Advance Image Downloader/Extractor API

Revision Number: 1.0

Last date of revision: 07/01/2023

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| 07/01/2023 | 1 | Initial HLD — V1.0 | Darpanarayan Bahadur |
|  |  |  |  |

## Contents

[Document Version Control. 2](#_TOC_250021)

[Abstract. 4](#_TOC_250020)

1. [Introduction 5](#_TOC_250019)
   1. [Why this High-Level Design Document? 5](#_TOC_250018)
   2. [Scope. 5](#_TOC_250017)
   3. [Definitions 5](#_TOC_250016)
2. [General Description. 6](#_TOC_250015)
   1. [Product Perspective 6](#_TOC_250014)
   2. [Problem statement 6](#_TOC_250013)
   3. [Proposed Solution 6](#_TOC_250012)
   4. [Further Improvements 6](#_TOC_250011)
   5. [Technical Requirements. 6](#_TOC_250010)
   6. [Data Requirements 7](#_TOC_250009)
   7. [Tools used. 8](#_TOC_250008)
   8. [Constraints 9](#_TOC_250007)
   9. [Assumptions. 9](#_TOC_250006)
3. [Design Details 10](#_TOC_250005)
   1. [Process Flow. 10](#_TOC_250004)
   2. [Event log 11](#_TOC_250001)
   3. [Error Handling 11](#_TOC_250000)
4. [Performance 10](#_TOC_250005)
   1. Reusability. 12
   2. Portability 12
   3. Security 12
   4. Resource Utilization 12
   5. Deployment. 12
   6. Key Performance Indicator. 12
5. Conclusion 14

Abstract

Image scraping is a process of extracting images from websites and other sources, and it can be used to gather wide range of data types, including images. Image scraping from website has become increasingly necessary in recent years due to the growing importance of visual content in the digital world. The use of images and other visual content has become essential for business and individuals to effectively communicate and engage with their audience to effectively communicate and engage with their audience online.

This API does exactly what it says, it can download as per users’ requirements of any kind at any date and time and from some of the most popular websites. Developers can use this website with any platform they want and have to make a post request with some required key and value pair and the download link will be ready to download the images once process is completed.

1. **Introduction**

###### Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* + - Present all of the design aspects and define them in detail
    - Describe the API being implemented
    - Describe the hardware and software interfaces
    - Describe the performance requirements
    - Include design features and the architecture of the project
    - List and describe the non-functional attributes like: o Security
      * Reliability
      * Maintainability
      * Portability
      * Reusability
      * Application compatibility
      * Resource utilization
      * Serviceability

##### Scope

The HLD documentation presents the structure of the system, such as the database architecture, API architecture (layers), API flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

* 1. **Definitions**



*Term*

*API*

*Database*

*IDE AWS*

*Description*

Application programming interface

Collection of all the information monitored by this system

Integrated Development Environment

Amazon Web Services

### General Description

#### Product Perspective

Advance Image Downloader/Extractor (Job) is a python based API, which will help the user download the image of any kind. These images will get downloaded as a job and the user will get notified over an email about the job completion. This web app will allow for multiple developer to integrate this API in their application to submit or interact with the program at the same time.

* 1. Problem statement

There is a growing demand for the ability to extract information from images in an automated and reliable manner. Current solutions either lack the ability to extract a wide range of information or are not accurate enough for practical use. As a result, developers seeking to build applications that rely on image recognition and analysis face significant challenges and limitations.

Create a job to download thousands of images from the internet for given requirements (e.g.: Cat, Dog). Let’s use Python and some web scraping techniques to download images.

The following use cases can be implemented.

* + - To download the specified number of images of the particular query from specified source.
    - To notify with the downloadable link to the user over an email
    - To let user, download the images through just single click on link.
    - To submit multiple job request by single/multiple users.
  1. PROPOSED SOLUTION

The solution proposed here is an Advanced Image Downloader/Extractor python-based API that can be implemented to perform above mention use cases. In the first case, the user has to hit a request with some required fields like name, schedule, email, etc. to get the job done. Once the job is completed, the user will get the confirmation link which can be used to download the images.

* 1. FURTHER IMPROVEMENTS
* The API could allow users to specify certain criteria for selecting images to scrape, such as image size, file type, or resolution.
* The API could automatically tag images with relevant keywords to make it easier for users to find and organize the scraped images.
* The API could analyze the structure and content of web pages to identify the most relevant images and improve the accuracy of the scraping process.
* Some websites may use CAPTCHAs to prevent automated scraping. The API could include a CAPTCHA bypass feature to allow it to continue scraping images from these sites.
* The API could use multiple concurrent connections to scrape images from multiple websites simultaneously, improving the speed of the scraping process.
* The API could include a feature that automatically retries failed image scrapes to improve the reliability of the scraping process.
* The API could extract metadata such as the source URL, file size, and dimensions of the scraped images.
* The API could integrate with other image analysis or data analysis APIs to provide additional functionality.

#### Technical Requirements

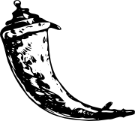
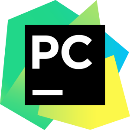
* The API will need a library or tool for executing the web scraping process, such as Beautiful Soup or Selenium.
* The API will need a library or tool for making HTTP requests to websites and receiving responses, such as requests.
* The API may need a library or tool for processing and manipulating the scraped images, such as Pillow.
* The API may need a database to store information about the scraped images and metadata, such as MySQL or MongoDB.
* The API will need a programming language for implementing the server-side logic, such as Python.
* The API will need a hosting environment, such as a cloud platform or on-premises servers, to run and serve the API.
* If the API is expected to handle a high volume of requests, it may need a load balancer to distribute incoming requests across multiple servers and improve the scalability of the API.
* The API will need monitoring and logging tools to track its performance and identify any issues that may arise.

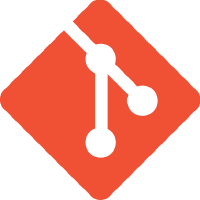
#### Data Requirement

The API will need user’s email, date and time, search query, number of images for performing this job and source from where it will be downloaded

#### Tools used

PyCharm as IDE, Python programming language, AWS, database, version control system and frameworks such as Flask, selenium are used to build the whole model.





* + - PyCharm is used as IDE.
    - Selenium is used for web-driver.
    - AWS is used for deployment of the model.
    - Cassandra is used to retrieve, insert, delete, and update the database.
    - Python Flask is used for routing.
    - Git and GitHub is used as version control system.

#### Constraints

The API is constrained by the technical capabilities of the web scraping library, image processing library, and hosting environment being used. For example, if the API is using a resource-intensive image processing library, it may not be able to handle a high volume of requests. The need to handle a large volume of requests from multiple users simultaneously. Ensuring that the API is scalable and can handle this demand may be a constraint. The need to ensure the security and privacy of the scraped images and metadata, as well as protect against unauthorized access or misuse of the API. Easy for users to understand and use, with comprehensive documentation and examples. Ensuring a good user experience may be a constraint.

#### Assumptions

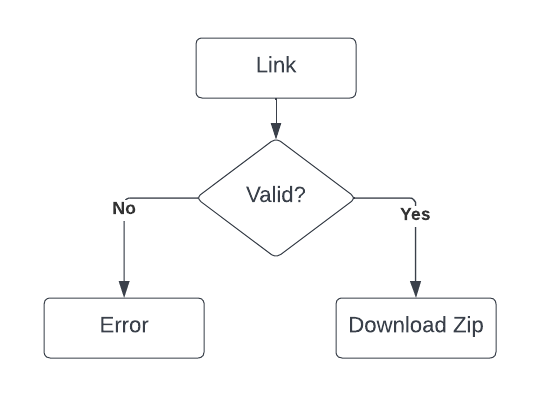
Assumption is that the user must have enough space in its hardware where the downloaded images can be stored. If the space inside the hardware is not sufficient enough then the application may not work as intended.

## Design Details

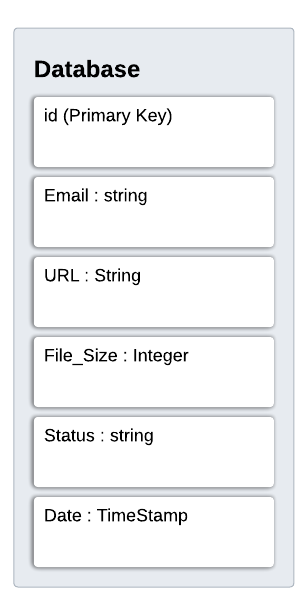
##### Process Flow

Process Image Downloader API follows the following architecture:

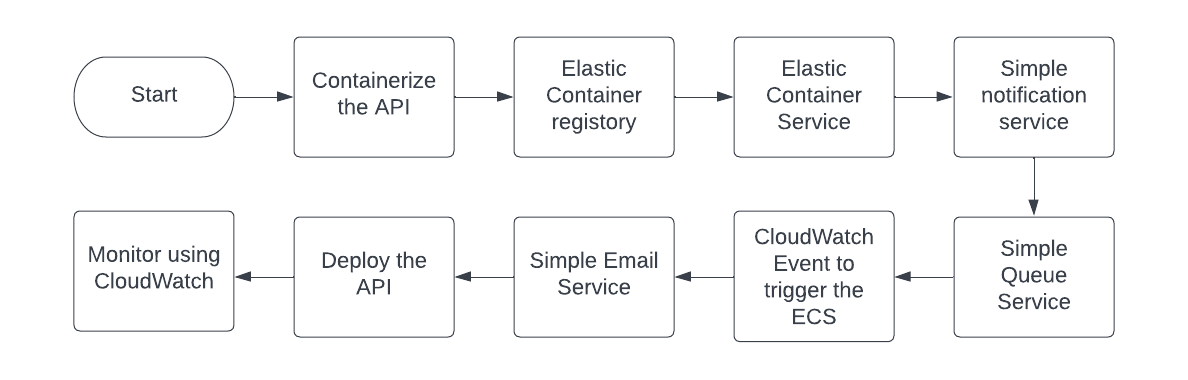
##### Proposed methodology



##### Database Design



##### Deployment Process



* 1. Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.
   1. **Error Handling**

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

1. Performance

For everything to run smoothly the API should be designed to scrape images from websites in an efficient manner, such as using multiple concurrent connections and avoiding unnecessary requests. The API should use image processing libraries that are optimized for speed, and should minimize the number of image processing operations performed on each image. The API server should have sufficient CPU, memory, and storage resources to handle the expected volume of requests. The API could cache frequently accessed images and metadata to reduce the number of scraping and processing operations needed. If the API is expected to handle a high volume of requests, it may need to use a load balancer to distribute incoming requests across multiple servers and improve scalability. The API should be regularly monitored using tools such as CloudWatch to identify any performance issues, and optimization techniques should be applied as needed.

#### Reusability

The code written and the components used should have the ability to be reused with no problems.

#### Portability

Since this is an API, It can be accessed via any platform until and unless that platform is connect to the internet.

#### Security

Since we are capturing the user’s email address, we have added the functionality by which the user request which consist of user’s email, search query etc. will get deleted from the database after certain interval of time (usually after 30 minutes).

#### Resource Utilization

For each user request, cloud will use multiple threads to simulate the multithreading environment until that process is finished. Databases has to perform retrieving, inserting and deletion operation until that process gets removed from the database.

* 1. **Deployment**



#### KPls (Key Performance Indicators)

1. Request volume: The number of requests received by the API per unit of time, such as requests per second or requests per minute.
2. Error rate: The percentage of requests that result in an error, such as a failed scrape or invalid request.
3. Response time: The time it takes for the API to process a request and return a response, measured in milliseconds or seconds.
4. Image quality: The accuracy and relevance of the scraped images, as measured by user feedback or manual evaluation.
5. Resource utilization: The percentage of CPU, memory, and other resources used by the API, as well as the API's impact on other resources such as the database or web scraping library.
6. User satisfaction: The satisfaction of users with the API, as measured by surveys or other feedback mechanisms.

## Conclusion

The Advance Image Downloader/Extractor (Job) will help the user download hundreds of images in single click. To simplify the process where at time we require bunch of images to work with.