through collaborated art pieces. I would love it if there could be a connection between different artists who have collaborated together. It would make for a great attribute."

- While this could potentially be feasible, we informed the customer that we had not incorporated an API with this data in the proposal, so it is out of scope.

“As a business owner, I take copyright very seriously, and don't want to accidentally break any laws. It would be helpful for my purposes to include the copyright status of an artwork on its instance page. Additionally, it would help to make this a sortable label in a search.”

- Informed the customer that this feature would be prioritized for implementation in Phase 1.
- During phase two we prioritized scraping instances of artworks that had some form of copyright information. We stored this information alongside the artwork in our database and it is now displayed with each artwork on the website.

Our User Stories to Our Developer (NBADB)

“I love learning about the histories of different sports teams, I always wonder about when my favorite team was founded. Is there any way your website could give me that information?”

“Hey guys, If possible I would like to see a map with all the NBA team's logos near the area they play. Since this is phase one, a static map with the logos on top would be great, but maybe in later phases make it interactable? Sorry for the late notice, if this isn't possible then please consider it for phase two!”

“Add current active roster of players for teams.”

“I think it'd be interesting to see how the teams are currently doing. This could also be a *sortable field*. If the current season score isn't possible, it'd be nice to see how they typically do (based on their previous season performance).”

“I am not an NBA fan, but I want to learn more about NBA players. I heard of this dude Luka Doncic and his name is spelt kinda funny. Is there any way I can find out where he is from?”

Phase II

Our Customers’ User Stories

“I like seeing the images and learning about them. When I find an image that interest me and I like the art style, it would be nice if I could find another image similar to it. I would appreciate it you could add a related images section on each image, so that I can find related images for images I like.”

- This sounds like a great idea! The Artsy API does have a KNN feature for finding similar artworks/artists, so I think we could look into this more. After we get the data, linking should just require some parsing of the API result.

“I would love it if you could add a link in each artwork instance that could direct me to more detailed information about that specific piece. This would be nice so if I wanted to know more then I would have a way to get that information. A button or a hyperlink would work just fine!”

- We will look into the feasibility of this. Right now we don't have an easy workflow for getting a large amount of info for an artwork, but we can try to look into this.

“It would be nice to see a preview image for artists and galleries on the grid-view pages. Something similar to the images that you can see on the Artworks grid view page. This would help me identify artists and galleries at a glance.”

- “We are prioritizing scraping data of instances that have thumbnails for both artists and galleries. We should get this done in this phase 2.”
- To achieve this we only stored instances of galleries and artists which had an associated thumbnail. This way, each gallery and artist will have some image that can be displayed on the website.

“As a museum goer, I would love if there was more visually on the instance pages, but specifically the Museum page. A Google Maps insert with the location of the museum, a dynamically updated way to display the museum's current art displays, or videos of the buildings would help bring the pages to life. This would increase both the utility and appeal of the web page.”

- “We should be able to get region, or more specific location data, for each instance of a gallery from one of our API targets. This will be focused on and should be achievable in phase 2.”

Our User Stories to Our Developer (NBADB)

“Hi, I really want to make some cool designs revolving around my favorite teams, and I really like how your website gives me the colors of different teams. However, I wonder if there is a way for me to better observe which color is which. For example, according to your website the Washington Wizard use "Loblolly" but I have never heard of that before. I would love if I could see or press a link to an example of the color so I could match the name to the color.”

“Hey guys, your instances look great!. For this phase, I would like the site to use the API that you are creating for your database to display more instances. This will require you to paginate the relevant data, and display it properly. Good luck!”

“I think it would be interesting if I could see a featured player, their team, and what arena they play in on each visit to your website. As a developer of my own website, I think it would be cool if I could scrape this information from your API. This way, when someone visits my website, they will be able to see a featured player. It would be cool if the featured player was someone who popped off in a game recently; if it is the offseason, it could be a player who contributed to a big win in the finals.”

“I have so many tabs open because of all of my homework assignments! And so many of them have the same old React logo as their favicon that it's hard to tell the tabs apart. I'd love to have a basketball favicon to tell where my NBA DB tab is! 🏀”

“Including player highlights and records on your website is important because it allows fans to relive great moments and accomplishments, creating a sense of nostalgia and excitement. It also provides a way for fans to compare and contrast players across different eras and appreciate their unique talents and abilities. Finally, it helps to establish your website as a go-to source for comprehensive and in-depth coverage of the NBA and its players.”

Phase III

Our Customers’ User Stories

“I think it would be nice to change the default React symbol on the tab manager to something more suitable to your project. I think that is a good way to add personality to your site.”

- “This will roll out in the next deployment.”

“Hi, I'm an art curator and I was wondering if it was possible for me to be able have a sorted view of images based on their quality. If the sort feature could be based upon either pixel quality (ex. 1080p, 720p, etc..) or some other score, that would help out a lot.”

- “Not too sure about the feasibility of this. The images that we scrape don't come with any information about quality that is easily filterable and I don't think we will have time to develop a way to evaluate image quality.”

“Hello, I would like to be able to view and filter images based on the category an image is in. Thanks. The reason for this is because I feel like it would be easier find images similar to ones that I like, if I continue to look at images of the same type. I suppose I should use the word style instead, filter images based on the style of the image.”

- “We are implementing search by medium in this phase. This should allow you to search for all styles of art.”

“I would like for there to be images in the model cards on the home page. Additionally, there should be a splash image in the background. Both of these steps will help to make the website appear more inviting and professional.”

- “Good idea, we will see if we can incorporate some sort of background image.”

“As a user I would love it if each model instance could give me a brief taste of what I would see when I click on the instance by having a photo embedded in it. It would help me decide which instance to choose to look at. I believe it would also make the site more vibrant and colorful!”

- “This should roll out with the next deploy”

Our User Stories to Our Developer (NBADB)

“I love to introduce friends to your website to learn more about their favorite teams. However, I always noticed the home page doesn't make it clear what you can find on the website other than a short description and small link to the about page which don't stand out much at first glance. I wonder if there is a way to make it so my friends know what they can do with the website as soon as they open it, maybe images to describe your instances or keywords that catch your eye.”

“I think it would be cool if I could see where players went to college, if that is the case. I could then search for my favorite players and see where they played in college, or search for a college to find out if any of their players made it to the NBA. Some players didn't go to an American college, or took a less traditional journey to the NBA, maybe through a foreign league; it would be nice if I could maybe see this too.”

“I'd like to get to know the developer team! I really like the aesthetics of the website so far, but using the current website, the stats and images never load... I think GeoJob's repository has some copy-able code, by the way.”

“Hey guys, sorry for the short notice. I was thinking, it would be interesting to have a section for players who have some sort of historical connection to each other. As a historical example, Shaq and Kobe are considered one of the most iconic duos of all time. On a less drastic scale, Curry, Klay, and Durant were considered the core of the winning Golden State team. It would be interesting to see these connections even if these players are no longer on the same team.”

“Having player and team stats readily available is super important for any NBA fan. Not only does it allow you to compare different players and teams, but it also helps you understand the strengths and weaknesses of each player and team. Plus, it's just fun to look at the numbers and see how your favorite players and teams are performing.”

Phase IV

Our Customers’ User Stories

TODO Add what they wrote for us & our response!

Our User Stories to Our Developer (NBADB)

TODO Put in stories we wrote for nbasb

RESTful API

Documentation

Specific documentation and example responses for each endpoint can be found here: https://documenter.getpostman.com/view/18824630/2s93CExHF3

Phase II Features

Pagination

For now we have decided that each call to the gallery, artist, or artwork endpoint should return up to 9 instances of each. These instances are each a JSON object contained within a list in the response. Each

paginated response also includes a next field that contains a well formed URL that can be requested to get the next page of information. A paginated response also has a field that indicates the size of the list that the response contains i.e. how many instances are in this page.

General Methodology

We used the MVC design pattern while implementing our server. When a request reaches an endpoint in the server (controller), the server may do some preprocessing before it makes a call to the model, which is a module that implements the functionality of our database for the controller. The model will do whatever database request is required and return a result to the controller. In our system, this result is an iterator of rows that were found in the query done by the model. The controller may then do some post processing (ex. Determine which page this data is from, format a JSON response) and finally return a result to the View.

Phase III Features

Search/Filter/Sort

We decided to implement these features using parameters on our already existing model endpoints. In our design phase, we decided that it would be best to always do operations in the order of search -> filter -> sort.

To support searching, we designed a function that could, given keywords and a database table to be searched, find instances of each model that had the given keywords in any of its fields. This function returns a list of rows (dictionaries). We opted for this approach because it allowed us to have much more control over the next steps of filtering and sorting. Specifically, we had some field types that we wanted to filter or sort by that would not play nice with the built-in database operations. Highlighting for searching was done using the *react-highlight-regex* package.

To support filtering, we again designed a function that could take a list of rows, the direction of filtering, and a column name. After we had designed this function, we could use it to filter any column above or below a given value. We also support filtering by multiple values at time by simply applying this function repeatedly to results of the previous call.

Once again, we designed a function that took a list of rows, a column name, and a sort direction in order to implement sorting. We used this to sort the searched and filtered results.

Phase IV Features

Provider Visualizations

We were able to find the documentation for our developer, NBAdb, here. Since our provider had their information in both a paginated and full-response form, we were able to utilize the full results endpoint to gather all of the information in one go for our visualizations.

To visualize the distribution of the arenas in the United States, we used React Simple Maps with markers and the "all arenas" endpoint. It is important to note that we had to filter the returned data by country and ensure that the latitude and longitude fields were not null. We really appreciated the developers adding the latitude and longitude!

To visualize the relationship between player age and salary, we used the basic scatterplot from Recharts. We had to do some calculations using the birthday supplied from the API in ISO format to change it into the player's age. To do this, we used the Date JavaScript functions.

Finally, to visualize the frequency of jersey numbers among players, we created a histogram using AG Grid. We found AG Grid to be the simplest of the three provider visualizations to implement.

Models

Galleries

A gallery is a physical host of artworks and artists. Information available on our website about any particular gallery:

- Number of artworks in this gallery
- Number of artists in this gallery
- Location/Region
- Description
- Thumbnail image
- Gallery website

Artists

An artist is someone who is credited with creating an artwork. Information available on our website about any particular artist:

- Which galleries host a work by this artist
- Which works of art this artist is credited with
- Number of credited artworks
- Number of galleries that this artist is presented in
- Biography/Blurb
- Year of birth
- Year of death
- Thumbnail image

Artworks

An artwork is attributed to an artist and is hosted by a gallery. Information available on our website about any particular artwork:

- Which artist this work is attributed to
- Which gallery hosts this artwork
- Thumbnail/Image
- Title of artwork
- Date when the work was created
- Medium
- Iconicity of this work
- Copyright information about this image

Tools

Postman

- Used to test API endpoints of data sources
- Used to test our API endpoints
- Hosts documentation about GalleryGuide's API

PlantUML, PlantText

- Used to create and render, respectively, a diagram of our database that illustrates the connections between our models

Flask

- Serves our webpage
- Hosts our API

SQLAlchemy

- Interface for setting up the metadata of a database and adding data while scraping
- Interface for database when querying from API endpoints

React-Bootstrap

- CSS framework

React

- JavaScript framework for frontend development

React Simple Maps, Recharts, AG Grid

- Used for creating SVGs for provider visualizations

Data Sources

Artsy API

- Used to get general information for each model
- https://developers.artsy.net/v2/

Wikipedia API

- Used to supplement the biographies of artists
- https://en.wikipedia.org/api/rest_v1/

Google Maps API

- Used to get map for the region of a gallery

- In the future, we could also use this to get a map of the hometown of artists. This would require that we narrow our specifications while scraping.
- https://developers.google.com/maps/documentation/maps-static

YouTube API

- Used to get an informational video or more interactive experience for the gallery, artwork, or artist
- Done by performing a YouTube search (requesting a video as response) and choosing the first video element ID to embed
- https://developers.google.com/youtube/v3/getting-started

Hosting

We decided to use [AWS EC2](https://aws.amazon.com/ec2/) to host our website. We decided to use EC2 because it will eventually allow us to serve our website and API from the same machine. Additionally, having an EC2 instance will give us extra flexibility when deploying.

Setup

For our EC2 Instance, we decided to use an Amazon Machine Image running Ubuntu 22.04, t2.micro instance type, with default storage.

As for our security groups, we configured 3:

Note that all of these are using TCP protocol and allow all source connections (0.0.0.0/0)

- SSH on port 20
- HTTPS on 443
- HTTP on 80

Also, during this step, a key pair is created. For my Windows system WSL2, the private key must be moved / copied from the mounted Windows drive to ~/.ssh using

```
cp /mnt/<Windows Drive Letter>/<path>/<privatekey.pem> ~/.ssh
```

Another thing you might have to do if you're on WSL like me is to change the permissions of the ```.pem``` file in case it's too open. When that happens, you'll get this when you try to ssh into your instance.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@
Permissions <permission> for '/path/.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
```

If that's the case, then:

`chmod 600 ~/.ssh/<name_of_pem>`

This makes it so that it is only read-writable by you.

You can also do `chmod 400` instead, which makes it only readable by you (thus blocking your write access). 600 is typically better in most cases.

[More information with ssh'ing into your instance using WSL can be found here.](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide//WSL.html)

>Another note: Amazon doesn't grant root access to EC2 instances by default (for seecurity purposes). Use the sudo command to obtain elevated privileges

Next, start your EC2 Instance and grab the IPv4 Address from the running instance.

Adding DNS Records using Route53

> Route53 is Amazon's Domain Naming System (DNS) webservice

1. Create a hosted zone. A hosted zone is simply a container that holds information about how you want to route traffic for a domain.
2. Quick create an Apex Record with `www.<yourdomainname>` and set the value (what it points to) to the IP Address of your instance.

> Note that: A Records: A records map a domain name to an IPv4 address. When a client requests a website, the DNS resolver looks up the A record for the domain name, and returns the associated IP address to the client. The client then uses the IP address to access the website.
>
> CNAME Records: CNAME records map a domain name to another domain name, instead of an IP address. When a client requests a website, the DNS resolver looks up the CNAME record for the domain name and returns the associated domain name. The resolver then repeats the process for the new domain name, until it finds the A record that maps the domain name to an IP address. The client uses the IP address to access the website.

3. Create another `A` alias record for just your `<domainname>` and set your alias target to your `www.<yourdomainname>` so that both records will hit the same target when entered in a web browser.

Configuring your name servers if you purchased your domain from else where.

If you bought your domain from a third party domain registrar and want yo use Route53 to manage your DNS for your domain, then you have to change the nameservers for your domain to the ones provided by Route53.

1. Copy the NS record values from your hosted zone for your domain. For Route53, these usually look something like

```
ns-592.awsdns-18.net.
ns-1532.awsdns-34.org.
```

ns-1237.awsdns-42.co.uk.
ns-48.awsdns-89.com.
```
2. Go to your domain registar, and under Nameservers or DNS management, or something similar, and replace whatever nameservers already present with the ones from your hosted zone.
3. Note that changes can take 24 - 48 hours to propagate.

> After this, you should be able to ssh into your instance by using
>
> `ss -i ~/.ssh/<pem_file> <user>@<domain_name>`
> and in our case it would be
>
> `ss -i ~/.ssh/<pem_file> ubuntu@galleryguide.me`
```