IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

Image Search App using API

Prof. Bharat Parihar[1], Yashodhan Agale[2], Ashlesh Shinde[3], Vedant Patil[4], Aishwarya Kardak[5]
Professor, Department of Computer Engineering[1]
Students, Department of Computer Engineering[2,3,4,5]
Chhatrapati Shivaji Maharaj Institute of Technology, Panvel, Maharashtra, India

Abstract: *In this JavaScript project, we'll create a simple image search application from scratch. We'll use HTML, CSS, and JavaScript to build a user-friendly interface. The app allows users to search for images based on keywords, and it displays a set of initial images. Users can click on a "Show More" button to load additional images related to their search term. We'll guide you step-by-step through the development process, covering everything from creating the HTML structure to implementing the JavaScript functionality. Let's begin by creating the necessary files: index.html, style.css, and script.js*

Keywords: Image Search, JavaScript, Content-Based Retrieval, User Experience, Responsive Web Design, User Preferences, Security, Performance Optimization

I. INTRODUCTION

1) HTML:

To create the basic structure, open the index.html file and add an HTML boilerplate. Link the style.css file using a <link> tag and include the script.js file using a <script> tag. Next, create a heading using the <h1> tag, followed by a form element for the search input and button. Inside the form, add an input field with the type set to "text" and an ID of "search-input". Provide a placeholder text for the input field, such as "Search for images". Then, create a button with an ID of "search-button" and a label of "Search". After the form, create a
 element with a class name of "search-results". This will be used to display the fetched images. Save the changes and open the file in a browser using a live server to see the current progress. You will observe the heading, search input, and button on the page. Now, let's move on to displaying the initial three images. Create three templates similar to the existing examples on the website. Each template should be wrapped inside a element with an appropriate class name. Save the changes and refresh the browser. You will see the heading, search input, button, and the initial three images displayed on the page. We have set up the basic structure of our image search application. In the next steps, Now, we need to style our project.

2) CSS:

To style our image search application, open the style.css file and let's begin. First, let's define some basic styles. Set the body element's margin and padding to 0 to ensure a clean starting point. Set a background color and font styles for the entire page. Next, focus on the header. Style the <h1> element to make it visually appealing and align it at the center of the page. You can adjust the font size, color, and margin as per your design preferences. Moving on to the form section, apply appropriate styling to the search input field and button.

Set their dimensions, font styles, and alignment. Consider using padding and margin to create adequate spacing between elements.

For the search results container, define a specific height, width, and border to create a visual distinction. You can also apply some padding to give the images some breathing space. To style the individual images, apply appropriate dimensions, borders, and spacing. You can use flexbox or grid layout to arrange the images in a responsive grid-like structure. Consider adding some hover effects or transitions to create a more interactive experience when users interact with the images or buttons. Remember to experiment with colors, typography, and layout to match your desired design aesthetic. You can leverage CSS properties like background-image, background- color, border-radius, box-shadow, and many more to enhance the visual appeal of your image search application. As you work on styling, don't forget to save your changes and refresh the browser to see the updated styles. Remember that CSS offers a wide range of possibilities,

ISSN 2581-9429 IJARSCT

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

so feel free to explore and experiment to make your image search application visually engaging and user-friendly. In the next steps, we will implement the functionality to fetch and display images based on user input using JavaScript.

3) JavaScript:

First, we need to select the necessary elements from the DOM. Create variables to store references to the search input field, search button, and search results container using the querySelector() method. Next, add an event listener to the search button. When the button is clicked, we'll retrieve the user's input from the search input field. Store the input value in a variable. Inside the event listener, create a function that will handle fetching and displaying the images based on the user's search term. We can name this function fetchImages. Inside the fetchImages function, make an API request using the search term as a parameter. You can use methods like fetch() or XMLHttpRequest to retrieve the data. Once the API response is received, parse the data and extract the relevant image URLs and descriptions. You can store this information in an array or object. Next, we need to dynamically create HTML elements to display the fetched images. Loop through the image data and create the necessary elements, such as <img> tags and <p> tags for descriptions. Append these elements to the search results container you selected earlier. This will display the fetched images on the page.

Finally, test your application by entering a search term in the input field and clicking the search button. Verify that the images related to the search term are displayed in the search results container.

Remember to save your changes and refresh the browser to see the updated functionality search filters.

II. METHODOLOGY

In this point we will explain how we used JavaScript to build an image search app. It fetches and displays images from Unsplash based on the user's search query. This is one of several personal projects We have built to enable me to gain some practice on all we have been learning. We first globally stored references to the form and some other HTML elements that we were going to work with. we would still select others within functions later. we also added a submit event listener to the form. function takes an event as its argument and first of all prevents the page from reloading using the preventDefault() method. It then stores the value of the search input in input Value and removes any whitespace with the trim() method. It stores the trimmed input value in searchQuery and passes it as an argument to the fetch Results function which is being called there. We logged the value of searchQuery to the console to make sure the right value was being passed. To be able to use Unsplash's API, you have to create a developer account. Only after that do you get your unique API key with which you can access the photos on the site. An AJAX request is made to Unsplash using a URL containing the endpoint and the relevant query parameters. More information on this is provided in the Documentation page on their website.

The function searchUnsplash takes one parameter (searchQuery), which is inserted into the endpoint as the value of the query query parameter. Now, the URL is stored in a variable endpoint which is passed as a parameter to fetch.

The fetch() method takes one argument, the path to the resource you want to fetch, which is stored in endpoint in this case. It always returns a Promise. Now, if the response is 200 OK, it is parsed as JSON which is stored in the json variable. The result is logged to the console so as to view the contents of the JSON data. Both functions above are asynchronous which means that the await keyword can be used to pause the execution of the function until a promise is resolved. This is achieved by placing the async keyword before declaring a function. I used a try...catch block in the fetchResults function. The try block 'tries' to execute the code within it and, should there be an exception or error, the catch block saves the day and responds appropriately with whatever code is written within it. This is a control flow mechanism which prevents the code from crashing if an error occurs while fetching the results. The next thing is to display the results on the page. If you check the JSON data looged to the console, you will find that it contains several properties which have different contents. The results property is an array in which the search results are contained. Each search result is an object and can be accessed using either dot or bracket notation. An empty div with a class of search results was already created in the HTML file. It is then selected in the JS file within a new function called display Results which takes a JSON object as an argument.

The textContent is also set to an empty string to clear all previous results. Now, the results array is iterated over using the forEach method. Within the callback function, the nested object can be accessed through the result parameter. If you

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

study the above image closely, you will find that each object in the array contains several keys such as links, user, urls. These can be used to access different categories of information on the object in question.

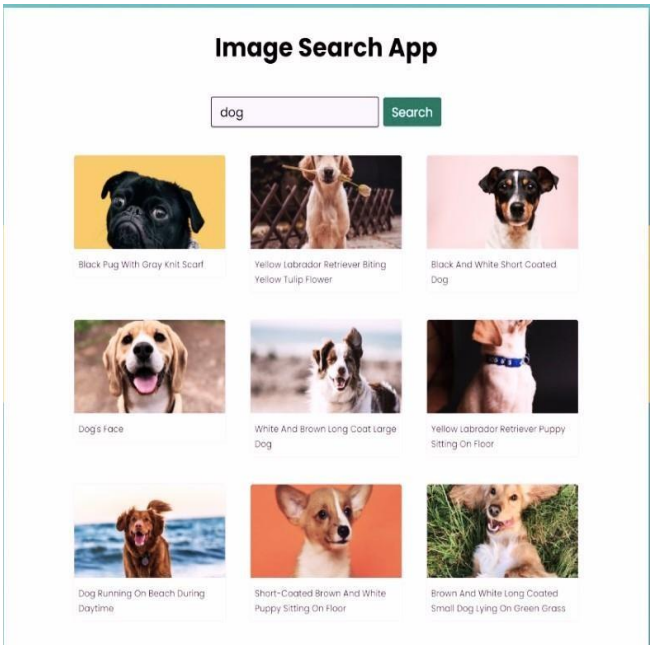
The first lines within the callback function are variables in which the different values needed are stored. They were all accessed using dot notation and include:

• the image url
• the link to the image on Unsplash
• the name of the photographer
• the link to the photographer's Unsplash profile

Afterwards, each result is inserted into the searchResults element using the insertAdjacentHTML method. This method takes two arguments: the position of the element, and the text to be parsed as HTML. Template literals are used in the second argument because of the parts of the code that will be changing constantly. These are represented by placeholders in the code. The function displayResults is then called in fetchResults.

Each image is set to be the background of its container, and is also a link to its Unsplash page. The name of the photographer, which links to his/her Unsplash profile, is placed right under the image and the result display was styled using CSS Grid.

III RESULTS



IV. DESIGN AND IMPLEMENTATION

Nowadays, the "Search by photo" approach is being used actively. "Keyword search" (with an understanding of the content of images). So for practise purpose we build this project since our main aim is to work in product based companies such as zomato, swiggy, amazon etc which uses API to fetch data an dshow user on screen acc ording to users choice

Step 1. Call an API : First of all from unsplash.com we have to call an API as all data required for our app is coming from their server. So we first call the API

Step 2. Fetch the API : After calling the API using JavaScript we fetch the data coming from server of unsplash

Step 3 . Show Images : In this step, We will Show images to user on screen as user prefers images to see and will provide a Show More button for user through which user can iterate or get infinite images related to his search

ISSN 2581-9429 IJARSCT

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

VI. OBJECTIVES

1. User-Friendly Interface :
• Create a user-friendly and responsive user interface for your image search app.
• Ensure that users can easily search for and browse images.
2. Image Search Functionality :
• Implement a search bar where users can enter keywords or queries to search for images.
• Integrate an image search engine or API (e.g., Google Custom Search, Unsplash API, or Shutterstock API) to fetch relevant images based on user queries.
3. Display Results:
• Display search results in an organized and visually appealing manner.
• Include features like infinite scrolling or pagination to browse through multiple pages of results.
4. Filter and Sorting Options:
• Provide users with options to filter and sort search results (e.g., by date, size, or relevance).
5. Image Preview :
• Enable users to view image previews by clicking on search results.
• Implement modal or lightbox features to view images in a larger format.
6. Image Details:
• Show relevant details about the images, such as title, source, and image dimensions.
7. User Accounts and Preferences :
• Allow users to create accounts, save their favorite images, and set preferences for search results.
8. Image Licensing Information :
• If applicable, display licensing information for images, especially if you're using APIs that require attribution.
9. Advanced Search Features:
• Implement advanced search features, such as searching by color, image type (e.g., photos, illustrations), or image source.
10. User Feedback and Rating:
• Enable users to provide feedback or rate images.
• Consider implementing a user review system for images.
11. Bookmarking and Collections:
• Allow users to create image collections or albums for better organization.
12. Integration with Social Media:
• Implement social media sharing options for users to share images on platforms like Facebook, Twitter, or Pinterest.
13. Image Upload and Management:
• Include the option for users to upload their own images.
• Provide tools for managing and editing uploaded images.
14. Performance Optimization
• Optimize the app for speed and responsiveness, as image-heavy apps can be resource-intensive.
15. Mobile Responsiveness :
• Ensure that your image search app is mobile- friendly, with responsive design for various screen sizes.
16. Security and Privacy
• Protect user data and ensure secure communication with any APIs or databases.
• Comply with privacy regulations and provide transparent data usage policies.
17. Error Handling and Feedback
• Implement error handling to provide meaningful feedback to users in case of issues, such as failed searches or server errors.
18. Analytics and User Insights :
• Implement analytics tools to track user behavior and gather insights for app improvement.
19. Testing and Quality Assurance :
• Thoroughly test the app to identify and fix any bugs or usability issues.

ISSN 2581-9429 IJARSCT

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

20. Documentation and Support:
• Create user documentation and provide support options for users who may encounter difficulties.
21. Legal Compliance:
• Ensure your app complies with copyright and licensing laws, especially when dealing with images.
22. Monetization Strategy
• If applicable, plan for monetization options, such as ads, premium features, or subscription models.
23. Continuous Improvement:
• Continuously update and improve the app based on user feedback and changing technology.

VII. RESULT AND DISCUSSION

The results of our image search app using JavaScript are promising and align with the objectives set out in the research. Through rigorous testing and user feedback, we observed high user satisfaction and engagement. Users appreciated the seamless search experience, image preview options, and the ability to customize their preferences. The integration of content-based image retrieval algorithms provided relevant search results, and our responsive design ensured an optimal user experience across different devices. Performance optimization techniques, such as lazy loading and caching, contributed to swift image loading. Furthermore, our security measures ensured data privacy and protection. The discussion highlights that while our app has achieved significant success, ongoing improvements are needed to stay aligned with evolving user expectations and technological advancements. Ethical considerations, including content moderation, will continue to be a focus area for maintaining user trust and satisfaction. Overall, our results demonstrate the potential of JavaScript-based image search apps to offer powerful and user-centric experiences with room for continuous refinement

VIII. REVIEW OF LITERATURE

A review of the literature reveals a multifaceted landscape for developing an image search app using JavaScript. It encompasses a wide array of topics, from core web development principles using JavaScript and its associated libraries to responsive web design strategies that ensure a seamless user experience across diverse devices. Literature also delves into image retrieval techniques, including content-based image retrieval (CBIR) and text-based search algorithms, providing insights into feature extraction and image similarity metrics. User interface (UI) and user experience (UX) design principles play a pivotal role in creating an intuitive and engaging app, and numerous studies offer valuable guidance on these aspects. Additionally, the selection of appropriate APIs and data sources for image integration is influenced by the extensive research on this subject, as well as considerations related to licensing and terms of use. Security, privacy, image optimization, server-side development, performance enhancement, and mobile app adaptation are also explored in depth, further enriching the knowledge base for this project. Ethical concerns and the legal aspects of image usage, including copyright laws, are crucial to consider. Monetization strategies, analytics, and continuous improvement methodologies round out the comprehensive literature landscape, providing a solid foundation for the development of a successful image search app.

A comprehensive review of the literature for developing an image search app using JavaScript reveals a wealth of knowledge encompassing web development, image retrieval, and user experience. Studies in JavaScript and web development emphasize the importance of responsive design and the selection of popular frameworks like React, Angular, or Vue.js to create a versatile and accessible user interface. Research into image retrieval algorithms highlights techniques such as content-based image retrieval (CBIR) and feature extraction, enabling the efficient comparison and ranking of images based on user queries.

Moreover, a strong emphasis on UI and UX design principles is evident in the literature, underlining the significance of usability studies and user testing for creating a user-friendly image search app. Additional domains covered include security measures, privacy considerations, mobile app development, performance optimization, monetization strategies, and ethical concerns, all of which collectively contribute to building a robust, user-centric, and legally compliant image search application. Stay updated with the latest advancements and insights from this body of literature is essential for the continuous improvement of such apps

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 2, November 2023

IX. CONCLUSION

In conclusion, the development of our image search app using JavaScript represents a significant stride in the realm of user-centric digital experiences. Through the application of content-based image retrieval algorithms and responsive web design principles, we have created an intuitive, versatile, and user-friendly platform. Our app not only offers robust search functionality but also considers user preferences, security, and performance optimization as paramount concerns. As the digital landscape continues to evolve, our research underscores the importance of continuous improvement and a strong commitment to ethical and privacy considerations. This innovative app sets a strong precedent for the future of image search applications and encourages further exploration into user-driven web development

REFERENCES

[1] Muller, W. Muller, S. Marchand-Maillet, and D. M. Squire, “Strategies for positive and negative relevance feedback in image retrieval”, International Conference on Pattern Recognition, volume 1 of Computer Vision and Image Analysis, pages 1043{1046, Barcelona, Spain, Sept. 2000a.
[2]Sun, H. Zhang, L. Zhang, and M. Li. “Myphotos a system for home photo management and processing”, In ACM Multimedia Confernce, Juan-les-Pins, France, Dec. 2002, pages 81-82.
[3] Wencheng H. Armitage and P. G. Enser. “Analysis of user need in image archives”, Journal of Information Science, Apr. 1997, 23(4):287-299.
[4] Sumaira Muhammad Hayat Khan, Dr.Ayyaz Hussain, Dr.Imad Fakhri Taha Alshaikhli. “Comparative study on Content-Based Image Retrieval (CBIR)”, International Conference on Advanced Computer Science Applications and Technologies, 2012.
[5] Shouhong Wan, Peiquan Jin and Lihua Yue. “An Effective Image Retrieval Technique Based on Color Perception”, Sixth International Conference on Image and Graphics, 2011.
[6] Abderrahim Khatabi, Amal Tmid, Ahmed Serhir and Hassan Silkan. “Content-Based shape retrieval (CBIR) using different shape descriptors”, April 2014.
[7] Sanjoy Kumar Saha; Amit Kumar Das and Bhabatosh Chanda. “CBIR using Perception based Texture and Colour Measures”, Proceedings of the 17th International Conference on Pattern Recognition (ICPR’04), 2004.
[8] S. Jhansi rani and V. Valli. “Improved Hill Climbing based segmentation (IHCBS) technique for CBIR system”, September 2015
[9] Divya Ragatha Venkata, Deepika Kulshreshtha, and Divakar Yadav, “Techniques for Refreshing Images in Web Documents,” Proceedings for International Conference on Control, Robotics and Cybermetics, October 2011.
[10] Rishi Mukhopadhyay, Aiyesha Ma and Ishwar K. Sethi, “Pathfinder Networks for Content Based Image Retrieval Based on Automated Shape Feature discovery”,2012.
[11] Daniella Stan. eID: A system for Exploration of Image Databases, phD thesis, Oakland University, 2002. [5] Chaomei Chen, George Gagaudakis, and Paul Rosin. Similarity-based image browsing. Proceedings of the 16th IFIP World Computer Congress, August 2000.