

## **Advance Image Downloader/Extractor**

**Revision Number: 1.0**

**Last date of revision: 29-08-2021**

## Document Version Control

Date Issued	Version	Description	Author
29-08-2021	1	Initial HLD – V1.0	Shreyas Parab

## Contents

Document Version Control. ....	2
Abstract.....	4
1 Introduction.....	5
1.1 Why this High-Level Design Document? .....	5
1.2 Scope. ....	5
1.3 Definitions .....	5
2 General Description. ....	6
2.1 Product Perspective .....	6
2.2 Problem Statement .....	6
2.3 Proposed Solution .....	6
2.4 Further Improvements .....	6
2.5 Data Requirements .....	6
2.6 Constraints.....	7
2.7 Assumptions. ....	7
3 Design Details .....	8
3.1 Main Design Features .....	8
3.2 Process Flows.....	8
3.3 Database Design .....	9
3.4 User Interface .....	9
3.5 Event log.....	10
3.6 Error Handling.....	10
3.7 Performance. ....	10
3.8 Reusability. ....	11
3.9 Portability .....	11
3.10 Security .....	11
3.11 Resource Utilization .....	11
3.12 Deployment. ....	11
4 Conclusion .....	12

## Abstract

In this time, the images are the most important data source. Be it a training Computer vision model on this, Finding the appealing wallpaper images, going through hundreds of crafts and arts varieties on single click or finding the news related to specific company for market analysis images are crucial in this scenario. This app does exactly what it says, it can download up to 500 images of any kind at any date and time. User just have to submit the query and the download link will be ready to download the images once process is completed.

# 1. Introduction

## 1.1 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 user interface being implemented
- Describe the software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
  - Security
  - Reliability
  - Portability
  - Reusability
  - Resource Utilization

## 1.2 Scope

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

## 1.3 Definitions

Term	Description
Database	Collection of all the information monitored by the system
IDE	Integrated Development Environment
Heroku	Heroku Cloud Service

## 2. General Description

### 2.1 Product Perspective

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

### 2.2 Problem Statement

There are often a times, we need bunch images to work with. We can consider the example such as training the Neural networks over the Cat and Dog images or going through hundreds of design element for designing the website. In such scenario's we need hundreds of images right away. This problem can be solved using the Advance Image Downloader application. The following use cases can be implemented:

- To download the specified number of images of the particular query
- 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 the single/multiple users

### 2.3 Proposed Solution

The solution proposed here is an Advance Image Downloader/Extractor python web-based application that can be implemented to perform above mention use cases. In the first case, the user has to input the search query, time when user wants job to run, email and the number of images user wants. This job will get executed at the date and time mentioned by the user. Once the job is completed, the user will get the confirmation link which can be used to download the images.

### 2.4 Further Improvements

- Letting user input the Image type they want in result. E.g., JPEG, PNG etc.
- We can show the expected time required to finish the job.
- Size of the downloaded files can be sent over an email.
- To give the country input so that user can schedule the job at their particular country time.

### 2.5 Data Requirements

- We will need user's Email, Date and time, search query and number of images for performing this job.

### 2.6 Tools Used

- Python Flask is used for backend development

- Front end is developed using HTML, CSS and JavaScript
- PyCharm is used as an IDE for developing this software
- Heroku is used for deployment of the model
- Cassandra is used to retrieve, insert, delete and update the databases
- Git is used as version control system



## 2.7 Constraints

It should have user friendly design and working. User should not be required to know how the images are getting downloaded at the backend. Internet connection is also a constraint for the application. Since the application fetched the data from the database and over the internet, it is crucial that there is an Internet connection for the application to function. Since the user can make multiple requests at same time, it may be forced to queue incoming requests and therefore increase the time it takes to provide the response.

## 2.8 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.

### 3. Design Details

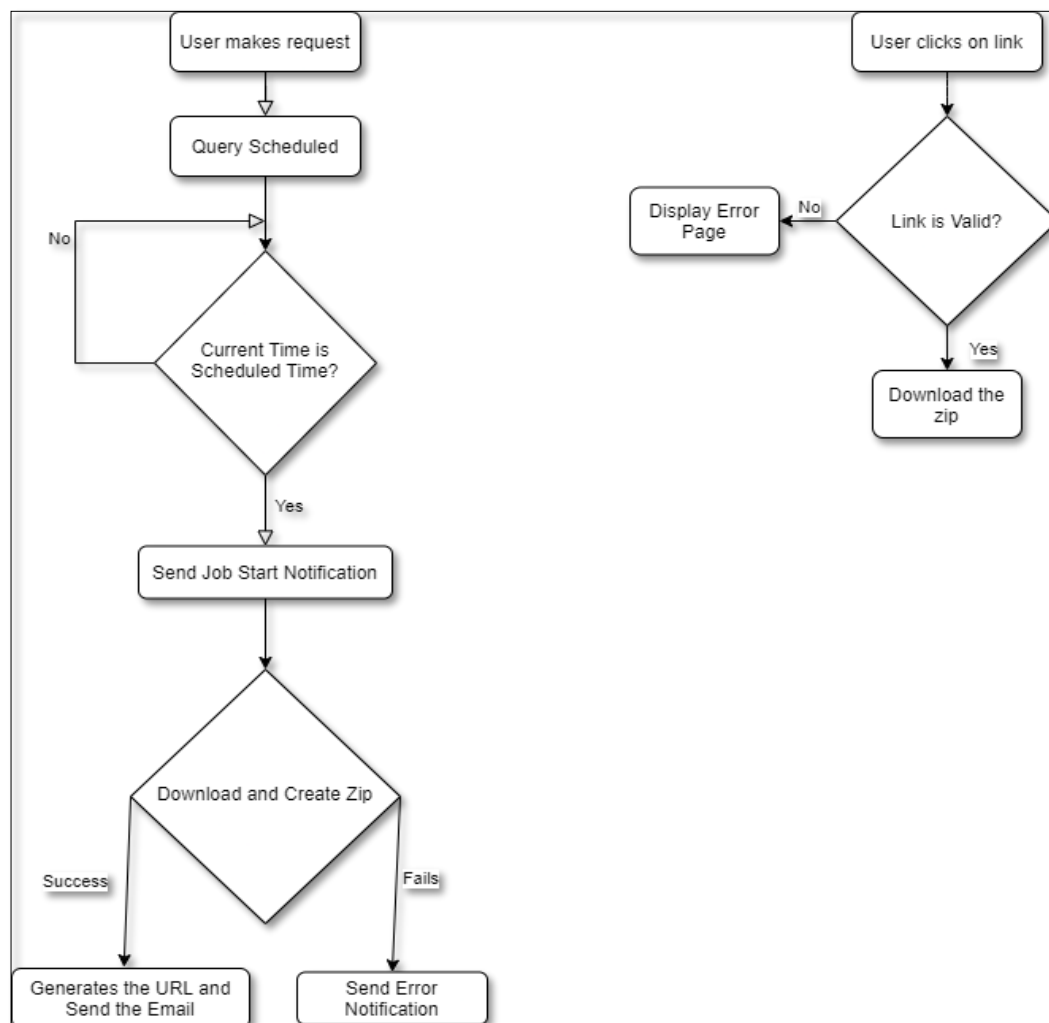
#### 3.1 Main Design Features

The main design features include parts such as Architecture, The User Interface Design, The Database, Process Relation. In order to make these designs easier to understand, the design has been attached with diagrams such as Flow Diagram.

#### 3.2 Process Flow

Advance Image Downloader app follows the following architecture:

##### Proposed Methodology



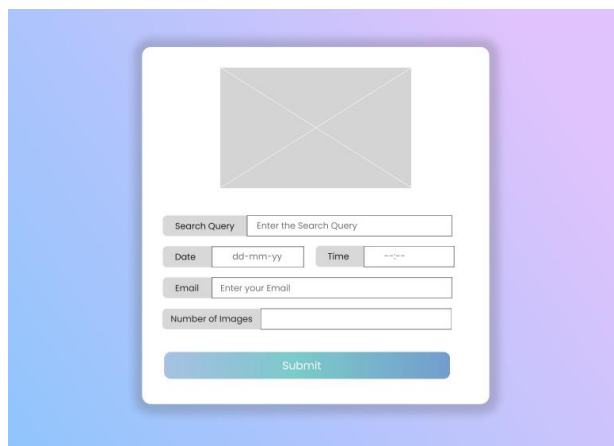
### 3.3 Database Design

The database schema consists of req\_id (uuid datatype), email (string datatype), url (string datatype)

AstralImage
req_id : uuid
email : string
url : string

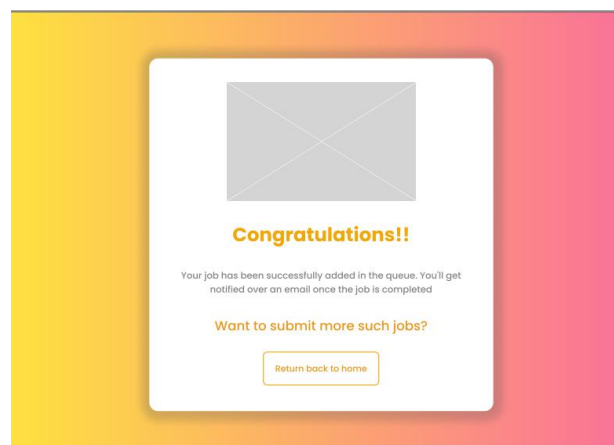
### 3.4 User Interface

#### 1. Homepage



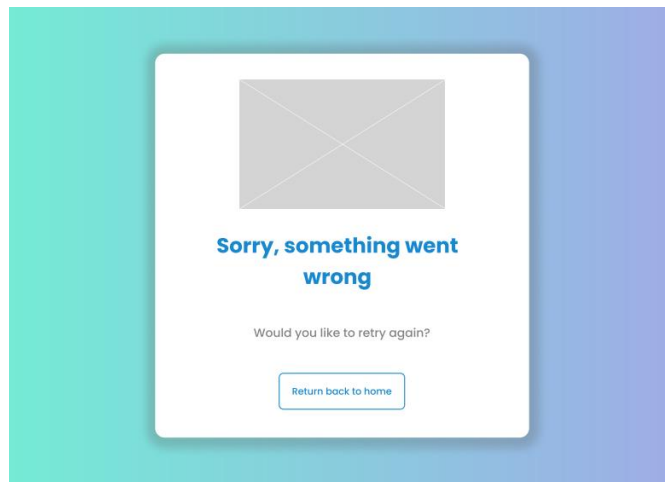
The homepage features a white form card on a blue-to-purple gradient background. At the top is a large gray placeholder for a profile picture. Below it are several input fields: 'Search Query' with a placeholder 'Enter the Search Query', 'Date' with a placeholder 'dd-mm-yy', 'Time' with a placeholder 'hh:mm', 'Email' with a placeholder 'Enter your Email', and 'Number of Images' with a placeholder '1'. A green 'Submit' button is at the bottom.

#### 2. Job Submitted



The 'Job Submitted' screen features a white card on a yellow-to-pink gradient background. It includes a large gray placeholder for a profile picture. Below the placeholder, the text 'Congratulations!!' is displayed in bold orange. A message follows: 'Your job has been successfully added in the queue. You'll get notified over an email once the job is completed'. Below this, the text 'Want to submit more such jobs?' is shown in orange. At the bottom is a yellow 'Return back to home' button.

### 3. Error Page



### 3.5 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 logging file as well.
4. System must be able to handle logging at greater scale because it helps debugging the issue and hence it is mandatory to do.

### 3.6 Error Handling

Error handling is done in two ways:

1. **UI part** – If user performs some incorrect action on UI, then the error page will be shown to the user which will have the appropriate error message.
2. **Email Part** – If the error comes while handling the user request at the backend, then user will receive an email regarding the same and repeating the job submission again.

### 3.7 Performance

For everything to run smoothly the app must be able to handle multiple requests at similar time to give the best user experience as possible. It should also be able to give the results back in the earliest time possible without making user to wait for result longer amount of time. User should get appropriate result from the app about what he/she is expecting to improve the performance of search in the app itself. The database server must be able to need to keep up with the database requests without any failure and data losses.

### 3.8 Reusability

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

### 3.9 Portability

Since this is a web app, it can be accessed via any OS system until and unless that OS is connected to the internet.

### 3.10 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).

### 3.11 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.

### 3.12 Deployment

The deployment of this web app is done on Heroku cloud. Deployment steps are as follows:

1. You can go over this GitHub repo to clone the app  
<https://github.com/Sparab16/Advance-Image-Downloader>
2. Create a new virtual environment
3. Setup the config files given in the repository
4. Create a new environment on Heroku
5. Install two build packs in the environment
  - a. <https://github.com/heroku/heroku-buildpack-chromedriver>
  - b. <https://github.com/heroku/heroku-buildpack-google-chrome>
6. Run the following commands in the terminal in your project root directory
  - a. If you haven't already logged into Heroku account

```
$ heroku login
```

- b. Deploy your changes

```
$ git add .  
$ git commit -am "Deployed on heroku"  
$ git push heroku master
```

## 4. Conclusion

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