

## Advance Image Downloader/Extractor

### Low Level Design Document

Written By	Harshad Kadam, Shreyas Parab
Revision Number	<b>1.0</b>
Last date of revision	<b>31-08-2021</b>

## Document Version Control

Date Issued	Version	Description	Author
31/08/2021	1	Initial LLD – V1.0	Shreyas Parab

## Contents

<b>Document Version Control.</b>	<b>2</b>
<b>Abstract.</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
<b>1.1 Why this Low-Level Design Document?</b>	<b>5</b>
<b>1.2 Scope .....</b>	<b>5</b>
<b>1.3 Constraints .....</b>	<b>5</b>
<b>1.4 Definitions .....</b>	<b>5</b>
<b>2 Technical Specifications .....</b>	<b>6</b>
<b>2.1 Architecture Design .....</b>	<b>6</b>
<b>2.2 User Interface .....</b>	<b>6</b>
<b>2.3 Schedulers .....</b>	<b>7</b>
<b>2.4 Email Notification .....</b>	<b>7</b>
<b>2.5 Web Scrapping .....</b>	<b>8</b>
<b>2.6 Database .....</b>	<b>8</b>
<b>2.7 File Creation .....</b>	<b>9</b>
<b>2.8 Event Log .....</b>	<b>9</b>
<b>2.9 Error Handling .....</b>	<b>10</b>
<b>2.10 Deployment .....</b>	<b>10</b>
<b>3 Technology Stack .....</b>	<b>11</b>
<b>4 User's Flow .....</b>	<b>11</b>
<b>5 Test Cases .....</b>	<b>12</b>

## 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 Low-Level Design Document?

The purpose of this document is to provide a detailed description of the Advance Image Downloader/Extractor system. It will explain the purpose and features of the system, the interfaces of the system, what system will do, the constraints under which it must operate and how the system will react to external process. This document is intended for developers of the system.

## 1.2 Scope

This is a web-based application, and it is designed to download the images of the search query that user puts in.

## 1.3 Constraints

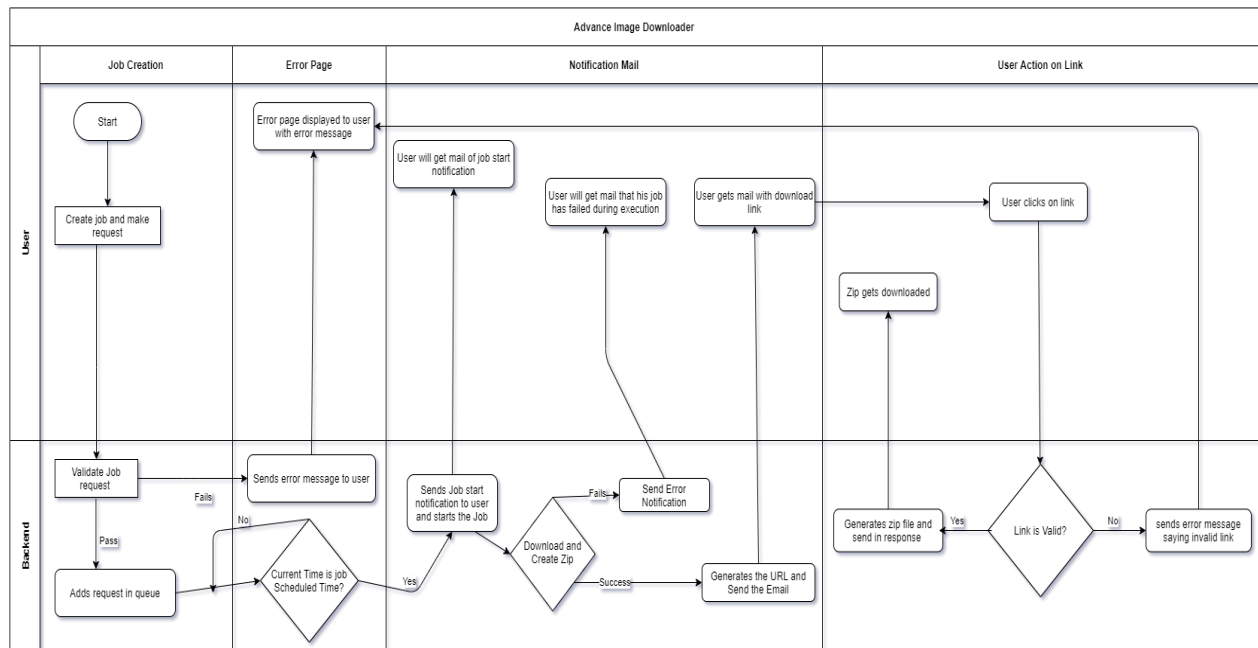
Internet connection is a constraint for the application. Since the application fetched the data from the database and the internet, it is crucial that there is an Internet connection for the application to function. Since the user can make multiple requests at same time, it may be forced to queue incoming requests and therefore increase the time it takes to provide the response.

## 1.4 Definitions

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

## 2. Technical Specifications

### 2.1 Architecture Design

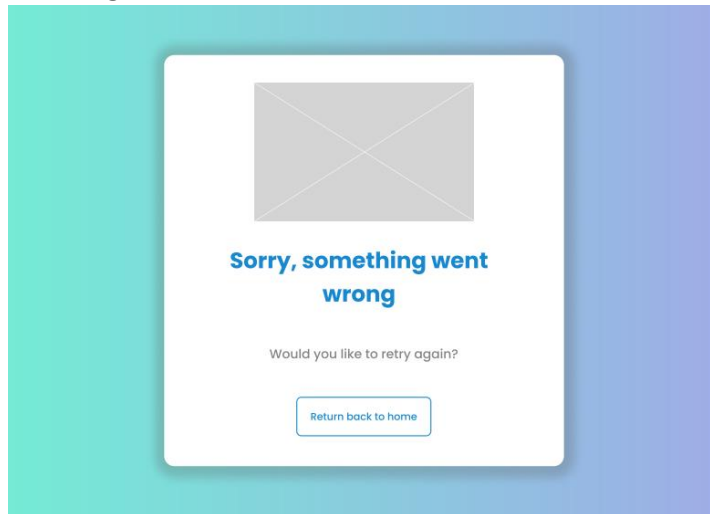
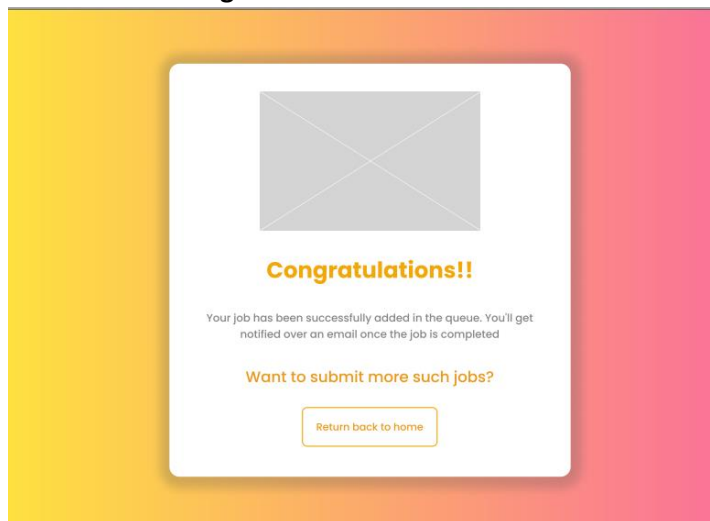


### 2.2 User Interface

The User Interface of this application is made using HTML, CSS, JavaScript and Bootstrap. HTML, CSS and Bootstrap are used for giving the custom style for the web app and also the skeleton is designed here. JavaScript is used for restricting the user not enter the past dates. The UI part consist of three pages which are as following:

#### a. Home Page

The screenshot shows a web form for downloading images. At the top is a large placeholder for a search query. Below it are input fields for 'Search Query' (with a hint 'Enter the Search Query'), 'Date' (format dd-mm-yy), 'Time' (format hh:mm), 'Email' (with a hint 'Enter your Email'), and 'Number of Images'. A blue 'Submit' button is at the bottom of the form.

**b. Error Page****c. Job Submitted Page****2.3 Schedulers**

The backend of this application is designed using Flask framework. Since this is the Job based web application user's request are submitted as a job request which will be queued until the given time comes.

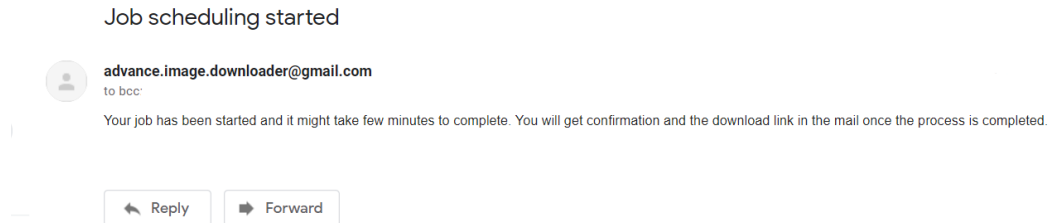
For the implementation of schedulers, [apscheduler](#) library is been used. Using the aptscheduler, we were able to handle the multiple requests of users and allow them to run at the specified time.

**2.4 Email Notification**

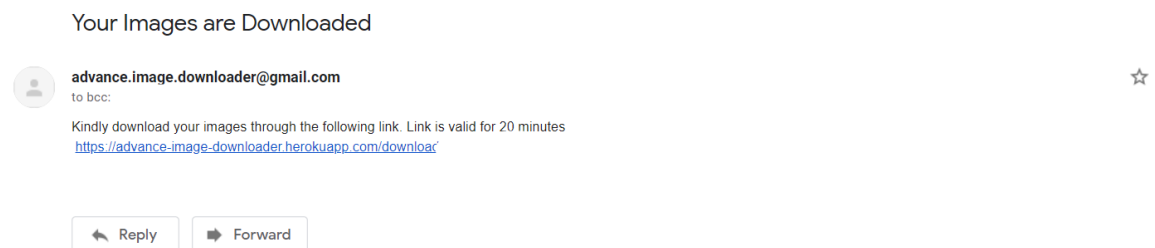
The user feedback is must for any application to support the great user experience. This is achieved by using the [smtplib](#) library of python. This library provides the interface to send the email on the given recipient's email address. The Gmail is used

as an email client to send the notification to the user. There are three types of email notification that the user can receive.

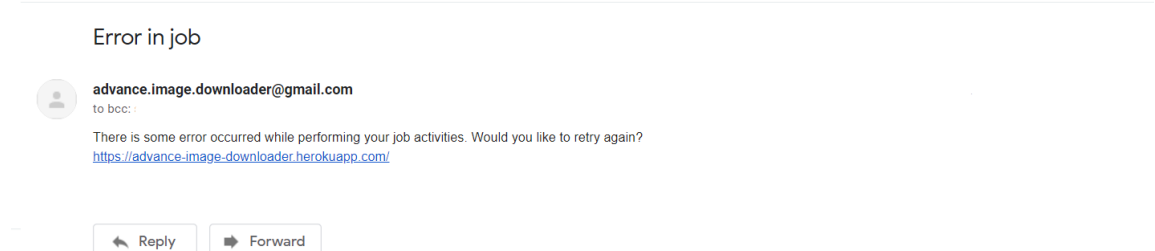
- a. **Job started** – Once the current date and time is equal to the scheduled date and time user will be notified by the email that the scheduling job has started.



- b. **Job ended** – Once the images get downloaded and files are created at server side, user will get notified with the download link over an email.



- c. **Job error** – If there is some error occurs at the backend side, the user will get notified of the same by receiving the job error notification.




## 2.5 Web Scrapping

The images need to be downloaded and then stored inside the database for further processing. This downloading from the internet part is done using [selenium](#) and [google-chrome-driver](#). User's search query will be fed to selenium web browser which will find the images related to search query and then inserted into the databases.

## 2.6 Database

The images which are scrapped using the selenium has to be stored somewhere and hence we have used the [Cassandra](#) Database for the same. Rather than storing the images inside the database, we are just storing the URL of the images to make it more space efficient in that case. The following database schema we have used:



 <b>AstralImage</b>
req_id : uuid
email : string
url : string

It consists of three columns, req\_id (datatype = uuid) is a unique request id which will be generated for each user request. Email (datatype = string) is the email address of the user who have requested for the images. url (datatype = string) is the link of the images which needs to be downloaded from the server.

Here all the three columns are set as the primary key, since all the three must have to be unique at every case (To not have duplicate images url inside the database)

## 2.7 Files Creation

Once the images url is saved inside the database. This URL's will be then used to download the images from the server using the python's [request](#) library. These downloaded images are then saved inside the temporary folder which will be then zipped and send as file once user clicks on the download link from email. This files which are created are then deleted after certain interval of time (usually 30 minutes). Also, the database records will get wiped out from the database after this time.

## 2.8 Event Log

The system should log every event so that the user will know what process is running internally. Logging is implemented using python's standard [logging](#) library.

### 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.

## 2.9 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.

## 2.10 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
```

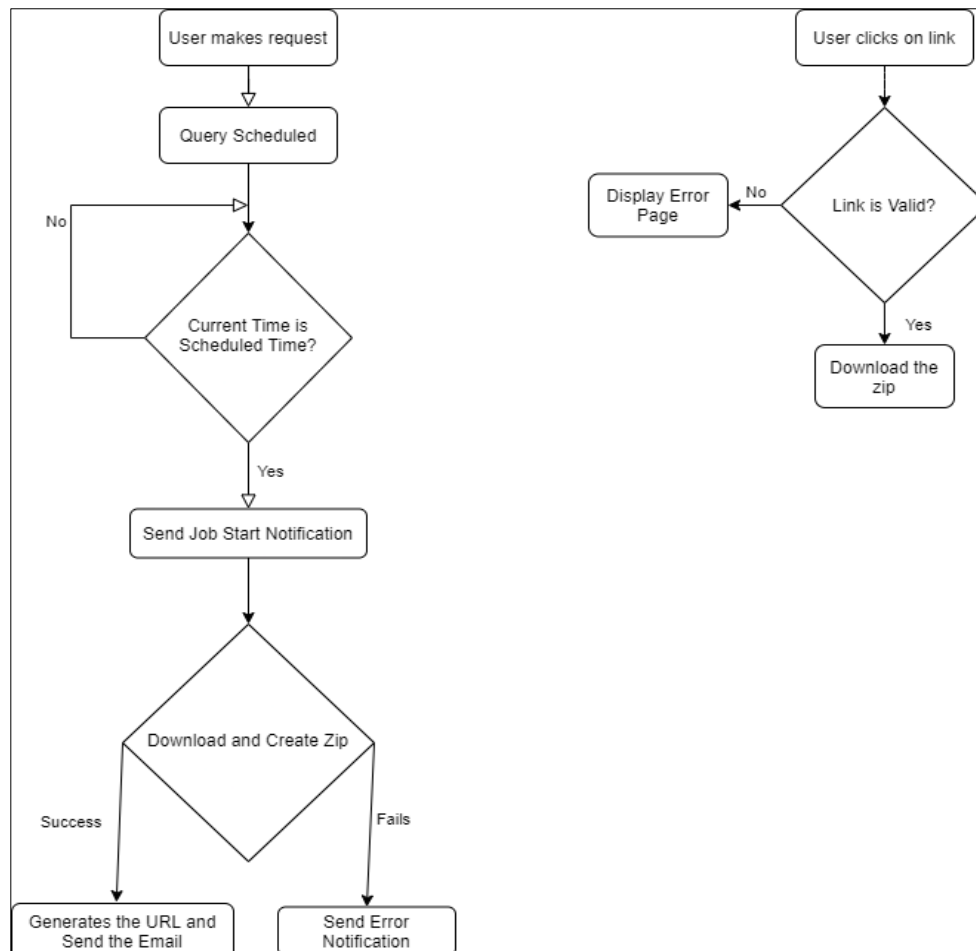
- b. Deploy your changes

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

### 3. Technology Stack

Front End	HTML, CSS, JS, Bootstrap
Back End	Python Flask, Selenium
Database	Cassandra
Deployment	Heroku

### 4. User's Flow



## 5. Test Cases

Test Case Description	Request Body	Expected Results
All fields with valid data	<pre>{   "search-query": "Best hd wallpaper"   "date": "2021-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "145" }</pre>	Job will get queued (Success)
Empty string in search query field	<pre>{   "search-query": ""   "date": "2021-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "145" }</pre>	User will get error message that search query is required (Failed)
Empty date in date field	<pre>{   "search-query": "Best hd wallpaper"   "date": ""   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "145" }</pre>	User will get error message that date is required (Failed)
selected date is less than current date (let's say current date is "2021-08-30" and current time is "11:30")	<pre>{   "search-query": "Best hd wallpaper"   "date": "2020-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "145" }</pre>	User will get error page showing error message (Failed)

Empty time in time field	{ "search-query": "Best hd wallpaper" "date": "2021-09-20" "time": "" "email": "wsq.wee@example.com" "images": "145" }	User will get error message that time is required (Failed)
selected date and time is less than current date and time (let's say current date is "2021-08-30" and current time is "11:30")	{ "search-query": "Best hd wallpaper" "date": "2021-08-30" "time": "11:28" "email": "wsq.wee@example.com" "images": "145" }	User will get error page showing error message (Failed)
Empty email in email field	{ "search-query": "Best hd wallpaper" "date": "2021-09-20" "time": "1:45" "email": "" "images": "145" }	User will get error message that email is required (Failed)
Invalid email in email field	{ "search-query": "Best hd wallpaper" "date": "2021-09-20" "time": "11:45" "email": "wsq.wee@" "images": "145" }	User will get error message that email is invalid (Failed)
Empty string in no of images field	{ "search-query": "Best hd wallpaper" "date": "2021-09-20" "time": "11:41" "email": "wsq.wee@example.com" "images": "" }	User will get error message that no of images is required (Failed)

Invalid string in no of images field	<pre>{   "search-query": "Best hd wallpaper"   "date": "2021-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "-1" }</pre>	User will get error message that no of images can should be greater than 0 (Failed)
0 in no of images field	<pre>{   "search-query": "Best hd wallpaper"   "date": "2021-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "0" }</pre>	User will get error message that no of images should be greater than 0 (Failed)
501 in no of images field	<pre>{   "search-query": "Best hd wallpaper"   "date": "2021-09-20"   "time": "11:41"   "email": "wsq.wee@example.com"   "images": "501" }</pre>	User will get error message that no of images can not be greater than 500 (Failed)
On receiving download link , user clicks on link with in expiration time		zip will get downloaded (Success)

---

On receiving download link ,  
user clicks on link after  
expiration time

Error page will be shown  
saying invalid link (Failed)