# Art Map

By: Connor Aguilera
Kanakapriya Krishnakumar
CSC 690

# Project Information:

**Project Name:** Art Map
**Website:** https://kkpriya.github.io/ArtMap/

# Project Report:

*Art Map is a user-friendly web application that allows the user to search for art murals, galleries, museums and other artistic locations in a desired city.*
This application serves to display information to the user in a bold and efficient way. The default location is Anaheim where it searches using Google Maps API to find Art Galleries and Museums. When found, markers populate the map that the user is able to click on to see the name of the location. Scrolling down the page, you see a list of five locations that the user can scroll through and observe. This section shows the name of the location, its location, the rating out of five stars as well as an “Add to Favorites” button that appends a list at the top of the search results with the name of the saved location.

# User Stories:

**1. User:**
Connor is visiting Seattle for the first time and is staying at a hotel. He doesn’t want to go to all the regular tourist traps. In most cities, it is hard to navigate and find cool and unique artsy locations. He found out about an application called *Art Map* that led him to art spots, murals and museums in the city. He started off by entering his hotel address into the search bar. He was taken to a page which gave a map and a list within a **8km** radius from his hotel with pins that show him unique art locations throughout the city. He was interested in going to this spot called Moma and was able to add it to a section called Favorites.

## List of Functional Specs:

1. **User** have the option to search an address.
2. **User** is provided with a list of art spots, murals, and museums.
3. **User** is provided with a map of pointers pointing to different spots in a specific city.
5. **User** is provided with an icon to zoom in/ out of the map to change search results.
6. **User** is given a list of top 5 art spots with address and rating.
7. **User** is able to save the spot in favorites.
8. **User** is provided with an about page which gives them more information about the website

## Categories Application is in:

Content-based retrieval, multimedia education systems, immersive environments

## Environment and Languages Used:

Atom text editor version 1.23.1 was used for development. HTML, Javascript and CSS were used.

## Multimedia Operations:

Author, annotate (Added by users using the “upload image” section), render, browse, search and retrieve, interact

## Data Types Inputted:

Images, maps, sounds

## Data Types Output:

Images and information

## API's Used:

Google Maps API
- Allows for the map view
- Allows for markers to be displayed on map with the place name

Google Places Autocomplete API
- For the search autocomplete

Google Places Details API
- To retrieve place information like address, rating, name

Google Geocoding API
- To use the location's latitude and longitude to then request the place's details through the Places Details API

## User Modes:

**Map Mode:** Users can manipulate the embedded Google Map, zoom in and out, explore markers and visualize the location of the locations.

**List Mode:** Scrolling down the page shows a list view of the top 5 spots shown on the map, allowing the user to get more information when wanted.

## Implementation:

This application was built using version 1.23.1 of the Atom IDE.

**The index page** of the site is just a landing page. It is used to show the name of the project, with a brief description inside the button that leads you to the application.

**The Art Map page** consists of a full screen embedded Google Map whose center starts in Anaheim, California. The HTML is used to set the divs and text in each section. The styles.css is used for the front end development, while the main.js does the backend/ API work.

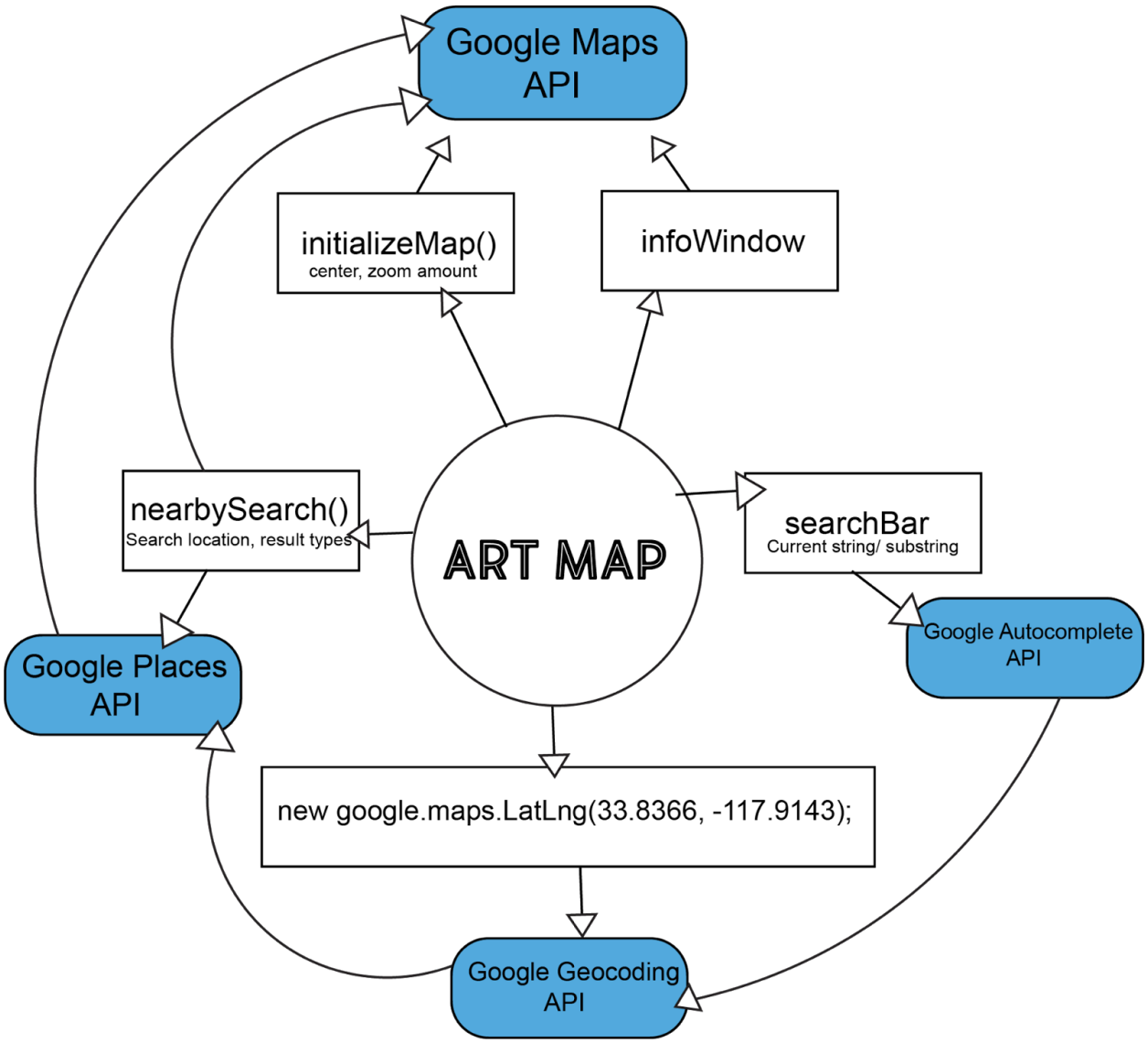
The **first section** of the main.js file is used to make the behavior of the list view section. A parallax effect is used for the background images while the portion sizes of each list is set as well.

The **second section** of the main.js creates the Google Map and Search Autocomplete. initializeMap() and initializeAutoComplete() are called to do said jobs. initializeMap() creates a new map object with the center being Anaheim and a zoom of 13. We have limited the range of searches to 8000 meters or 8 kilometers in order to not overwhelm the user with locations out of their way of travel. InfoWindows are initialized in order to show the place name above the clicked markers. When given the location, a new google.maps.places.PlacesService(map); request is made using the initialized map as the parameter. Using that service we make a nearbySearch looking for two types of locations: art_gallery, museum.

When a nearbySearch is made, the callback function is called passing the results json object array given by Google as the parameter. At this point, we are able to access most information about the location. (Google does not allow access to things like the locations website with just a nearbySearch query. That requires the place_id and a separate API call.

**The callback function** is where we take the place information array and populate our HTML elements with the correct information. The place's name populates the headers, the locations populate the info below it and the location's average rating out of 5 stars is given below that. A button is used to add the specific location to a "favorites queue" where the name of the location is appended to a list at the top of the page and stored during your current session. This remains through different location searches, creating a diary of locations the user can access.

**clearMarkers()** is called whenever a new search is made, clearing the markers on the map the same way the results[] list is cleared by Google. **createMarkers()** is called after clearMarkers() in order to populate the new location with the names of locations returned in the json object.


Google Maps API
initializeMap()
center, zoom amount
infoWindow
nearbySearch()
Search location, result types
ART MAP
searchBar
Current string/ substring
Google Places API
Google Autocomplete API
new google.maps.LatLng(33.8366, -117.9143);
Google Geocoding API

# Work Accomplished:

- Create full website
- Get all API's to work
- Retrieve Google's Autocomplete suggestions
- Convert the autocomplete suggestion into a usable geocoding latitude and longitude
- Using the latitude and longitude to populate a search for the area
- Display a nearby search of art galleries and museums around the city the user searches
- Display each location returned on the map with an infowindow with their name above it
- Create a list view of the places with more information below the map
- A functional Favorites list that stores the name of the desired location

# Work We Did Not Accomplish:

- Keeping the favorites list stored across sessions of the site (refresh clears the favorites list, however a different search does not)
- User annotation of place information
- Finding searches made on specific days
- Current location