HOMEWORK ASSIGNMENT 8

CSCI 571 – Summer 2022

Abstract

Angular, TypeScript, Bootstrap, Responsive Design, JavaScript in Server Side, Node.js, Express, AJAX, JSON, and Artsy API

Baskin Senbaslar

baskin.senbaslar@usc.edu

Homework 8: Angular, TypeScript, Bootstrap, Responsive Design, JavaScript in Server Side, Node.js, Express, AJAX, JSON, and Artsy API

1 Objectives

1. Get experience with creating backend applications using JavaScript/Node.js in server side with Express framework.
2. Get experience with using Angular, TypeScript, and Bootstrap in client side and creating responsive frontend.
3. Get experience with using HttpClientModule of Angular for AJAX.
4. Get experience with Artsy API.
5. Get experience with Google Cloud Platform (GCP).

2 Homework Description Resources

1. Homework Description Document (This document)
2. Grading Guidelines
3. Web Reference Video: https://www.youtube.com/watch?v=7gw0OByU0Bc
4. Mobile Reference Video: https://www.youtube.com/watch?v=rooC7e59Xss
5. Piazza

3 General Directions

1. The backend of this homework must be implemented in JavaScript using Node.js Express framework. Refer to Node.js website for installing Node.js and learning how to use it. Have a look at "Getting started" guides in Express website to learn how to create backend applications using Express. axios library can be useful to make requests from your Node.js backend to Artsy servers. **Implementing the backend in anything other than Node.js will result in a 4-point reduction.**
2. The frontend of this homework must be implemented using the Angular framework. Refer to Angular setup docs for installing Angular and creating Angular projects. Angular "Tour of Heroes" app tutorial is a very good tutorial to see different Angular concepts in action. **Implementing the frontend in anything other than Angular will result in a 4-point reduction.**
3. You are expected to create a responsive website. For that reason, we require you to use Bootstrap, a CSS framework for responsive, mobile-first web development. It will save you from the burden of dealing with CSS peculiarities (Some of you may have felt like this in Homework 6) and the website you create will be responsive automatically if you develop within the framework provided by Bootstrap. Please refer to Bootstrap docs for reference (especially look

at “Layout” section, and the components that you want to use). Refer to this post to learn how to add Bootstrap to Angular projects. **Not using Bootstrap will result in a 4-point reduction.**

4. The backend of this homework must be deployed to GCP. The backend should serve the frontend as well as other endpoints you may define. Please refer to Homework 7 for deploying Node.js applications to GCP.
5. You must refer to the homework description document (this document), grading guidelines, the reference videos and instructions in Piazza while developing this homework. All discussions and explanations in Piazza related to this homework are part of the homework description and grading guidelines. Therefore, please review all Piazza threads before submitting the assignment. If there is a conflict between Piazza and this description and/or the grading guidelines, answers in Piazza are valid.
6. The homework will be graded using the latest version of the Google Chrome browser. Developing homework using the latest version of Google Chrome is advised.

4 Description

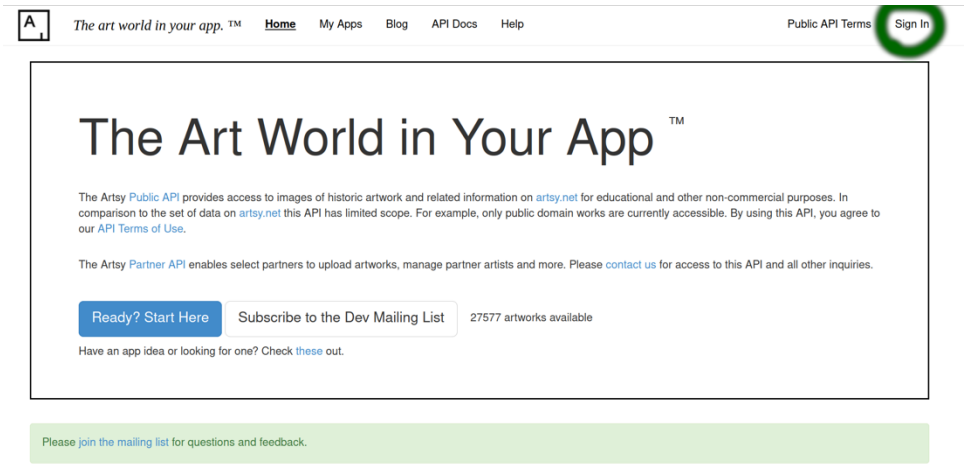
In this homework, you are going to create a web page that allows its users to search for artists using the Artsy API. The results of the search will be listed as a list of artist cards. When a user clicks on an artist card, details about the chosen artist will be shown underneath. The details will contain two tabs, one containing information about the artist (name, birthday, deathday, nationality and biography) and other containing artist’s artworks. Each artwork belongs to several categories which will be shown when the categories button below the artwork is clicked.

4-1 Artsy Developer Registration (Same as Homework 6)

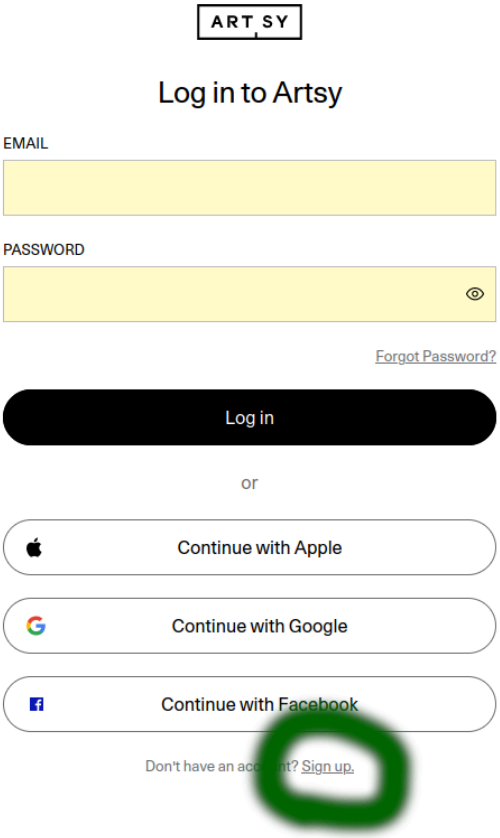
To use Artsy API, you need to create a developer account in Artsy, create an app and get a client ID and a client Secret.

In your browser, navigate to https://developers.artsy.net/.

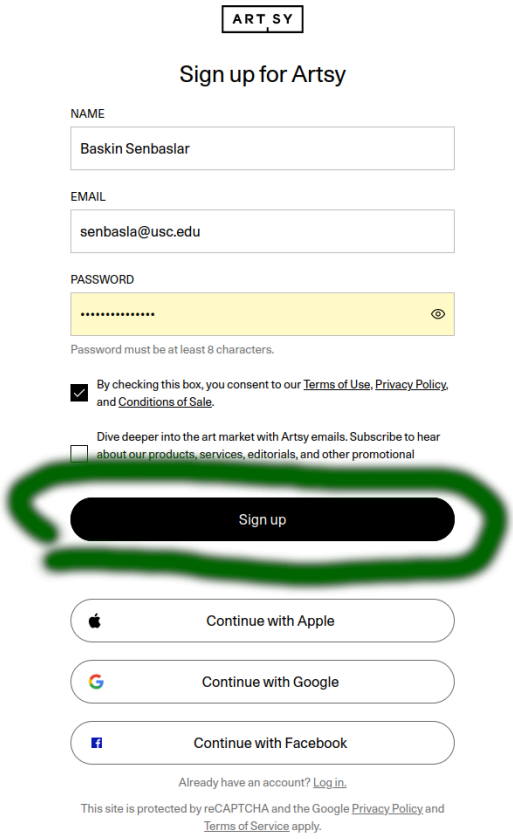
Click to the “Sign In” button on the top right of the page to navigate to login page.



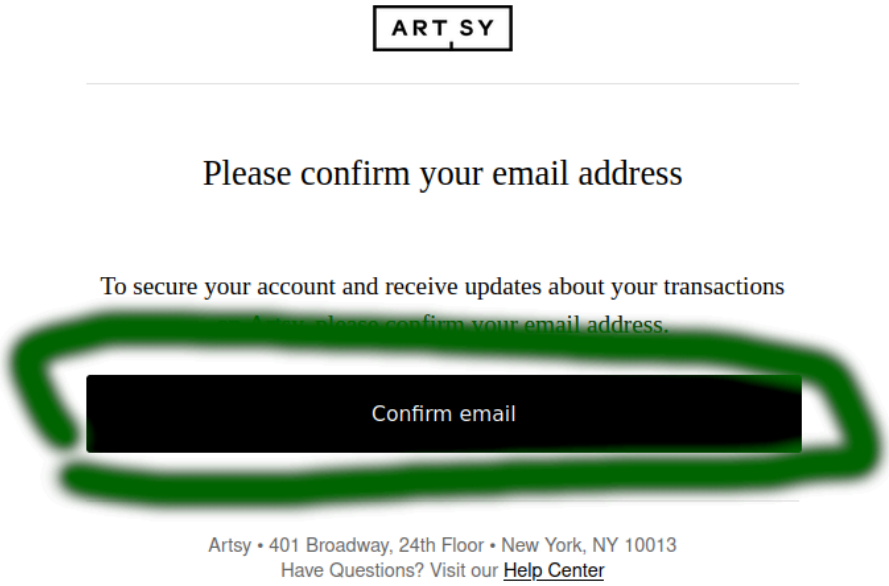
Click on the “Sign up” button below the login page.



Fill out your information in the registration form. You don't have to use your USC email. Click on the "Sign up" button.

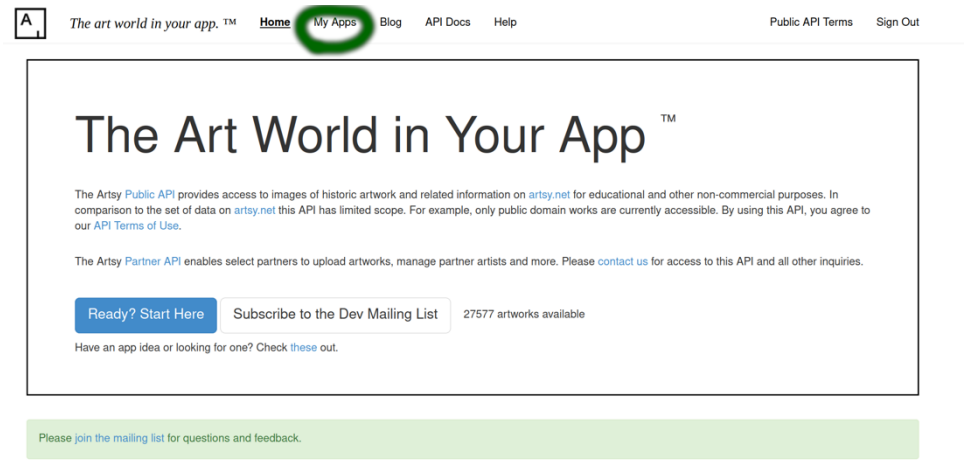


Artsy will send you a confirmation email. Click on the “Confirm email” button in the received email.

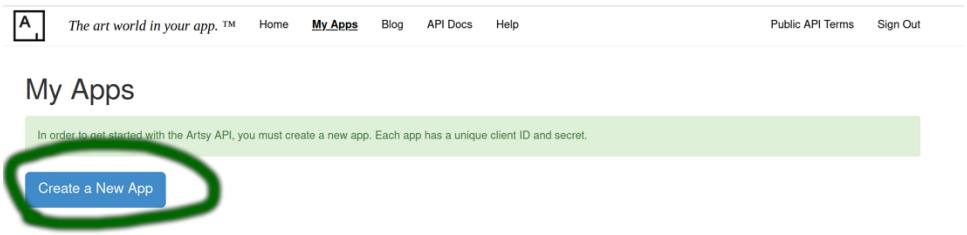


Navigate to https://developers.artsy.net/. You should be logged in. If not, you can login by clicking the "Sign In" button on the top right corner.

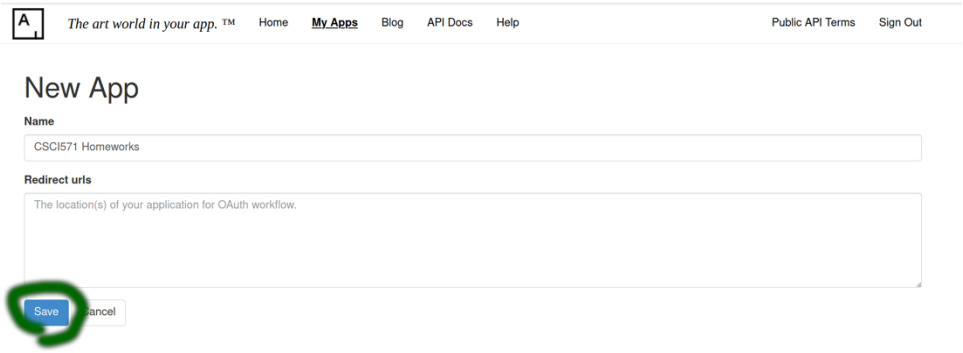
Click on the My Apps button.



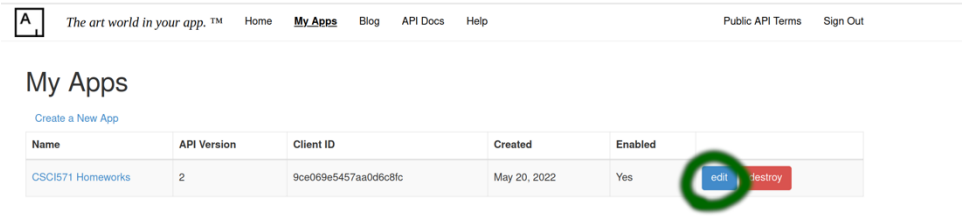
Click on the "Create a New App" button on the opened page.



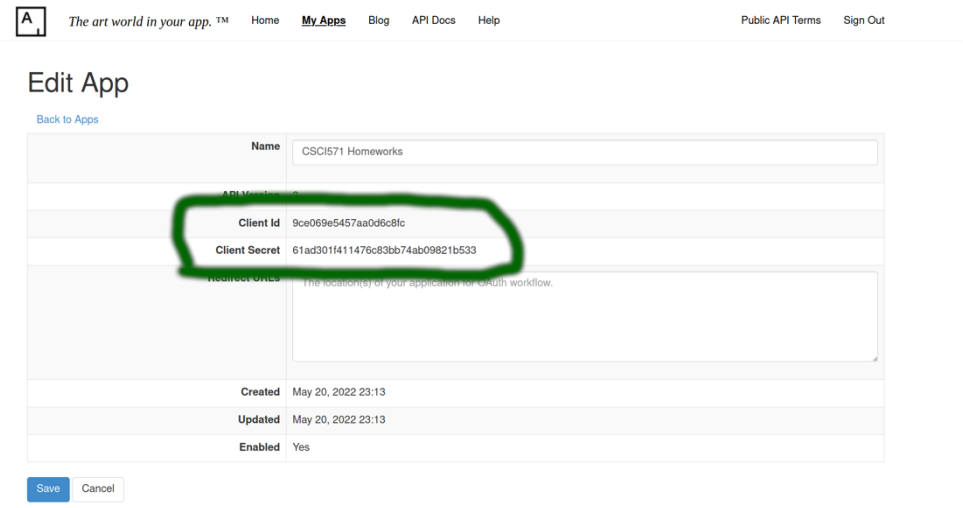
Give a name to your app (it can be anything), and leave “Redirect urls” empty. Click on “Save” button.



Your new app will be listed in the “My Apps” page. Click on the “edit” button near the app you just created.



Save “Client ID” and “Client Secret” values to a document on your machine. You will use them to make requests to Artsy.



4-2 Artsy API Endpoints

The full list and documentation of Artsy API endpoints are given in https://developers.artsy.net/v2/. In this homework, you will use 5 Artsy API endpoints: Authentication, Search, Artists, Artworks, and Genes.

Search, Artists, Artworks, and Genes endpoints requires X-XAPP-Token header, which you will get through Authentication endpoint.

4-2-1 Authentication Endpoint (Same as Homework 6)

Authentication endpoint has the base URL https://api.artsy.net/api/tokens/xapp_token. You send your request using HTTP POST method. It requires two body parameters: **client_id** and **client_secret**. Set client_id to the Client ID that is assigned to your app, and client_secret to the Client Secret that is assigned to your app, which are described in the previous section. Authentication endpoint returns a JSON response in the following form:

```
{
  "type" : "xapp_token",
  "token" : "...",
  "expires_at" : "2022-05-27T23:46:35+00:00"
}
```

The **token** field of the response will be used while accessing Search and Artists endpoints. **Notice that the token returned from the Authentication API expires in 1 week.** You can either implement a logic that refreshes the token when it expires (elegant solution) or get a new token before each request (not elegant solution). For the purposes of this homework, we will not grade how you implement this logic. You are free to do it either way.

4-2-2 Search Endpoint (Same as Homework 6)

Search endpoint allows you to search queries in Artsy DB. It has the base URL https://api.artsy.net/api/search. You send your request using HTTP GET method. It accepts query parameter **q**. You send your search query through this parameter. The maximum number of results can be set by **size** parameter, which has the maximum value of 10. In this homework, you will always set it to 10. In addition to the query parameter, you should set X-XAPP-Token header in your request to the token returned from Authentication endpoint.

Example request: https://api.artsy.net/api/search?q=picasso&size=10

The server returns a JSON result, excerpt from which is given below.

```
total_count:          7483
offset:               0
q:                    "picasso"
_links:
  self:
    href:             "https://api.artsy.net/api/search?offset=0&q=picasso&size=10"
  next:
    href:             "https://api.artsy.net/api/search?offset=10&q=picasso&size=10"
_embedded:
  results:
    0:
      type:           "artist"
      title:          "Pablo Picasso"
      description:    null
      og_type:        "artist"
      _links:
        self:
          href:       "https://api.artsy.net/api/artists/4d8b928b4eb68a1b2c0001f2"
        permalink:
          href:       "https://www.artsy.net/artist/pablo-picasso"
        thumbnail:
          href:       "https://d32dm0rphc51dk.cloudfront.net/i3rCA3IaKE-cLBnc-U5swQ/square.jpg"
    1:
      type:           "artist"
      title:          "Paloma Picasso"
      description:    null
      og_type:        "artist"
      _links:
        self:
          href:       "https://api.artsy.net/api/artists/52b32400275b24533a00019c"
        permalink:
          href:       "https://www.artsy.net/artist/paloma-picasso"
        thumbnail:
          href:       "https://d32dm0rphc51dk.cloudfront.net/88_9TcM9tcyhE0TPmKOHEA/square.jpg"
```


The search results are listed in [“_embedded”][“results”] field. Search endpoint returns search results across 5 types: artist, artwork, profile, gene and show. In this homework, we are only interested in artist results. After making a request to the Search API, you will filter results by choosing the ones with “og_type” equal to “artist”. The ID of artists can be extracted from the ["_links"]["self"]["href"] field of each result. The part after the last slash symbol is the ID of the artist. (e.g. if the artist link is https://api.artsy.net/api/artists/4d8b928b4eb68a1b2c0001f2, artist ID is 4d8b928b4eb68a1b2c0001f2)

Artsy search results are paginated, meaning that it returns a specific number of results after each request (set by the size parameter) and if the remaining results are wanted, another request can be made to the link in the [“_links”][“next”][“href”] field of the response. In this homework, you will only use results from the first page and will not make additional requests.

We are interested in [“title”] (name of the artist) and ["_links"]["thumbnail"]["href"] (image for the artist) fields as well as the IDs of artists in this homework. When an artist does not have an image, ["_links"]["thumbnail"]["href"] field contains “/assets/shared/missing_image.png”.

4-2-3 Artists Endpoint (Same as Homework 6)

Artists endpoint is used to get details about an artist. It has the base URL https://api.artsy.net/api/artists. You send your request using HTTP GET method. It accepts a query parameter **id**, denoting the ID of an artist. Differently from the Search API, you append the **id** parameter

to the link without specifying the name of the parameter. (i.e., You send request to https://api.artsy.net/api/artists/{id}) Similar to the search endpoint, you should set X-XAPP-Token header in your request to the token returned from Authentication endpoint.

Example request: https://api.artsy.net/api/artists/4d8b928b4eb68a1b2c0001f2

The server returns a JSON result as given below.

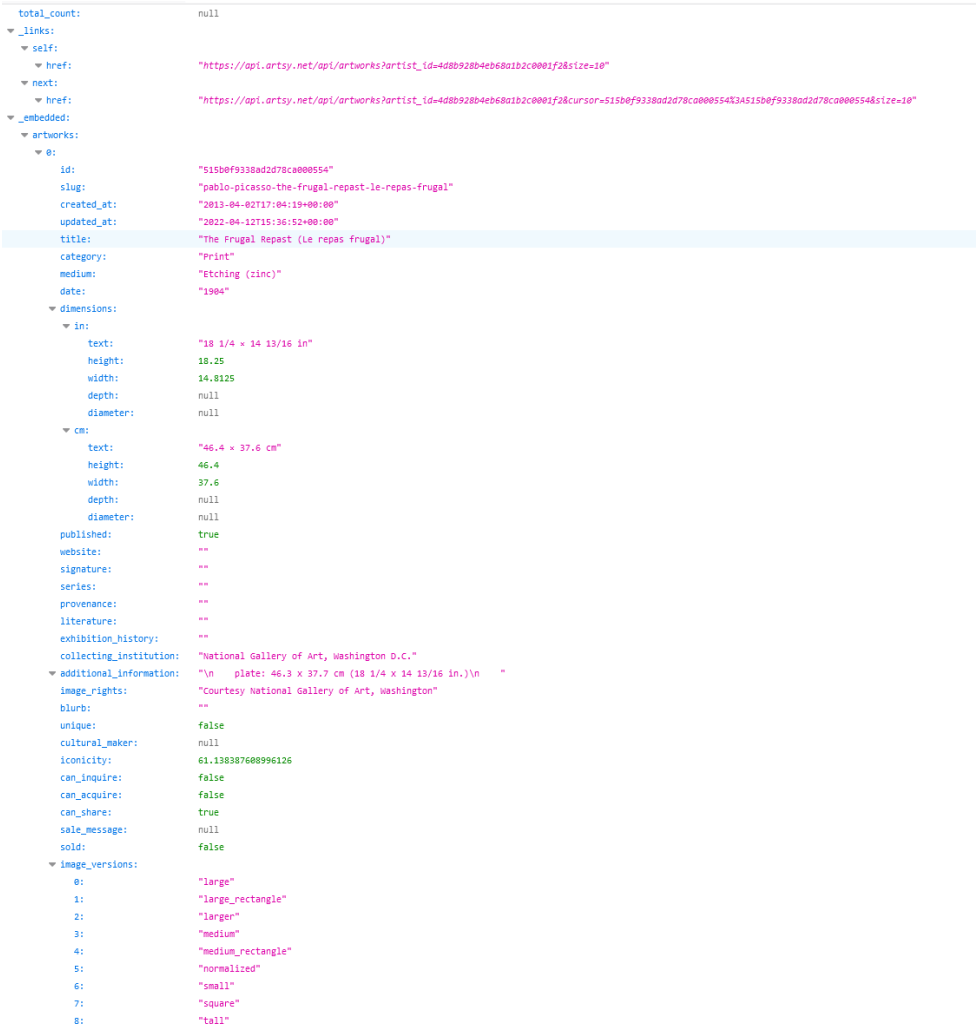
In this homework, we are interested in [“name”], [“birthday”], [“deathday”], [“nationality”] and [“biography”] fields of this response.

4-2-4 Artworks Endpoint

Artworks endpoint is used to get a list of artworks that belong to other entities (i.e., artists, partners, shows, collections, users). In this homework, we are interested in retrieving the artworks of a specific artist. Artworks endpoint has the base URL https://api.artsy.net/api/artworks. You send your request using HTTP GET method. It accepts a query parameter **artist_id** denoting the ID of the artist, artworks of which are being retrieved. Similar to the search endpoint, the maximum number of results can be set by **size** parameter, which has the maximum value of 10. In this homework, you will always set it to 10. Similar to the search and artists endpoints, you should set X-XAPP-Token header in your request to the token returned from Authentication endpoint.

Example request: https://api.artsy.net/api/artworks?artist_id=4d8b928b4eb68a1b2c0001f2&size=10

The server returns a JSON result excerpt from which is given below.



The artworks are listed in [“_embedded”][“artworks”] field. We are interested in [“id”] (ID of the artwork), [“title”] (name of the artwork), [“date”] (creation year of the artwork) and [“links”][“thumbnail”][“href”] (image of the artwork) fields of each artwork result.

Similar to search results, artworks results are paginated, meaning that the endpoint returns a specific number of results after each request (set by the size parameter) and if the remaining results are wanted, another request can be made to the link in the [“_links”][“next”][“href”] field of the response. In this homework, you will only use results from the first page and will not make additional requests.

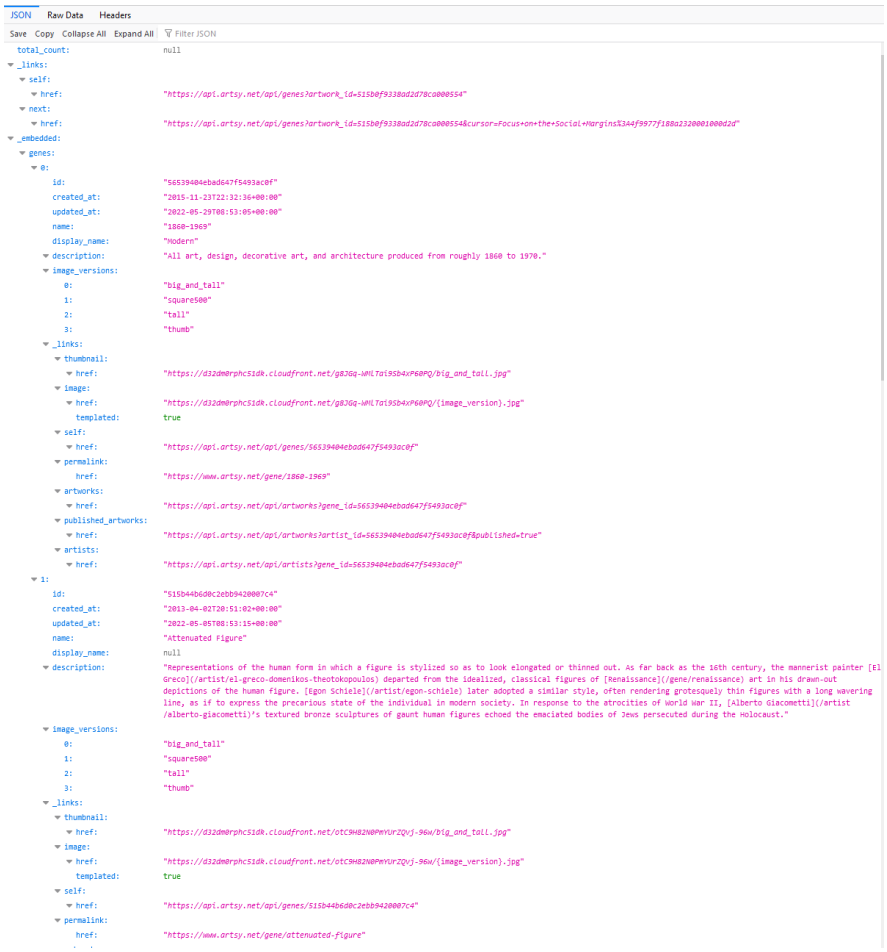
4-2-5 Genes (Categories) Endpoint

Genes endpoints is used to get metadata about artists or artworks in Artsy. You will use it to get metadata about artworks. Since gene is a strange word for this context, we use the word “category” instead. Genes endpoint has the base URL https://api.artsy.net/api/genes. You send your request using HTTP GET method. It accepts a query parameter **artwork_id** denoting the ID of the artwork for which

categories are being retrieved. Similar to the search, artists and artworks endpoints, you should set X-XAPP-Token header in your request to the token returned from Authentication endpoint.

Example request: https://api.artsy.net/api/genes?artwork_id=515b0f9338ad2d78ca000554

The server returns a JSON result except from which is given below.



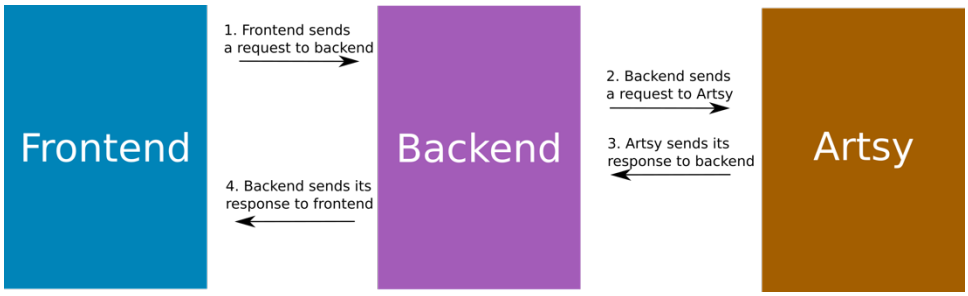
The categories are listed in [“_embedded”][“genes”] field. We are interested in [“name”] (name of the category) and [“_links”][“thumbnail”][“href”] (image of the category) fields of this response.

Similar to search and artworks results, gene results are paginated, meaning that the endpoint returns a specific number of results after each request (set by the size parameter) and if the remaining results are wanted, another request can be made to the link in the [“_links”][“next”][“href”] field of the response. In this homework, you will only use results from the first page and will not make additional requests.

4.3 System Overview (Similar to Homework 6)

The system contains three components: 1) browser (frontend), 2) Node.js application (backend) and 3) Artsy servers. You will implement the frontend and the backend. Your backend will have two functionalities: serving the frontend static files to the browser and responding frontend’s AJAX requests by fetching data from Artsy servers. You will not directly call Artsy API from the frontend as it requires

disclosing Client Id and Client Secret (and/or XAPP_TOKEN) to the public. The data flow diagram after an AJAX call is shown below.



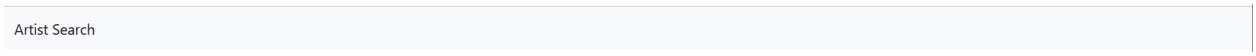
Sending a request to anywhere except your backend from the frontend will result in a 4-point penalty. Please do not directly call Artsy API endpoints from your frontend.

All requests from frontend to backend must be implemented using HTTP GET method, as you will not be able to send us sample backend endpoint links as a part of your submission if you use HTTP POST.

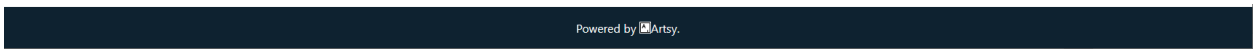
4-4 Page Layout

The page contains a header section, a content section and a footer section arranged vertically.

The header section is a static bar (Hint: Check Bootstrap navbar component) containing site title “Artist Search” on its left. The footer section is another static bar containing Artsy attribution “Powered by Artsy.”, in which the Artsy logo is placed before the word “Artsy”. When Artsy text on the footer is clicked, the page is redirected to Artsy homepage.



Page Header

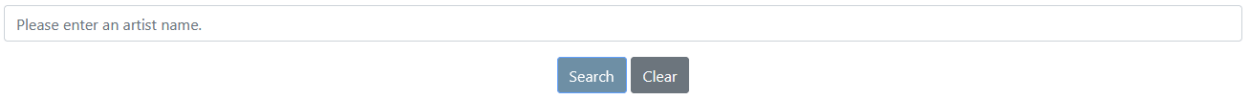


Page Footer

The content section contains the contents of the page, in which the following features will be shown.

4-5 Search Form

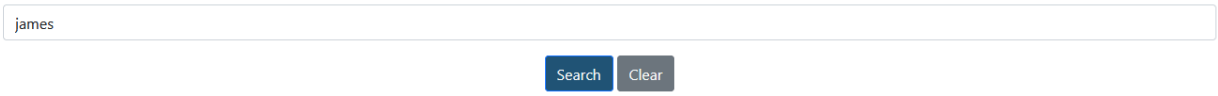
When a user opens the page for the first time, there is only a search form (Hint: Check Bootstrap Forms) in the content section. Search form contains three items: an input section, a search button and a clear button, shown below.



Search Form

The behavior of the Clear button is described in Section 4-8.

Search button is enabled if and only if the input section contains some input as shown below. (Hint: check **disabled** attribute).



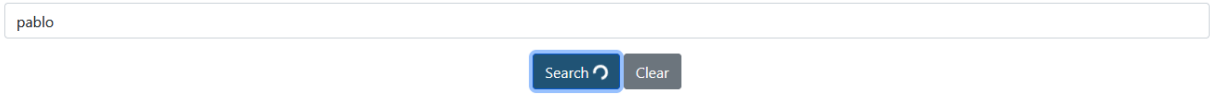
Search Button is Enabled when Input Section is not Empty

Artist names will be entered into the input section of the search form. The input section has the placeholder value “Please enter an artist name.”, which is shown when the input field is empty. When the input section is focused (i.e., input section is clicked and the cursor is blinking within the field), its style changes as shown below.

Input Section when Focused

User can initiate to search by either **1) clicking search button** or **2) by hitting enter key**. User cannot initiate search when the input section is empty because the search button is disabled.

Similar to Homework 6, when the search is initiated, your frontend will do an AJAX call to the backend, sending the input text. Upon receiving the request, your backend will make a request to Artsy Search API, forwarding the input text to Artsy Search API via parameter **q** as described in the Search Endpoint section above. Until a response is received from the backend, a spinner (Hint: Check Bootstrap spinners) will be shown near the search button as shown below.



Spinner is Shown Until a Response Arrives

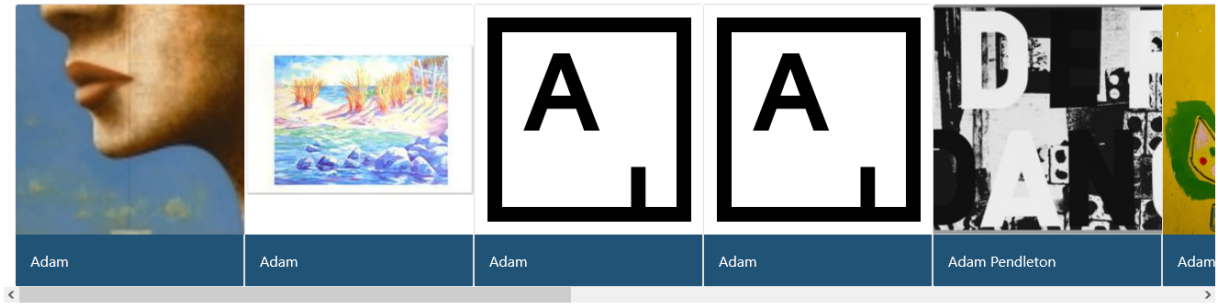
When the response arrives, spinner will be hidden, and the results are shown in the result list.

4-6 Result List

The results of the search are shown as a list of cards (Hint: Check Bootstrap Cards), similar to Homework 6. Each card contains the artist image with link retrieved from ["_links"]["thumbnail"]["href"] field of an artist result in the response of search endpoint. Below the image, the artist name is listed corresponding to the [“title”] field of the artist result. The list of result cards is scrollable. The artist names have a blue background color (HTML Color Code: #205375).

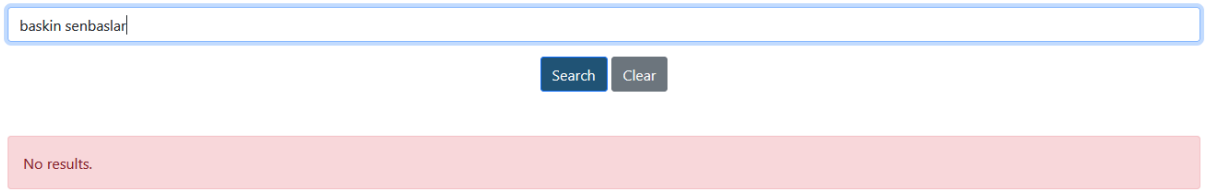
Result List

If a particular artist does not have an image, Artsy returns "/assets/shared/missing_image.png" in the ["_links"]["thumbnail"]["href"] field as mentioned in Section 4-2-2. In this case, you will place Artsy logo as the artist image as shown below.



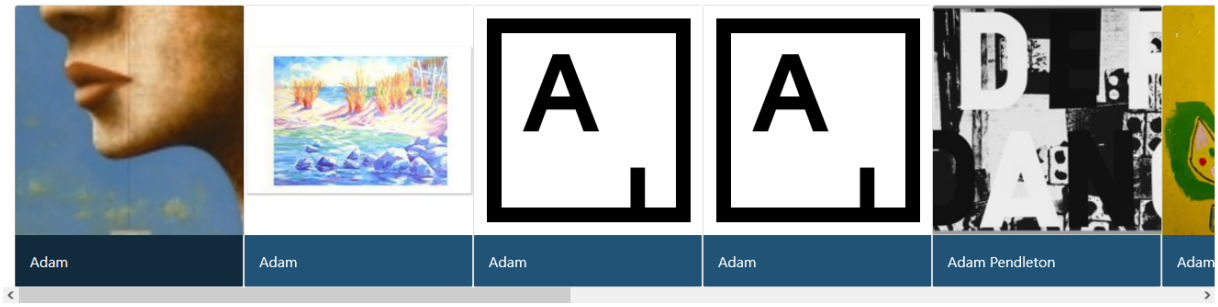
Artsy Logo is Shown if an Artist Does Not Have an Image

If the search results are empty, i.e. no artists match the given query, an error message containing text "No results." will be shown as below (Hint: Check Bootstrap Alerts).



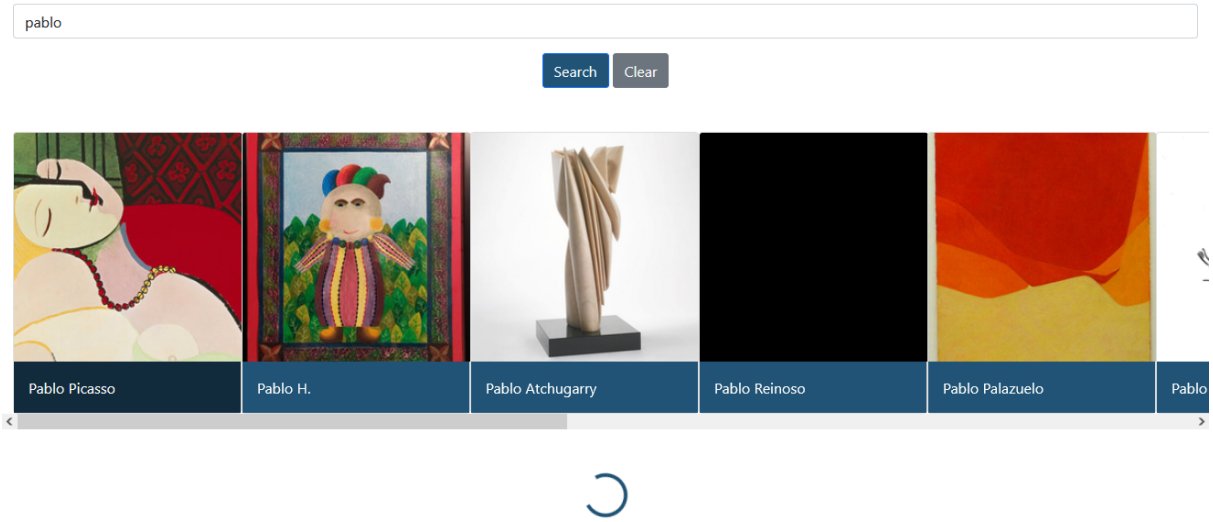
No Artists Match the Query

When a card is hovered, its background color changes to darker blue (HTML Color Code: #112B3C) as shown below.



The Background Color Change of a Hovered Card

When an artist card is clicked, your frontend will do an AJAX call to your backend to get artist details. In order to get artist details, you must send artist ID to the backend. For that, you have to associate artist IDs with cards, which you can do during initial search by sending artist IDs from your backend to the frontend as a part of the response. Until a response is received from the backend, a spinner is shown below the result list as shown below.



Spinner is Shown Until a Response Arrives

The selected card's background color is permanently set to darker blue after the click, which denotes the active card for which details are being shown. It will remain darker blue until another card is selected, at which point the background color of the newly selected card will be set to darker blue and the original card's background will set to original blue.

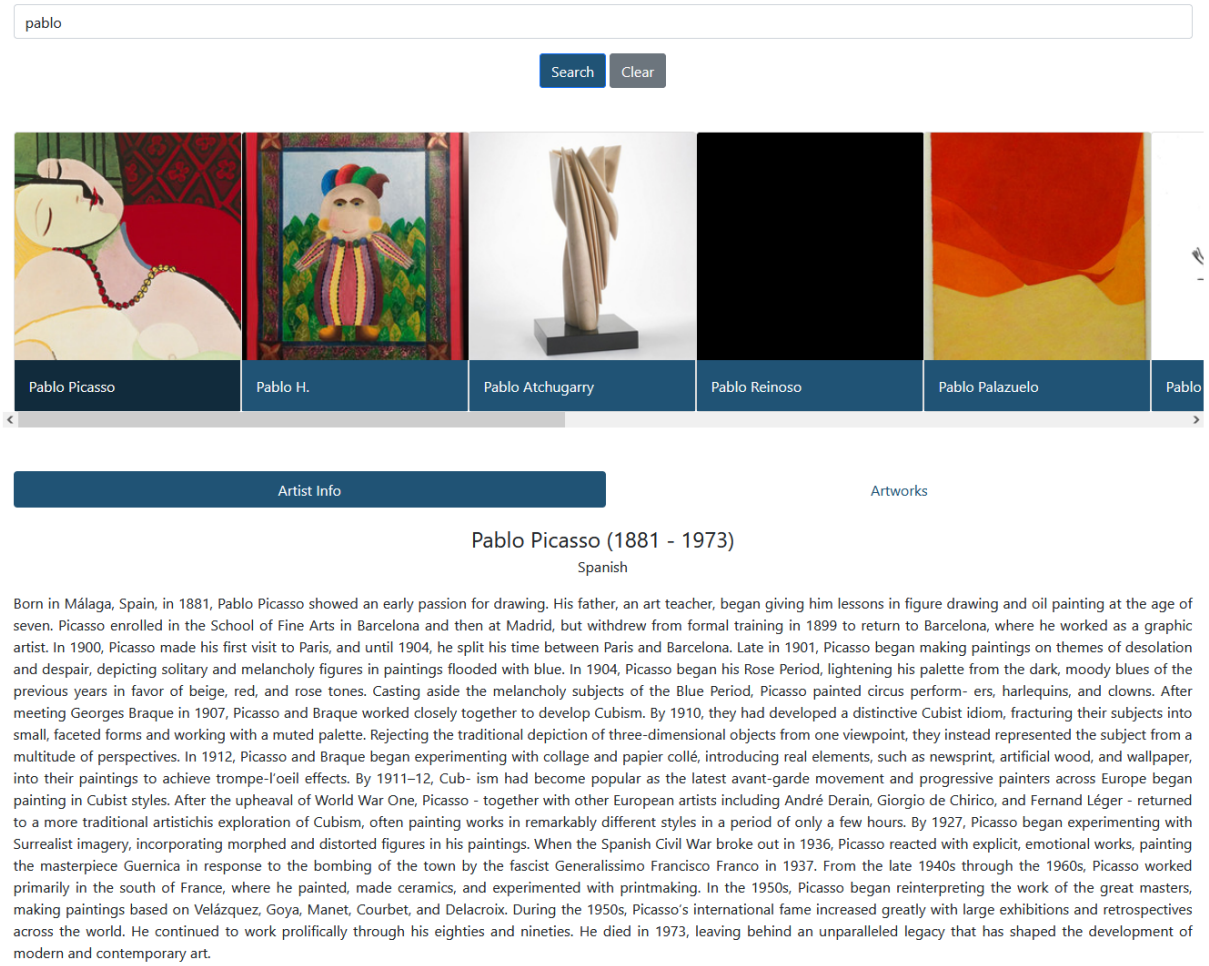
When the response arrives for artist details, you will hide the spinner and show the artist details.

4-7 Artist Details

Artist details are shown in two tabs: Artist Info and Artworks (Hint: Check Bootstrap Navs and Tabs).

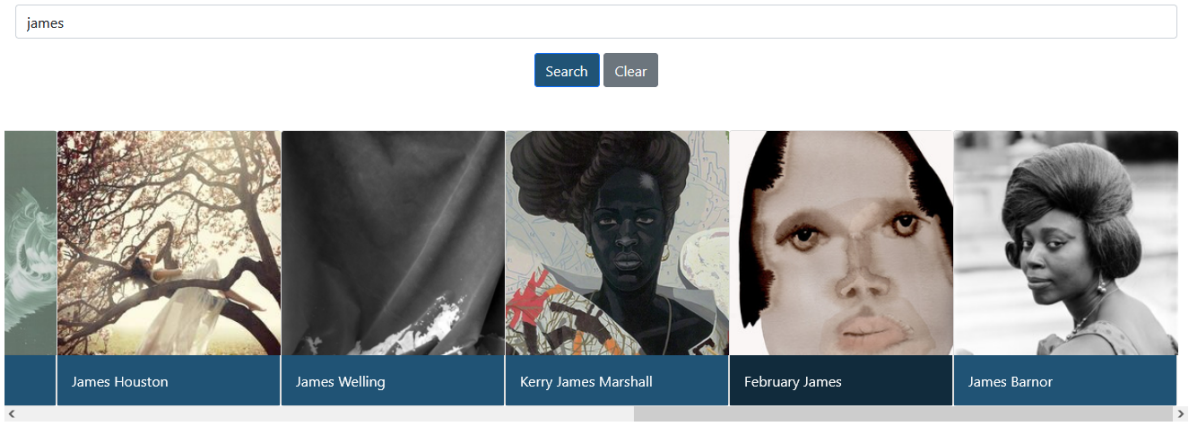
4-7-1 Artist Info Tab

Artist Info tab contains the same information as Homework 6. It includes artist name, artist birth year, artist death year, artist nationality and artist biography retrieved from Artists endpoint as shown below.

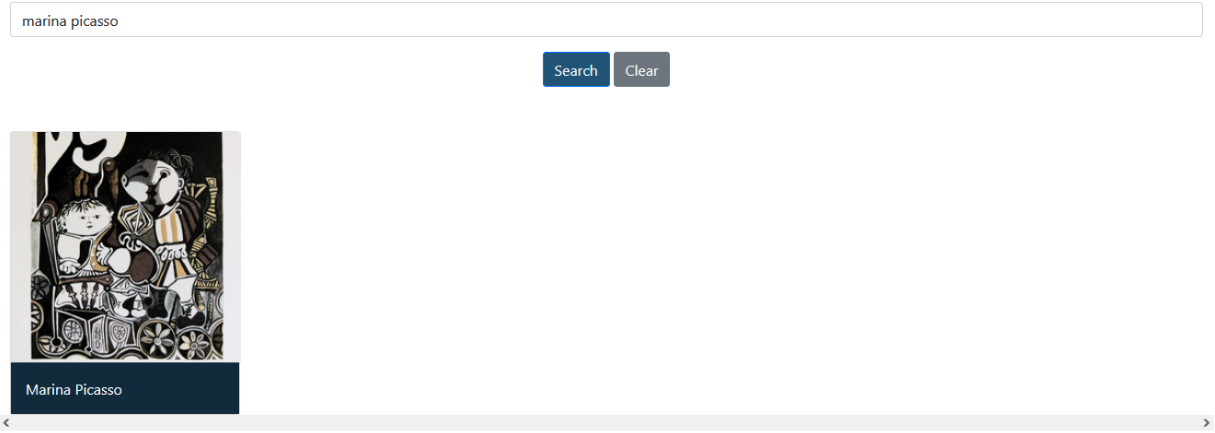


Artist Info Tab

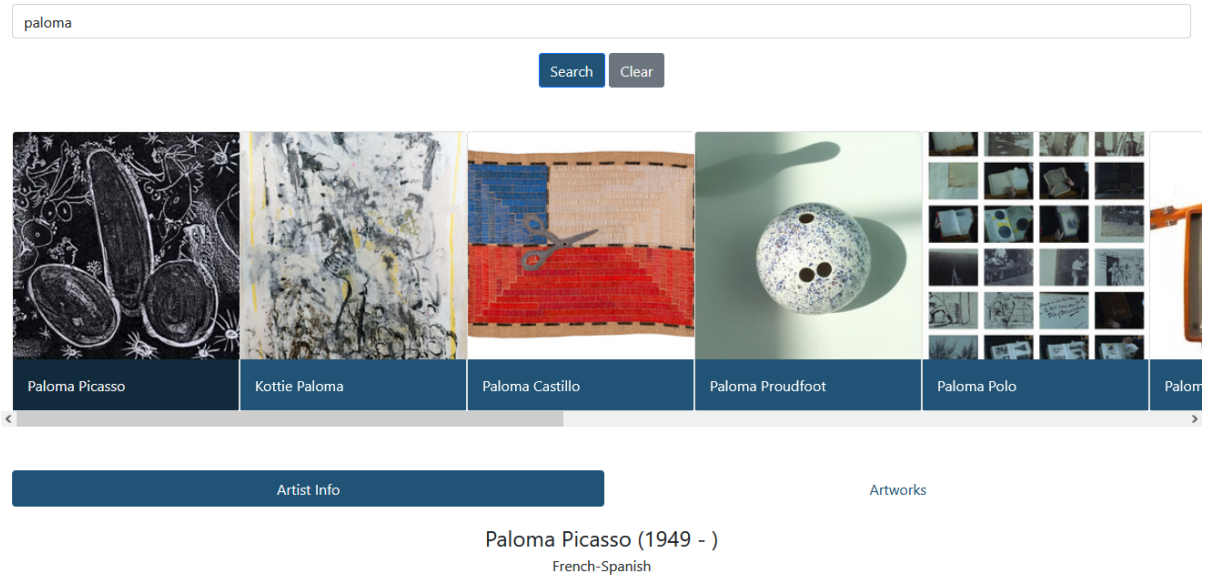
First line contains the name of the artist followed by birth and death days. Next line contains the nationality of the artist. Next, the biography is shown. This information corresponds to [“name”], [“birthday”], [“deathday”], [“nationality”] and [“biography”] fields of the artists endpoint response. Similar to Homework 6, if any of the fields are empty, you should simply leave them blank. Examples are given below.



No Biography, No Birth Year, No Death Year



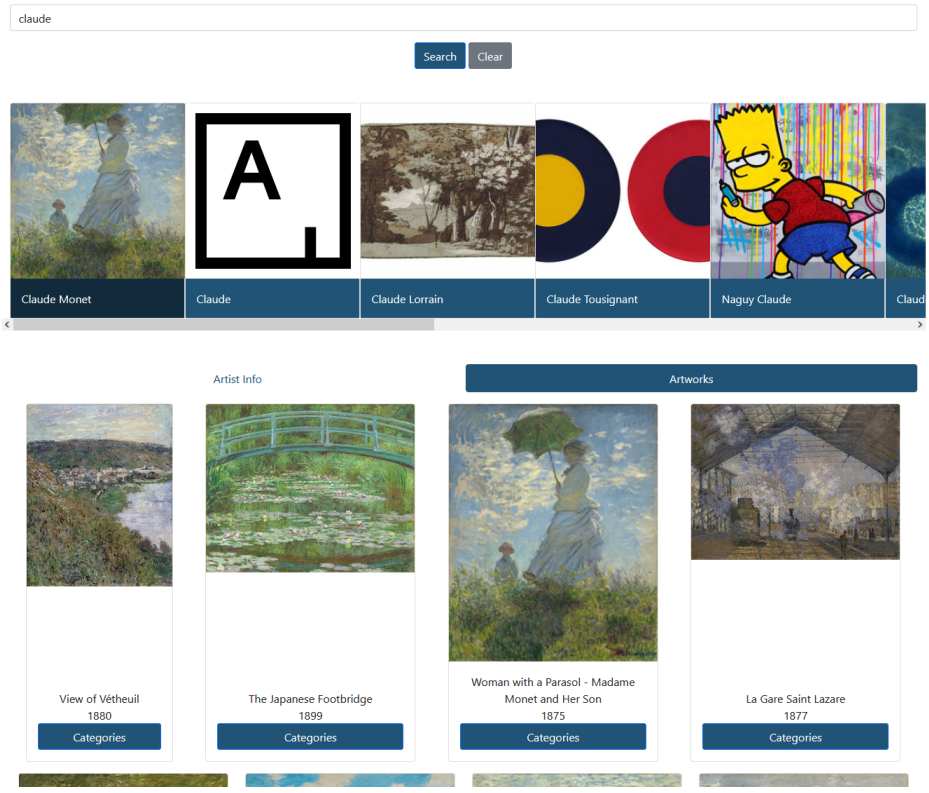
No Nationality, No Death Year, No Biography



No Death Year, No Biography

4-7-2 Artworks Tab

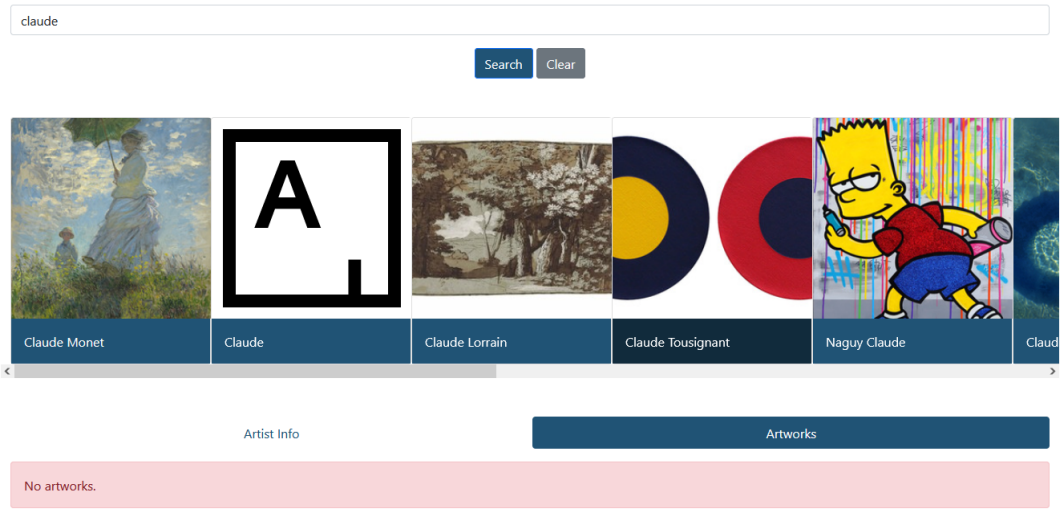
Artworks tab contains the artworks of the selected artist. Artworks are retrieved from the Artworks endpoint described in Section 4-2-4. Example is shown below.



First Four Artworks of Claude Monet (Page Contains More)

Each artwork has an image corresponding to the [“links”][“thumbnail”][“href”] field of the artwork (See Section 4-2-4), a name corresponding to the [“title”] field of the artwork (See Section 4-2-4) and a creation year corresponding to the [“date”] field of the artwork (See Section 4-2-4). Artworks are listed using Bootstrap Cards.

If an artist does not have any Artworks in Artsy, you should show an error message containing text “No artworks.” as shown below.

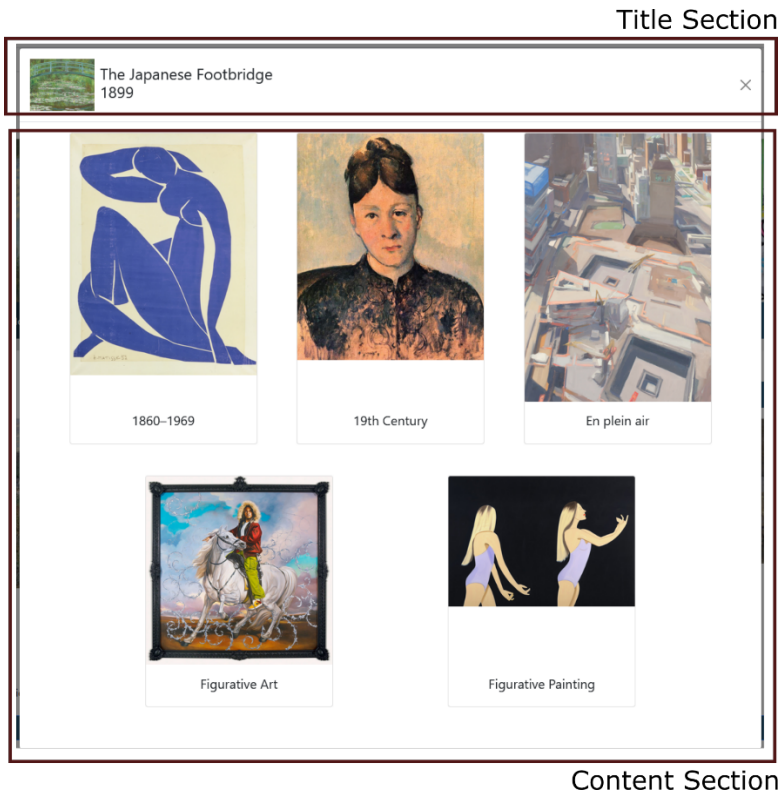


Artist does not have any Artworks

A “Categories” button below each artwork opens a modal (Hint: See Bootstrap Modals) in which categories for the artwork retrieved from the Genes Endpoint (See Section 4-2-5) are shown.

4-7-3 Categories Modal

The categories modal has a title and a content section separated by a line as shown below.



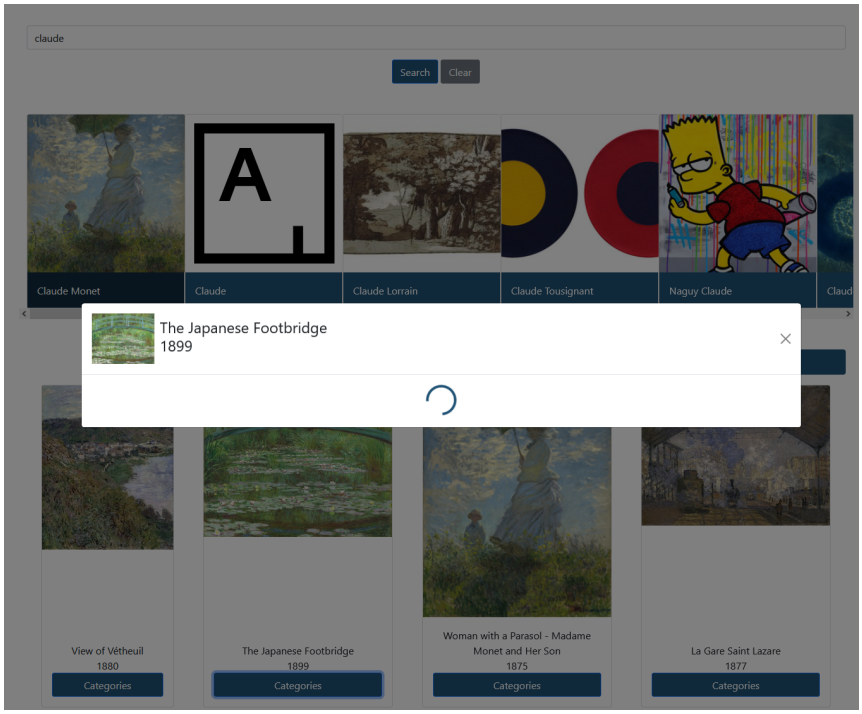
Categories Modal Sections

The title section contains the image of the artwork, name of the artwork and creation date of the artwork. Content section contains the categories of the artwork retrieved from the Genes Endpoint.

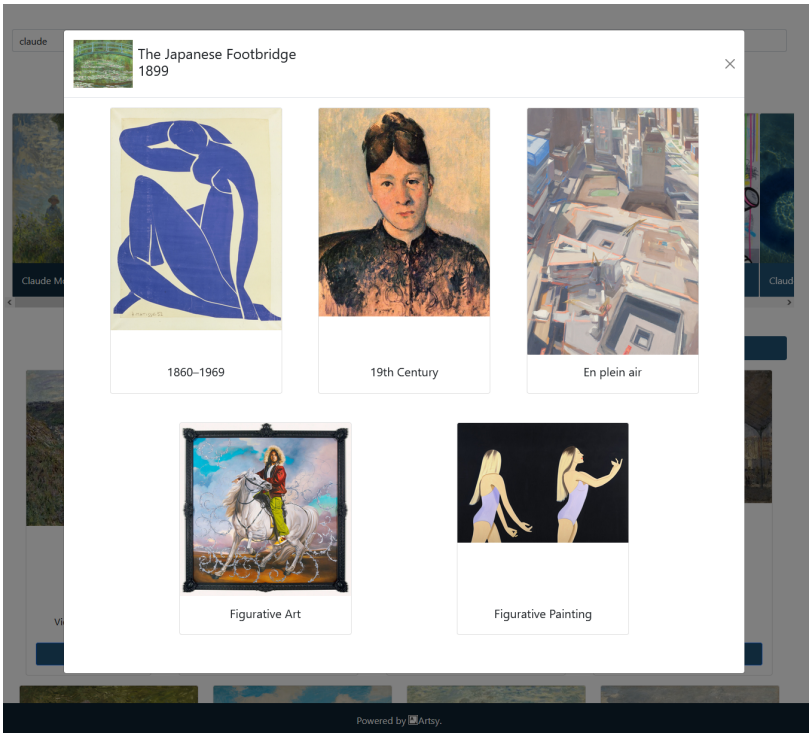
Each category has a name corresponding to the [“name”] field of the gene (See Section 4-2-5) and an image corresponding to the [“_links”][“thumbnail”][“href”] field of the gene (See Section 4-2-5).

When the “Categories” button of an artwork is clicked, the modal is opened, and a request is sent to the backend to retrieve the categories (or genes) of an artwork. In order to get categories, you must send artwork ID to the backend. You can associate artwork IDs with “Categories” buttons using the IDs provided by the Artworks endpoint.

Until a response arrives, a spinner is shown in the content section of the modal as shown below. Spinner is replaced by the categories when the response arrives. Each category is shown using a Bootstrap Card.

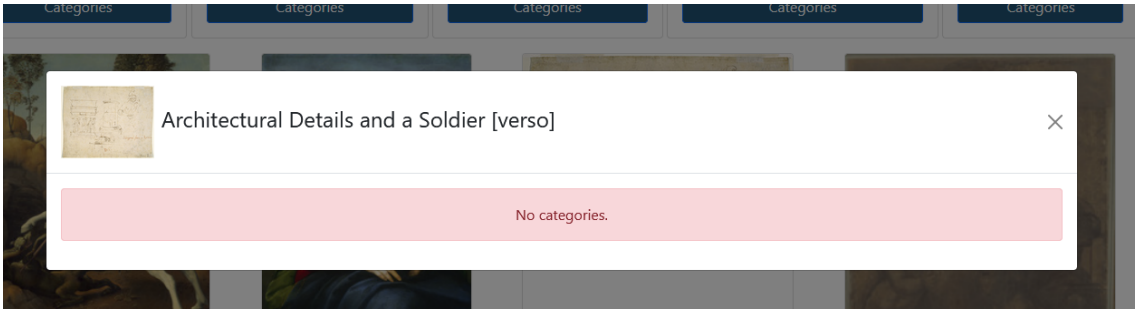


Spinner is Shown until Categories Response Arrives



Spinner is Replaced with Categories when Response Arrives

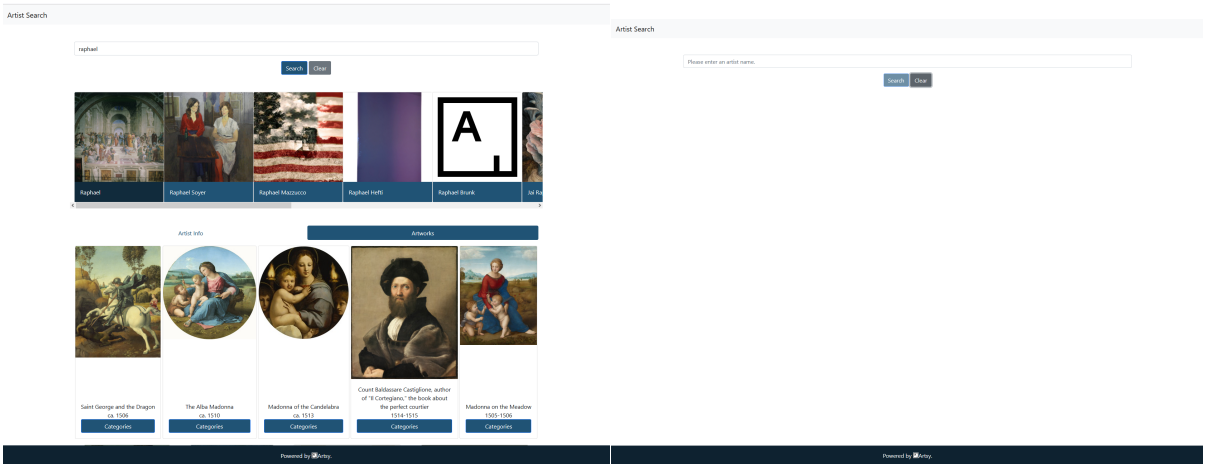
If a particular artwork does not have any categories, an error message containing the text “No categories.” Is shown in the content section of the modal as shown below.



Artwork does not have any Categories

4-8 Clear Button of the Search Form

The clear button of the search form brings to page to its initial state. It clears the input section, result list and artist details as shown below.



Before and After Clicking the Clear Button

4-9 Different Use Cases

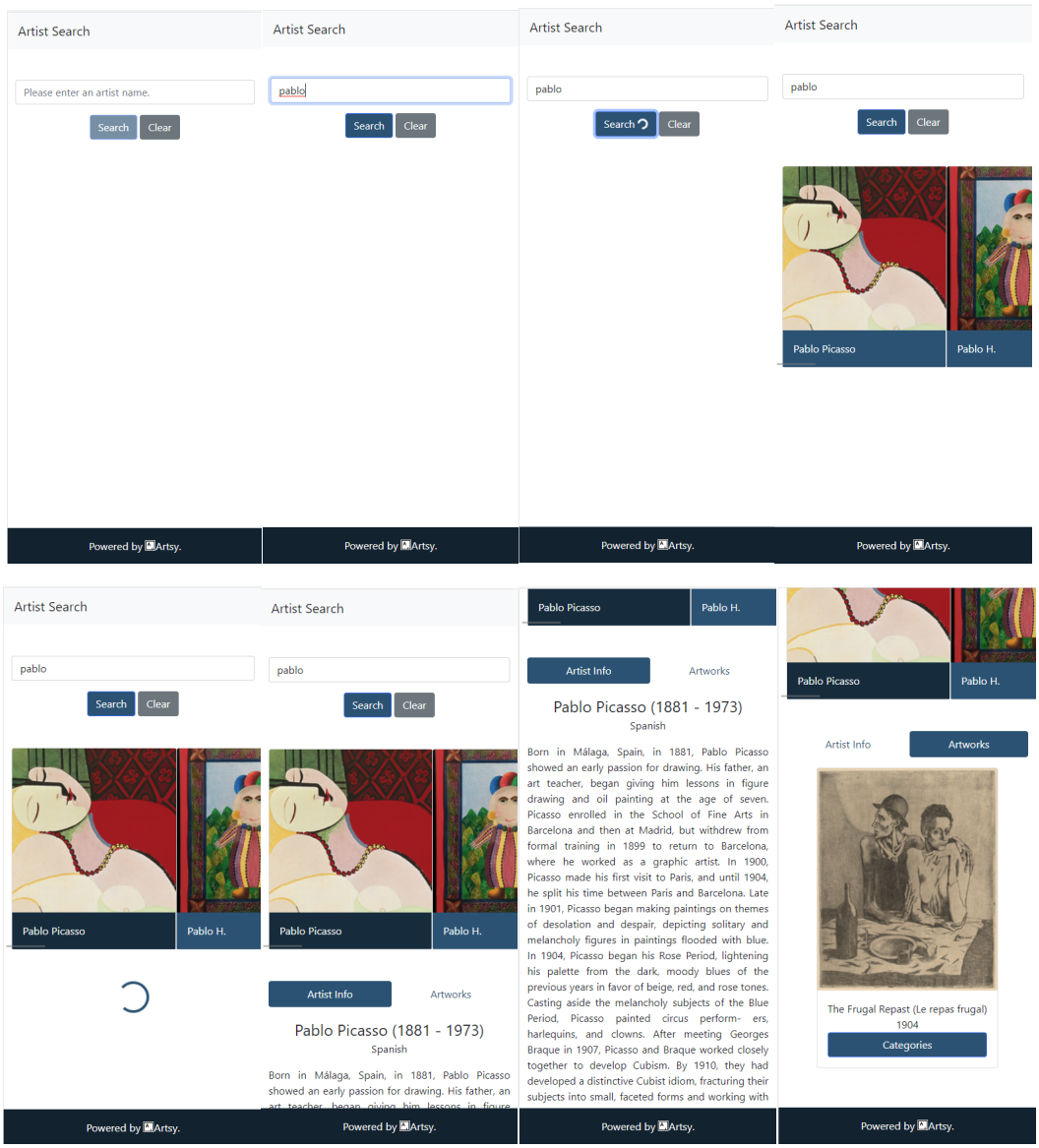
When a user clicks on another search result while the details of an artist are being shown, the website will hide the artist details and show the spinner below result list until a response arrives. The background color of the newly selected card will be set to darker blue, and the background color of the previously selected card will be set to original blue. When the response arrives, the details of the newly selected artist will be shown, and the spinner will be hidden.

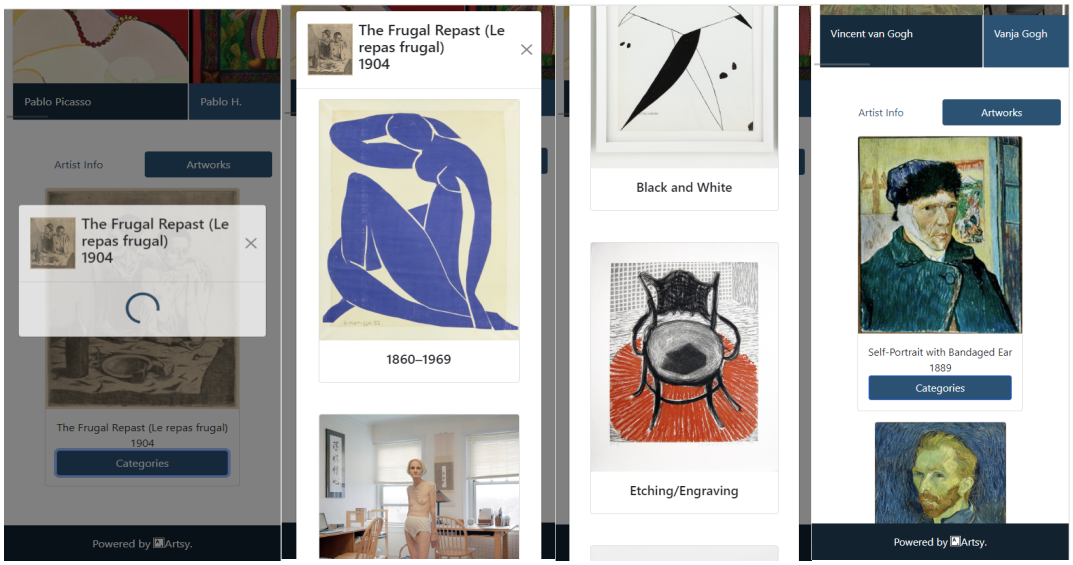
When a user searches a new artist using the search form while the details of an artist are shown, the details and the result list will not be changed, and the spinner of the search button will be shown until a response arrives. Result list will be replaced when the response arrives, and the artist details will be cleared. Spinner will be hidden, and all background colors of results will be set to original blue.

When a user searches a new artist using the search bar while search results are being shown without any details, the spinner of the search button will be made visible until a response arrives. Search results will not be hidden, but they will be replaced whenever results arrive. When the response arrives, spinner will be hidden.

4-10 Responsive Design

The webpage you develop must be responsive. One easy way to test responsive behavior is to use Google Chrome Responsive Design Mode in Developer Console. Here are some snapshots for Google Pixel 5 phone using the Google Chrome Responsive Design Mode. All functions should work on mobile devices.





5 Homework Submission

In your course homework page (on GitHub Pages), update the Homework 8 link to refer to your deployed website. Also, provide an additional link to one of your cloud query entry points, below the homepage link. Your project must be hosted on GCP. Graders will verify that these links are indeed pointing to GCP.

Also, submit your source code file to DEN D2L as a **ZIP archive**. Include your frontend and backend source code, plus any additional files needed to build your app (e.g., yaml file). The timestamp of your last submission will be used to verify if you used any grace days.

6 Notes

- You can use jQuery for Homework 8, but it is not needed.
- You must use Bootstrap for Homework 8. **If Bootstrap is not used, a 4-point reduction will be applied.**
- Appearance of the webpage should be like the reference videos as much as possible.
- Artsy lacks information for a lot of artists. Here are some examples for testing:
    - Pablo Picasso: Has every detail in Artist Info, has a single artwork.
    - Vincent van Gogh: Has everything except biography in Artist Info, has 10 artworks.
    - Yayoi Kusama: Has everything except death date (Since she is not dead) in Artist Info, has no artworks.
    - Claude Monet: Has every detail in Artist Info, has 10 artworks.
    - Raphael: Has everything except biography in Artist Info, has 10 artworks, some of which does not have categories.

7 Hints

- You should use the domain name of the GCP service you created in Homework 7 to make requests. For example, if your GCP server domain is example.appspot.com,

https://example.appspot.com will become your base URL. The example subdomain in the above URLs will be replaced by your choice of subdomain from the cloud service.

8 Static Resources

- You can find the Artsy logo required for this homework in https://csci571.com/hw/hw8/images/images.zip.