2025 Academic Year
Computer Science

Clouding Computing & DevOps

Project: Images Gallery

Submit to: Sr Aung Naing Thu

Submit by:

Group1 -	Ngadawng Ja Seng Htoi	(3Cs-16)
	Jangmaw Sunday Awng	(3Cs-2)
	Labang Myu Ring Awng	(3Cs-3)
	Lahtaw Bawk Li	(3Cs-5)
	Mwihpu Nan Lung	(3Cs-11)
	Sham Sut Ring Naw	(3Cs-17)
	Wabaw Zau Ing	(3Cs-19)
	Zunwa Zau Ring	(3cs-21)

Contents

Objective

- Cloud storage – To be stored in the cloud without requiring dedicated memory.
- Storage saving – To save hard drive / Memory space.
- Backup – Automatically backup to prevent data loss.
- Recovery – To be recovery using another device, if the devices breaks.
- Security – Encrypted and securely stored on a cloud server.
- Collaboration – Users can see and work on the same file at the same time for better teamwork

Abstract

G1N8CSF Gallery Pro is a photo management web application designed to allow users to easily upload, organize, and view their photos. This application is built with a modern UI/UX design and allows photos to be displayed in either a Masonry (wall-style layout) or Grid view. It utilizes ImgBB cloud services. Users can filter photos by Date, Name, or Size, and can also check information such as the total number of photos, usage duration, and total data storage size.

I.System Architecture

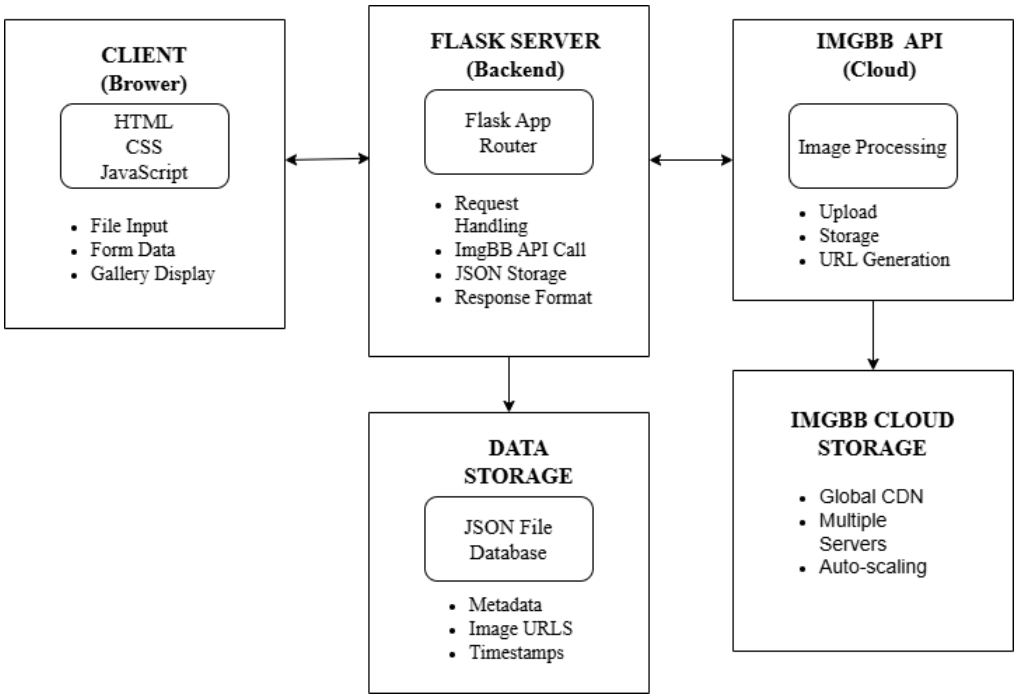

Figure1.1 Architecture Diagram

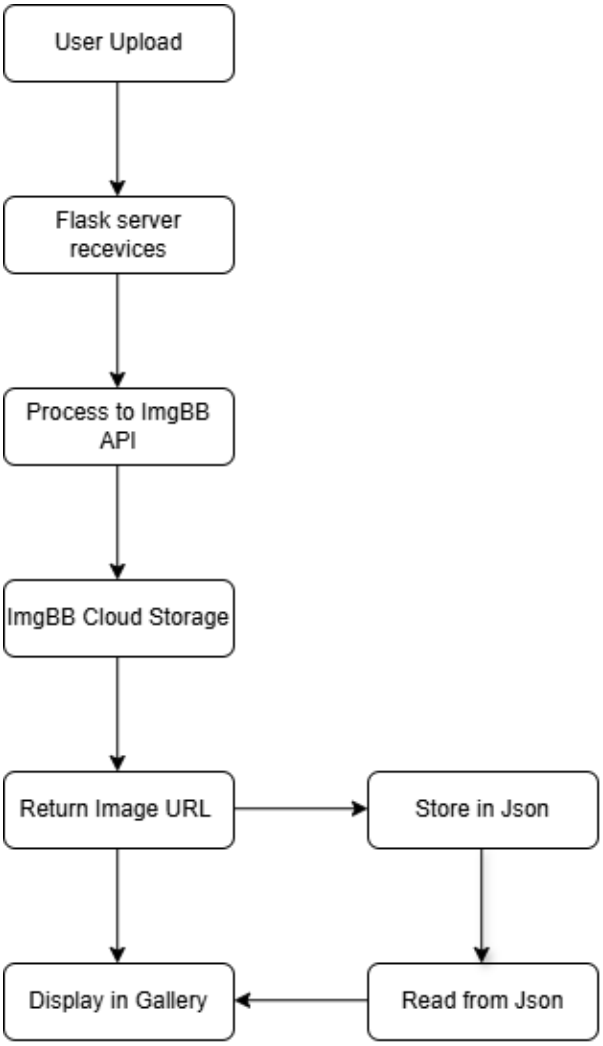

Figure1.2 Data Flow Diagram

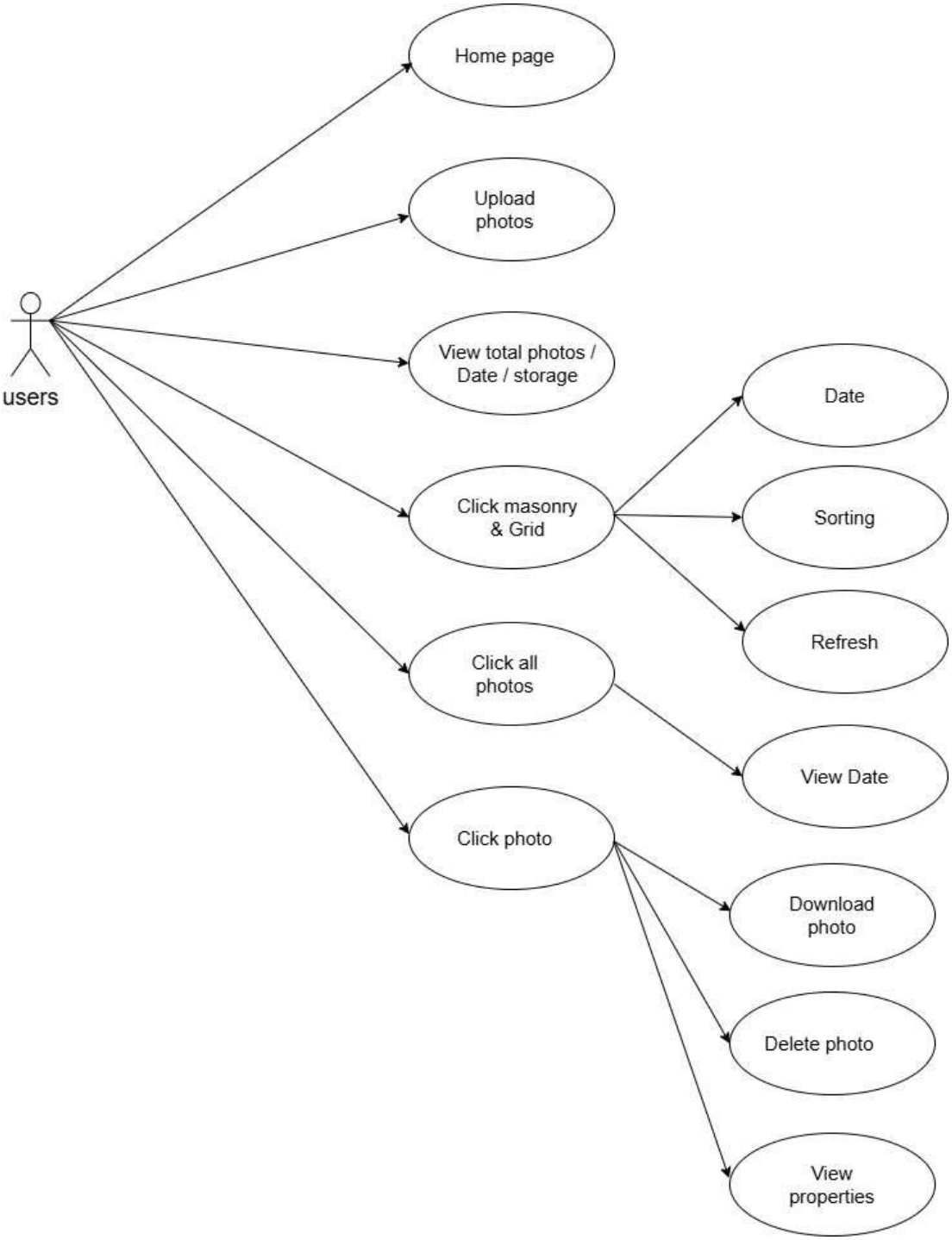

Figure1.3 Usecase Diagram

Figure1.4 User Interface Diagram

II. Design Justification: Images Gallery Web Application

1. Why imgBB Cloud for Image Storage?

- *No Server Hassle*: Instead of managing our own complex file storage and servers, we use imgBB's cloud. They handle all the difficult parts like storing the actual images, resizing them, and delivering them fast worldwide using a CDN.
- *Focus on Features*: This allows us to focus on building the gallery's features (like sorting by date, name, and size) instead of worrying about how to store and serve photos.
- *Saves Time and Money*: It's free to start and saves us from the cost and time of setting up our own image storage system.

2. Why Python Flask as the Main Framework?

- *Simple and Lightweight*: Flask is a minimal and flexible framework. It gives us just what we need to create web pages and handle user actions (like uploads and deletes) without unnecessary complexity.
- *Quick Development*: It let us build the application logic very quickly, connecting the front-end (what the user sees) to the back-end (the server) in a straightforward way.

3. Why These Specific Python Libraries?

- *requests*: This library is used to talk to the imgBB API. When you upload a photo, our Flask app uses requests to securely send that image to imgBB's servers.
- *Pillow (PIL):* This is an image processing library. We use it to read the photos you upload and extract important metadata – like the file size, dimensions, and type – which we then display in the gallery (e.g., "22.1 MB Total").
- *python-dotenv*: This is for security and configuration. It lets us store sensitive information like our imgBB API key in a separate, safe file that doesn't get shared with the public. This keeps our app secure.

III. Step-by-Step Product Selection Documentation

When deciding which cloud service to choose for a web app image gallery, several key points must be considered. When selecting a cloud service, the main factors to evaluate are cost, performance, security, and scalability. Among these, the most important one depends on your project's priority — for example, startups may prioritize cost, while large-scale systems may focus more on performance and security.

When choosing a cloud storage provider such as AWS S3, Google Cloud Storage, or Azure Blob Storage, it's important to compare their strengths and weaknesses. For data redundancy and backup support, AWS provides strong and reliable options. For a web app image gallery with growing user traffic, auto-scaling features are also very important.In terms of APIs and integration, AWS offers powerful and flexible options that are easy to use with other cloud services.

Security is also a major factor. When selecting a cloud service, it is important to check for features such as encryption, access control, and compliance certifications. Regional availability (data storage region) should also be considered because storing data in a closer region improves access speed and helps meet data protection requirements.

If you have already tested the free trials of multiple cloud providers, you should compare and evaluate the results in terms of cost, performance, and ease of use before making your final decision.

There are many cloud storages services available — AWS S3, Google Cloud Storage, Azure Blob Storage, and ImgBB are among the most common. AWS S3 is highly efficient for advanced users and easy to set up. However, it requires a credit card even for free-tier usage. Google Cloud Service is strong in AI/ML, Data Analytics, and Kubernetes, but it also has some weaknesses such as:

- High pricing
- Limited enterprise support
- Regional restrictions
- Management complexity

Although it is cost-effective for startups and SMBs, it may not compete as strongly at the enterprise level. Azure Blob Storage tends to have higher costs, is more complex to manage, and has some performance limitations with less mature enterprise features. ImgBB is more suitable for small-scale projects like an image gallery because it is simple and easy to use. Therefore, the ImgBB cloud service was chosen. There are several reasons for this choice, mainly due to its simplicity, ease of use, and suitability for smaller-scale image hosting projects -

- · Cost-Effectiveness: imgBB provides sufficient storage space and bandwidth in its free tier. For students' project, it offers a high-quality service without any financial cost.
- · Simplified Architecture: There is no need to maintain a personal file storage server. Only the image path needs to be stored in the database, eliminating the need to write complex file handling logic.
- · Built-in CDN: imgBB includes a CDN, enabling fast image loading. This ensures good performance for users from different geographic location. Easy Integration: It offers a simple REST API for uploading, and the documentation is clear. There is also an option for direct uploads from the frontend.
- · Scalability: If the application grows, upgrading to imgBB's paid plan is possible without needing to modify the application code.

IV. Scalability Design

In the Gallery Pro application, as the number of users increases, Load Balancers will be implemented to distribute traffic across multiple servers, preventing system slowdowns. Through the use of a Content Delivery Network (CDN), images will be served from the nearest server. Additionally, image compression and Lazy Loading systems will be integrated to enhance the application's performance.

To protect against system failures, an Automated Daily Backup system will be implemented, automatically backing up data every 24 hours and storing it in different geographic locations. With the Point-in-Time Recovery system, data from any specific time period can be restored. For application updates, the Blue-Green Deployment strategy will be used, allowing updates to be performed without service disruption.

The application will be designed with Multiple Server Instances to ensure redundancy, enabling 24/7 uninterrupted usage. A Real-time Health Monitoring system will continuously monitor the status of the servers 24 hours a day and perform automatic recovery when needed.

V. Implementation Roadmap

Project over view

A simple, scalable, resilient web application for uploading and viewing images, built using modern cloud native technologies

5-Day Implementation Roadmap: Images Gallery Web App

Technology Stack:

· Backend: Python Flask

· Frontend: HTML, CSS, JavaScript, JSON

· Database/Storage: imgBB API (for image storage and metadata)

· Cloud Service: imgBB Cloud Services

Day 1: Project Setup & Basic Flask Structure

Key Activities:

· Set up Python virtual environment and install Flask

· Create Flask application structure

· Set up project folders (static, templates)

· Create HTML layout and CSS styling

· Integrate imgBB API credentials into configuration

Deliverables:

· Working Flask development environment

· Basic web page accessible at http://localhost:5000

· Project structure ready for feature development

Day 2: Image Upload & imgBB Integration

Key Activities:

· Implement image upload form in HTML

· Create Flask route to handle file uploads

· Integrate imgBB API for image upload functionality

· Handle upload responses and store image metadata

· Display success/error messages to users

Deliverables:

· Functional image upload system

· Images successfully stored in imgBB cloud

· Basic user feedback system

Day 3: Gallery View & Image Display

Key Activities:

· Create gallery display layout with CSS Grid/Masonry

· Implement Flask route to fetch and display images

· Parse imgBB API response to get image URLs and metadata

· Display images in responsive gallery grid, masonry

· Show image information (image size, date)

Deliverables:

· Functional gallery view showing uploaded images

· Responsive design that works on different screen sizes

· Basic metadata display

Day 4: Advanced Features & Sorting

Key Activities:

· Implement sorting functionality (by date, name, size)

· Add JavaScript for client-side sorting

· Create download/save functionality for images

· Implement delete functionality using imgBB API

· Enhance UI with loading indicators and better styling

Deliverables:

· Full CRUD operations (Create, Read, Update, Delete)

· Sorting features working

· Download functionality implemented

· Improved user interface

Day 5: Testing, Polish & Deployment

Key Activities:

· Comprehensive testing of all features

· Fix bugs and improve error handling

· Optimize performance and loading times

· Final UI/UX polish and responsiveness check

· Deploy to production server

· Document the application and setup process

Deliverables:

· Fully functional and tested application

· Deployed application accessible via public URL

· Project documentation

· Bug-free user experience

Daily Time Allocation (Approximate):

· Day 1: 6-8 hours (Setup and foundation)

· Day 2: 6-7 hours (Core upload functionality)

· Day 3: 5-6 hours (Gallery and display)

· Day 4: 6-7 hours (Advanced features)

· Day 5: 6-8 hours (Testing and deployment)