**Software Design**

**CMSC 495 Week 3**

**By: Shelley Schoppert, Jake Gonzalez, Justin Smith, Timothy Strickland**

**University of Maryland, Global Campus**

**9/11/2022**

# **Table of Contents:**

[**Table of Contents:**](#_ba7l4058zpnw) **1**

[**Introduction**](#_e13dz768d923) **2**

[Purpose:](#_kr0g222m266w) 2

[Scope:](#_5wzvfg8tisry) 2

[Definition:](#_140rccmyfhf4) 2

[**References**](#_qw3n22sx1ny2) **3**

[**Decomposition description:**](#_vnwb3kr23dzz) **4**

[Module Description:](#_9djy8lqif43c) 4

[User Interface:](#_v16l9ll0734r) 4

[Home page](#_jyzihuf9vtbd) 4

[About page](#_caqjt49o6jtx) 4

[Results/Data page](#_grtonoqysbae) 4

[Data Manager:](#_nrceitrsgmww) 4

[**Dependency Diagrams:**](#_6mihmhndbhb8) **5**

[**Detailed Design:**](#_cum88lok36mp) **7**

[ExifManager.py](#_6tzdfv3p7lp1) 7

[Data\_Info.py](#_scrotkj1gout) 7

[Web.py](#_pj3atlt16npu) 8

[**Known Issues:**](#_l8xz8d6tc3vv) **9**

# 

# **Introduction**

## **Purpose**:

The purpose of this document is to lay out the design of the Metadata Analyzation program.

## **Scope**:

The program will lay-out the design structure of the program. The user interface will be accessed through a web browser. There are three main modules that will support the user interface.

## **Definition**:

EXIF - Exchangeable Image File

GUI - Graphical User Interface

JPEG - Joint Photographic Experts Group

PNG - Portable Network Graphics

CSV - Comma Separated Values

GPS - Global Positioning System

# 

# **References**

References for this document include the following:

* · IEEE Std. 1233-98: IEEE Guide for Developing System Requirements Specification
* · IEEE Std. 1058-1998: IEEE Standard for Software Project Management Plans

# **Decomposition description**:

## **Module Description**:

There are four main modules in this system: The User Interface and the Data Manager.

## **User Interface**:

### Home page

* + Upload file button – Allows user to select a photo to upload. (May need to limit size and amount).
  + Submit button – when clicked
    - checks for and only accepts .jpeg, .jpg, and .png files
    - sends data to backend to pull EXIF data
    - Redirects user to the Results/Data page
  + Link to About page – when clicked takes user to About page

### About page

* + Describes to user the how to use the application and what type of data is being pulled from the image
  + Link to Home page – redirects user to the Home page

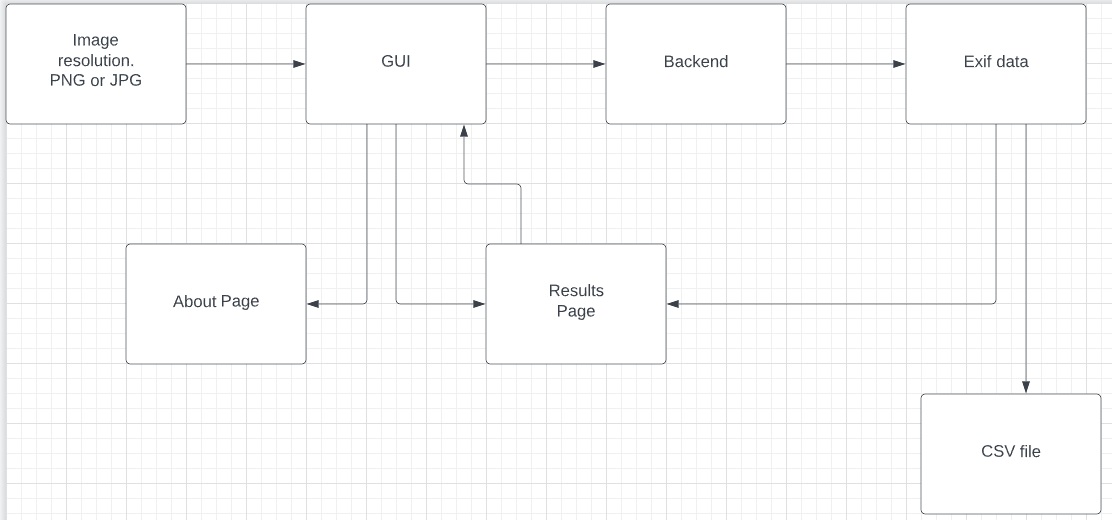
### Results/Data page

* + Results of EXIF data will be displayed for user
  + Button to redirect user to upload another photo on the Home page
  + Link to About page - when clicked takes user to About page
  + Link to Home page - when clicked takes user to Home page
  + Button to export EXIF data to a CSV that the user can save locally

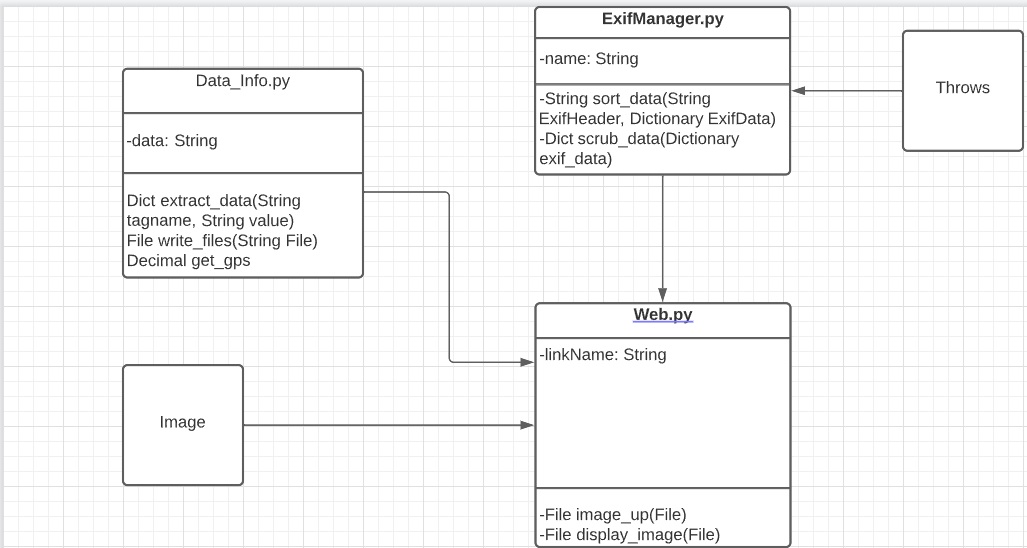
## **Data Manager**:

The Data Manager handles the data processing from the user interface. In general the images loaded into the user interface will be processed by one or more of three classes. ExifManager.py, Web.py and/or Data\_Info.py. These classes are explained later.

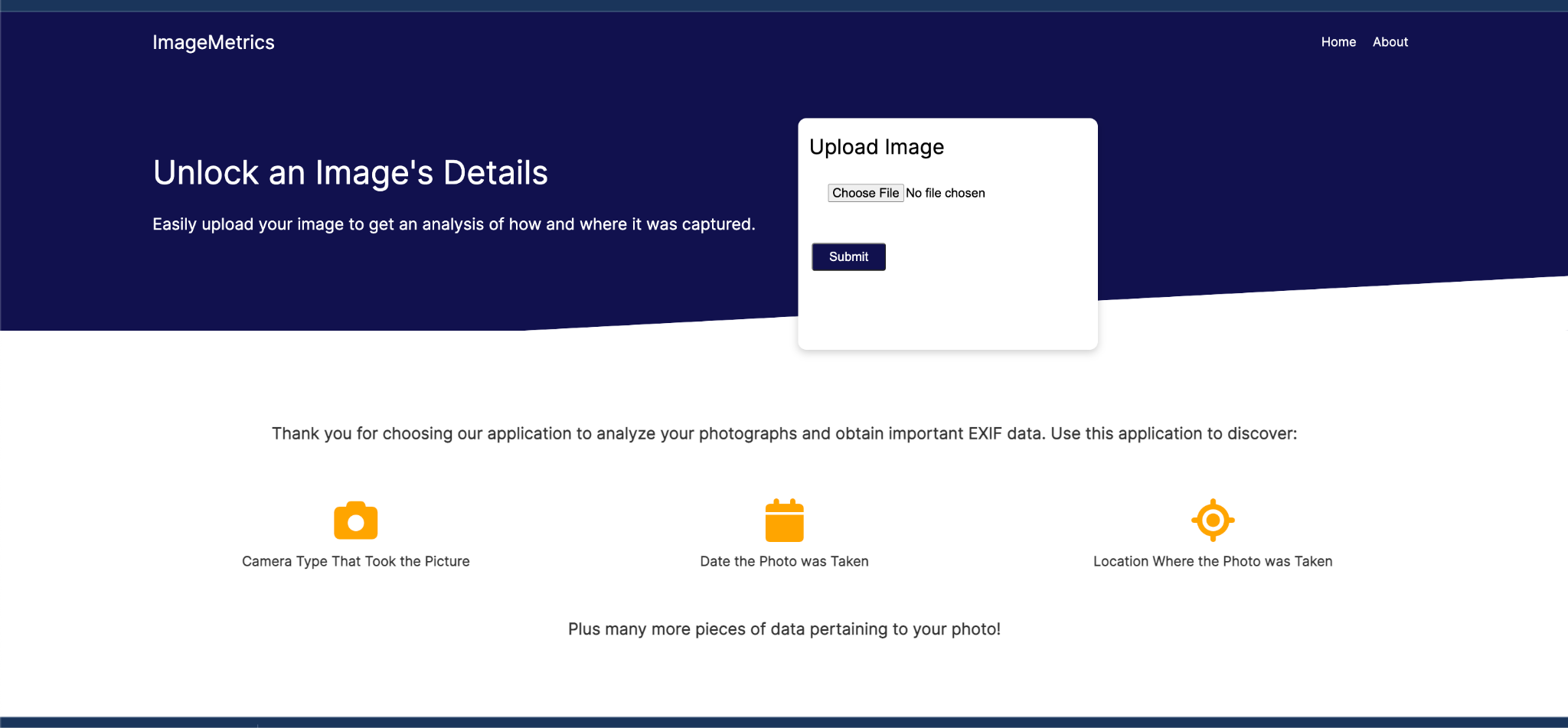
# **Dependency Diagrams**:

****

**Logical Flow Diagram**

****

**Class Diagram**

****

**User Interface Screenshot**

# **Detailed Design**:

## **ExifManager.py**

sort\_data(String exif\_tag, Dictionary exif\_data)

* Check if the requested exif tag is present as a key in exif\_data
* Throws if the tag doesn’t exist
* Return the requested tag value

scrub\_data(Dictionary exif\_data)

* Copy the param dictionary to a new variable
* Cannot iterate and remove from the same dictionary at the same time
* Iterate over the original param
* If a key:value pair is found that does not have meaningful data (none or an empty string), remove it from the new dictionary
* Return the new (scrubbed) dictionary

## **Data\_Info.py**

get\_data(filename)

* Define how data will be opened via image and file.

extract\_data(exifdata)

* + Loop through tags
  + Decode values
  + num of expected tags
  + Return dictionary of exif data

write\_files(String file)

* + Create variable for csv writer
  + Loop through tagname, value
  + Write to rows, columns
  + Return csv file with exif data

get\_gps()

* + Check if ‘GPSInfo’ exists in dictionary
  + Get keys, values of dictionary
  + Convert to decimal form
  + Take north, east key data and convert to longitude/latitude
  + Return decimal degrees

## **Web.py**

* Determine max image file size
* Link static folder for images

image\_up(File file)

* + Check for appropriate file
    - Png or jpg allowed
  + Save filename to static folder
  + Open image with extract\_data method
  + Return html file, exif data for uploaded image

display\_image(File file)

* + Set dimensions in html file, assign filename
  + Return image file from folder
  + Display image

# **Known Issues:**

* Lack of database functionality is limiting what we can do.
  + Cannot create “users” without such. This makes editing exif data and returning files difficult or complicated.