# Photo File Upload Guide

## Introduction

This guide will walk you through the process of uploading photo files to the application. The application allows users to upload main photos along with their raw files. The uploaded files are stored in the data dictionary under project's root path, also stored a related record in the MongoDB database under the photo\_collection collection and can be accessed through the application.

## Uploading Photo Files

1. Open your web browser and navigate to the application's URL:

http://localhost:8888

1. Log in to the application using your credentials.
2. Once logged in, you will be redirected to the dashboard. Click on the "Upload" button to navigate to the file upload page.
3. On the file upload page, you will see a form with the following fields:

Main Photo: Click on the "Choose Main Photo" button to select the main photo file from your computer.

Raw Photo 1: Click on the "Choose Raw Photo 1" button to select the first raw photo file from your computer.

Raw Photo 2: Click on the "Choose Raw Photo 2" button to select the second raw photo file from your computer.

1. After selecting the files, you can enter the following optional information for each photo:

ISO

Aperture

Shutter

EV +/- (Exposure Value)

1. Once you have filled in the required fields, click on the "Upload" button to upload the files.
2. The application will display a success message along with the uploaded files' names.

## Notes

The application supports the following image file types: APNG, BMP, GIF, JPEG, PNG, SVG, TIFF, and WEBP.

The maximum file size for each image is 100MB.

The uploaded files are stored in the data dictionary under project's root path and also stored a related record in the MongoDB database under the photo\_collection collection.

## Troubleshooting

If you encounter any issues while uploading photo files, please check the following:

Make sure the selected files are in the supported image formats.

Ensure that the file size of each image does not exceed the maximum limit.

Verify that the MongoDB server is running and the application is properly connected to the database.

If the issue persists, please contact the application support team for further assistance.

# RESTful API Endpoints and Route Handlers

## Introduction

This document outlines the RESTful API endpoints and their corresponding route handlers about photo upload module. The file contains routes for uploading and retrieving files, and it utilizes MongoDB for storing file-related data.

## API Endpoints

1. **GET /fileUpload/**: This endpoint renders the file upload page. It requires the user to be authenticated.
2. **GET /fileUpload/:filename**: This endpoint retrieves a specific file based on the filename provided in the URL. The file is sent back to the client if it exists; otherwise, a 404 error is returned.
3. **POST /fileUpload/different**: This endpoint handles the upload of a main file along with two raw files. Each file is renamed and saved in a directory named after the user's login ID. The file-related data is then inserted into the MongoDB collection photo\_collection. The original filename is converted to UTF-8 and a unique filename is generated to avoid conflicts.

## Route Handlers

Each API endpoint is associated with a route handler that processes incoming requests and sends responses accordingly. The route handlers perform various tasks such as authentication, handling multipart/form-data requests...

The isAuthenticated middleware is used to ensure that certain routes can only be accessed by authenticated users.

The multer middleware is used for handling multipart/form-data, which is primarily used for uploading files.

## Example Usage

To use the API endpoints, you can make HTTP requests to the corresponding URLs using a tool like Postman or cURL. For example, to retrieve a specific file, you can make a GET request to /fileUpload/filename.jpg.

## Additional Notes

The application's configuration, including the database connection string and file upload settings, is defined in the config.js file.

The helper object contains utility functions that are used in the route handlers, such as generating unique filenames and normalizing file paths.

The fileUpload.ejs file is the view template that is rendered when the file upload page is requested.