Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Proposal
on
"ImGrep : A Smart Image Search Engine"

[Code No: COMP 207]

(For partial fulfillment of II Year/ II Semester in Computer Engineering)

Submitted by

Bijan Bakabal Thapa (Roll No. 7)
Manish Bhattarai (Roll No. 11)
Sadit Rasaili (Roll No. 40)
Sameep Kakshapati Shrestha (Roll No. 20)
Slok Pradhan (Roll No. 38)

Submitted to

Project Coordinator

Department of Computer Science and Engineering

Submission Date: 09/05/2025

Abstract

The explosion of digital imagery has created a pressing need for more intelligent and intuitive ways to organize, search, and interact with visual content. While existing platforms offer basic keyword or metadata-based search, they often fall short in understanding the deeper context and semantics of an image. *ImGrep* addresses this gap by providing a smart, mobile-first image search engine that leverages AI and computer vision to deliver rich, context-aware results.

The project will be developed using a modern tech stack, including Flutter for the mobile frontend, and Flask, PostgreSQL, Docker, and OAuth for the secure, scalable backend. Core technologies like OpenAI CLIP and natural language processing will enable features such as semantic image search, object detection, facial recognition, OCR, and AI-generated image descriptions. Users will also benefit from voice assistant integration, image editing tools, and timeline-based exploration features.

Expected outcomes include a fully functional mobile application capable of advanced image retrieval, AI-enhanced interactions, and intuitive search through both voice and text. By combining intelligent search, user-centric utilities, and modern UI features, *ImGrep* aims to redefine how people manage and explore visual content in the digital age.

We recommend *ImGrep* for users seeking a powerful, AI-driven solution for managing and retrieving image data in a more meaningful and accessible way.

Keywords: Image search, computer vision, semantic retrieval, voice assistant, OCR, Flutter, Flask, OpenAI CLIP

Table of Contents

List of Figures

Abbreviation

i. AI: Artificial Intelligence
ii. API: Application Programming Interface
iii. CLIP: Contrastive Language-Image Pretraining
iv. HDD: Hard Disk Drive
v. NLP: Natural Language Processing
vi. OAuth: Open Authorization
vii. OCR: Optical Character Reader
viii. OS: Operating System
ix. RAM: Random Access Memory
x. SSD: Solid State Drive
xi. SQL: Sequential Query Language
xii. UI: User Interface
xiii. SDK: Software Development Kit

Chapter 1 Introduction

1.1. Background

The proliferation of digital content, particularly images, has transformed the way individuals and organizations archive, search, and utilize visual media. With the increasing use of smartphones, social platforms, and digital photography, image data has grown exponentially across both personal and professional domains. However, despite the abundance of images, the tools available for intelligent and meaningful image retrieval remain limited in scope. Conventional image search engines largely depend on metadata and keyword tagging, which often fail to capture the semantic richness and contextual meaning embedded within images.

Current mainstream platforms like Google Images or social media tools offer limited depth in personalized image organization, object recognition, or context-aware retrieval. These solutions tend to overlook the potential of combining natural language processing (NLP), and image recognition technologies to power smarter, more intuitive image searches. As users demand more than simple filename or tag-based filtering, there is a growing need for systems that can interpret visual content, understand queries in natural language, and offer semantically relevant results.

To address these challenges, we propose ImGrep : an intelligent, feature-rich image search engine tailored to the next generation of visual data interaction. Leveraging technologies like OpenAI CLIP, OCR, facial recognition, and semantic search algorithms, ImGrep aims to redefine how users retrieve, analyze, and manage images across a variety of contexts.

1.2. Objectives

The primary objectives of the *ImGrep* project are as follows:

i. Develop a mobile-based intelligent image search engine using Flutter for a seamless user experience.
ii. Implement multi-modal search capabilities including metadata, text captions, object recognition, and semantic similarity.
iii. Integrate AI-driven enhancements such as image description generation, OCR (optical character recognition), and facial recognition.
iv. Provide utility tools like image editing (resize, coloration), compression, and grouping of similar images.
v. Incorporate a voice assistant to allow users to access and control features through natural language commands for improved accessibility and convenience.
vi. Ensure the backend is scalable and secure using Flask, PostgreSQL, Docker, and OAuth-based authentication.
vii. Enable cloud-based storage and optional timeline-based discovery

1.3. Motivation and Significance

In today's data-rich world, visual content plays a pivotal role in storytelling, documentation, research, and daily communication. Yet, existing solutions for managing and retrieving images fall short of user expectations, especially in terms of context-aware search and intuitive AI assistance. Whether for professional photographers looking to organize their archives, researchers needing semantic filtering, or everyday users trying to find a specific memory from thousands of photos, current tools offer limited flexibility and intelligence.

The motivation behind ImGrep is to harness state-of-the-art AI to provide users with a powerful yet user-friendly platform that not only searches by filename or date but

understands the meaning behind a query. For example, users could search for "sunset with mountains" or "documents containing handwritten notes" and receive results that match both the visual content and the implied semantics.

What distinguishes ImGrep from other image search systems is its holistic approach blending natural language understanding, computer vision, and intelligent retrieval into a mobile-first experience. By doing so, it meets the growing demand for smarter digital asset management, bridging the gap between raw visual data and human interpretation.

1.4. Expected outcomes

The successful completion of the ImGrep project will deliver the following key outcomes:

i. **Comprehensive Mobile Image Search Engine**: A fully functional Flutter-based application capable of ingesting and retrieving images.
ii. **Advanced AI Features**: Integration of CLIP-based semantic search, object detection, facial recognition, OCR, and AI-generated image captions.
iii. **User-Centric Utilities**: Support for basic image editing, compression tools, grouping of similar images, and a timeline-based archive feature.
iv. **Robust Backend Architecture**: A Flask-powered backend with PostgreSQL for scalable data storage, containerized via Docker, and secured with OAuth-based user authentication.

These outcomes will establish ImGrep as a modern, intelligent, and versatile platform for visual data management empowering users to interact with images in ways that go far beyond traditional keyword search.

.

Chapter 2 Related Works

In exploring the development of advanced image search platforms, it is essential to examine existing systems and technologies to identify the gaps that our project seeks to address. This section highlights relevant projects and tools, assessing their capabilities and limitations in comparison to our proposed image search engine.

2.1. Google Photos

Google Photos is a widely used image management platform that combines cloud storage with AI-powered features. It offers functionalities such as automatic image organization, facial recognition, object detection, and metadata-based search (e.g., date, location). One of its notable strengths is the ability to search using natural language queries like “photos of dogs at the beach.” However, Google Photos operates as a closed system, with limited options for user customization, extensibility, or integrating third-party tools. In contrast, our image search engine aims to offer a more modular, customizable platform that emphasizes semantic understanding, natural language processing, and advanced search capabilities beyond simple metadata filtering.

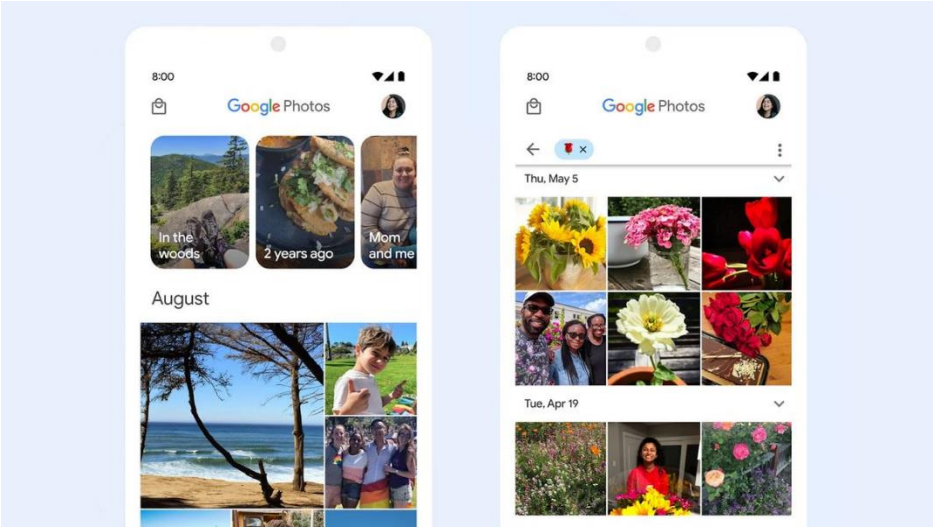

Figure 1 Google Photos UI

2.2. Pinterest Lens

Pinterest Lens is a visual discovery tool that allows users to search for ideas and products simply by pointing their camera at an object. It uses computer vision to recognize objects and return visually similar pins, making it a powerful tool for inspiration and shopping. However, Pinterest Lens is primarily consumer-focused and tightly integrated into the Pinterest ecosystem, limiting its utility for developers or broader applications. Our project aims to address this gap by offering a developer-friendly image search engine that incorporates similar visual recognition capabilities such as object detection and product matching while also supporting advanced features like semantic search and customizable image filtering.

Figure 2 : Pinterest Image Search UI

2.3. Samsung Bixby Vision

Samsung's Bixby Vision is an AI-powered feature integrated into Samsung devices that enables users to search for information by pointing their camera at real-world objects. It can identify text, translate languages, recognize landmarks, shop for similar products, and scan QR codes. While Bixby Vision offers a broad set of features, it is tightly bound to Samsung's hardware ecosystem and lacks open developer access or extensibility for third-party platforms. Our project addresses this limitation by offering a device-agnostic, developer-friendly image search engine that not only supports visual recognition but also introduces advanced capabilities like semantic search, real-time filters, and interactive image organization.

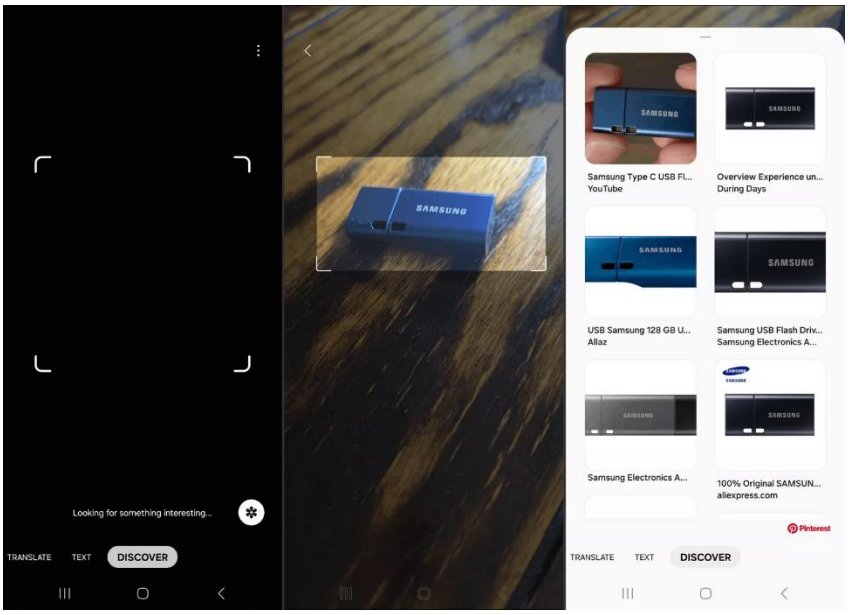

Figure 3: Samsung Bixby Vision UI

Chapter 3 Procedures and Methods

Developing the Image Search Application process will be systematic and incremental in order to end up with a productive, user-friendly, and reliable gallery image search platform utilizing natural language query. We shall adopt modern methods of development, including Agile practice, in order to achieve flexibility, rapid prototyping, and ongoing enhancement through the course of the project timeline.

3.1 Development Phases

i. Requirement Analysis:

The first phase will be spent gathering and analyzing functional and non-functional requirements of the application. This will include identifying user needs through interviews and surveys to determine common patterns of search, privacy concerns, and usability expectations. Current AI-based search systems will be analyzed to create efficient search and indexing mechanisms. This is where the scope of the project will be established, delineating key features such as prompt-based search, image tagging, and integration with the device's gallery.

ii. System Design:

At this phase, we will design wireframes and UI mockups to represent user interaction flows. Special attention will be paid to designing an intuitive interface that simplifies image search and display functionality. Backend architecture will be designed to handle image indexing, OpenAI CLIP-based embedding generation, and secure user authentication via OAuth.

Key design aspects will be:

a. Successful synchronization of device gallery photos to the app's local or cloud storage.
b. Backend services for image embedding processing and storage.
c. OAuth-based secure and frictionless user login integration.

d. Responsive web and mobile-friendly UI to provide ease of access and navigation.

iii. Implementation

The implementation phase will involve writing the application according to the technology stack that has been selected:

a. Frontend:

The frontend will be built with Flutter, which will enable us to build a fast, responsive, and visually pleasing mobile app for both Android and iOS devices. Flutter's cross-platform nature will enable us to have a seamless user experience across devices, with access to native performance and features. The user interface will be simple and intuitive, with the ability to easily enter search queries, view image results, and maintain gallery sync in a seamless mobile environment.

b. Backend:

Backend will be coded using Flask for the generation of REST APIs and server-side operations. OpenAI CLIP integration will allow creation and storage of image embeddings for semantic search. OAuth will be utilized for authentication and authorization of users securely. PostgreSQL will serve as the primary database for the storage of image metadata and embeddings, while Docker will be employed for containerizing the backend services for deployment that is scalable and consistent.

iv. Testing

A comprehensive testing plan will be employed in order to confirm the reliability, performance, and security of the application. This will include:

a. Unit Testing: to confirm individual components and functions are tested.

b. Integration Testing: to confirm seamless interaction among frontend, backend, and database layers.
c. User Acceptance Testing (UAT): with actual users to test application in real-time usage and provide feedback on usability and precision of results.

Automated and manual test cycles will be executed during development to detect and correct issues early.

v. Documentation

System architecture, API specs, deployment steps, and user manuals will be documented in detail while developing the system. Technical documentation will be aimed at supporting future maintenance and development. A final project report will summarize the objectives, methodology, issues, and outcome of the development process.

Chapter 4 System Requirement Specifications

4.1. Software Requirements

i. Programming Languages: Python 3.8+, Dart (Flutter SDK)
ii. Backend Framework: Flask (Python)
iii. Frontend Framework: Flutter (Dart)
iv. Database: PostgreSQL 12 or higher
v. Authentication: OAuth 2.0 via Flask-OAuthlib
vi. Image Processing: OpenCV or Pillow
vii. Image Embedding & Search:
    a. OpenAI CLIP (Contrastive Language–Image Pretraining) for semantic image embeddings
    b. TensorFlow/Keras or PyTorch models
viii. ORM: SQLAlchemy
ix. Storage:
    a. File system for storing user-uploaded images and profile picture
    b. Flask static routes or signed URLs for image retrieval
x. Containerization: Docker and Docker Compose
xi. Other Tools: Git (version control), Postman (API testing)

4.2. Hardware Requirements

i. Development Machine Requirements
    a. Processor: Intel Core i3 or higher
    b. Memory (RAM): 4 GB or higher
    c. Storage: Solid-State Drive (SSD) / Hard Disk Drive (HDD) with at least 10 GB of available space; more may be required depending on dataset size
    d. Graphics Processing Unit (GPU): Recommended for running OpenAI CLIP and accelerating image embedding tasks

ii. Deployment Server:
    a. Environment: Docker-compatible host (e.g., Linux-based VPS, cloud container service)
    b. Storage: Adequate disk space to store uploaded images, extracted embeddings, and user profile pictures; scalable storage (e.g., mounted volumes) is recommended for production environments
    c. Networking: Stable internet connection with support for REST API access and file transfers
    d. GPU Support: For high-throughput inference using CLIP.

4.3 System Requirements

i. Supported Platforms: Linux, Windows, macOS; Android/iOS (mobile app)
ii. Dependencies:
    a. Docker and Docker Compose
    b. Python virtual environment tools
    c. Flutter SDK and mobile testing environment
    d. PyTorch with OpenAI CLIP model
    e. Image file handling libraries (Flask-Uploads)

4.4 Functional Requirements

i. OAuth-based user authentication (Google, GitHub, etc.)
ii. Upload, store, and index user-submitted images
iii. Store and serve user profile images securely
iv. Extract semantic embeddings from images using CLIP
v. Store image features and metadata in PostgreSQL
vi. Retrieve and display visually similar images using content-based search
vii. RESTful API for:
    a. User registration and login
    b. Uploading and retrieving user profile images
    c. Image upload and similarity search

d. Fetching image metadata and search results

4.5 Non-Functional Requirements

i. Fast and scalable image search with semantic similarity
ii. Secure and private storage of user data and images
iii. Efficient serving of static images (profile and uploaded images)
iv. Responsive UI for both image upload and result viewing
v. Modular, containerized architecture using Docker
vi. Maintainable codebase and extensible backend

Chapter 5 Project Planning and Scheduling

5.1. Tasks

The Gantt chart provided in Figure 4 outlines the timeline for various tasks involved in project development, giving a clear picture of time allocation across different phases. This chart allows for structured planning by visually mapping out each task's start and end dates, contributing to an organized project schedule.

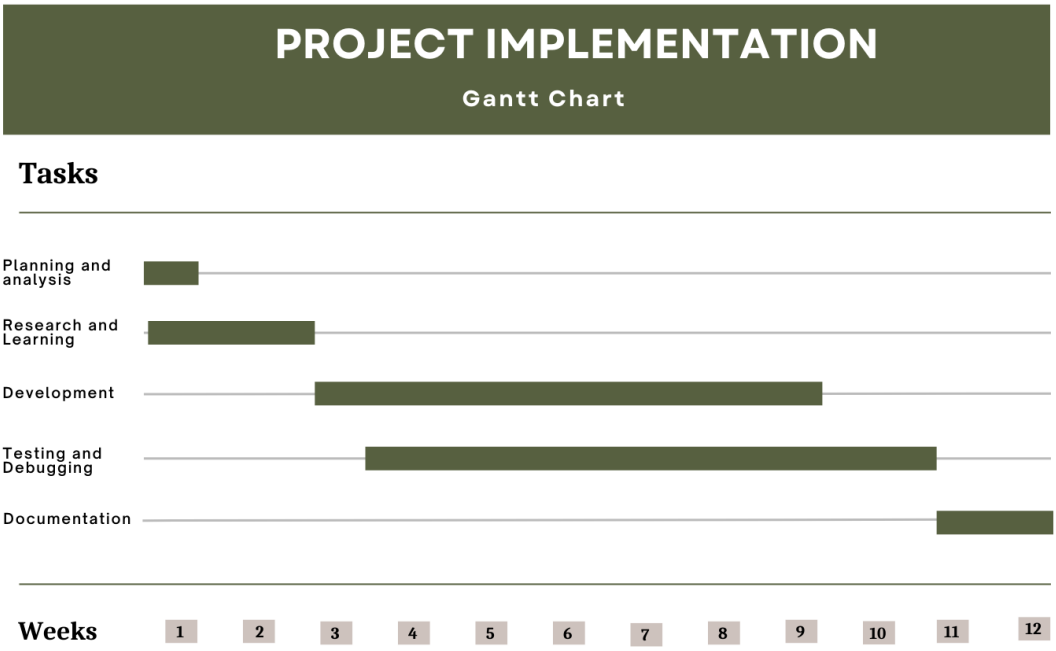

Figure 4: Gantt Chart

Tasks and Phases

i. **Planning and Analysis:** Initial stage focused on problem identification, requirement gathering, and setting project objectives. This phase is scheduled for the first week to ensure a clear project direction.

ii. **Research and Learning:** Occurs early in the project, spanning weeks 2 to 3, to gather necessary insights and knowledge related to the project scope.

iii. **Development:** The core phase of the project, stretching from week 4 to week 8, where the actual work on the project deliverables takes place.

iv. **Testing and Debugging:** Starts in week 5 and continue alongside development, extending to week 11 to identify and resolve any issues in the project.

v. **Documentation:** Final stage, occurring in week 12, to compile all findings, processes, and outcomes, ensuring a complete and accurate project record.

The Gantt chart structure provides a roadmap that helps monitor project progress and allocate resources efficiently. With a breakdown of terminal and summary elements, this visual tool aids in meeting project milestones and managing time effectively.

References

Authlib. (n.d.). *Authlib documentation*. https://docs.authlib.org/

Celery Project. (n.d.). *Celery documentation*. https://docs.celeryq.dev/

Docker Inc. (n.d.). *Docker documentation*. https://docs.docker.com/

Docker Inc. (n.d.). *Docker Compose documentation*. https://docs.docker.com/compose/

Flask Documentation. (n.d.). *Flask web development documentation*. Pallets Projects. https://flask.palletsprojects.com/

Flutter. (n.d.). *Flutter documentation*. https://docs.flutter.dev/

OpenAI. (2021). *CLIP: Connecting text and images*. https://github.com/openai/CLIP

OpenCV. (n.d.). *OpenCV-Python tutorials*. https://docs.opencv.org/

Pillow Contributors. (n.d.). *Pillow (PIL Fork) documentation*. https://pillow.readthedocs.io/

PostgreSQL Global Development Group. (n.d.). *PostgreSQL documentation*. https://www.postgresql.org/docs/

PyTorch. (n.d.). *PyTorch documentation*. https://pytorch.org/

SQLAlchemy. (n.d.). *SQLAlchemy documentation*. https://docs.sqlalchemy.org/

TensorFlow. (n.d.). *TensorFlow Hub*. https://www.tensorflow.org/hub

Towards Data Science. (2019, September 14). *Image similarity using deep learning*. https://towardsdatascience.com/image-similarity-using-deep-learning-8d7482c4d26

oauth.net. (n.d.). *OAuth 2.0 — OAuth*. https://oauth.net/2/

pub.dev. (n.d.). oauth2_client | *Flutter package*.
https://pub.dev/packages/oauth2_client