A Data Science/Machine Learning Project Proposal

on

Information Retrieval System for Unstructured Image Data

Submitted in Partial Fulfillment of the Requirements for

the Degree of Bachelor in Software Engineering

under Pokhara University

Submitted by:

Alish Wosti, 231705

Nikhil Khatri, 231721

Roshan Bhatta, 231727

Date:

Dec 15, 2025

Department of Software Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

1. INTRODUCTION

In the modern digital era, people use smartphones and computers every day to save and share information. One very common way of saving information is by taking screenshots. Users take screenshots of many things such as online payment receipts, error messages while coding, chat conversations, social media posts, lecture slides, and important notes. Over time, a person's phone gallery can contain thousands of screenshots.

Although storing screenshots is easy because digital storage is cheap, finding a specific screenshot later becomes very difficult. Most mobile phones and operating systems can only search images by date or file name. They cannot search the actual text written inside an image. Because of this, users often scroll for a long time and still fail to find the screenshot they need. As a result, the screenshot gallery turns into a "digital graveyard," where useful information is stored but hard to retrieve.

To solve this problem, this project proposes a **Multi-Modal Retrieval System**. This system is designed to organize and search screenshot data in a smarter way. It uses **Optical Character Recognition (OCR)** to read and extract text from screenshots. Once the text is extracted, **Sentence-BERT** is used to understand the meaning of the content, add useful tags, and store the information properly.

With this system, users can search their screenshots using simple natural language queries. For example, a user can type *"Show me the receipt for the sushi dinner last month"* or *"Find the coding error screenshot from yesterday."* The system will then search through the extracted text and show the most relevant screenshots. This makes screenshot management easier, saves time, and helps users quickly access important information when they need it.

2. PROBLEM STATEMENT

2.1 Retrieval Bottleneck

Even though mobile phones today have very large storage capacity, the way users search for information inside their image galleries has not improved much. People keep collecting screenshots over time, and many of these screenshots contain important information such as online payment details, study notes, class materials, and chat records. However, most gallery applications only treat these screenshots as normal images made of pixels. They do not understand or recognize the text and meaning inside the images.

Because of this limitation, useful information remains "hidden" even though it is clearly visible in the screenshots. When users need to find a specific image, they are forced to manually scroll through hundreds or even thousands of screenshots. This process is slow, frustrating, and wastes a lot of time. As a result, users become less efficient and may even fail to find important information when they need it most.

2.2 Lack of Semantic and Contextual Understanding

Current search systems in operating systems mainly depend on **metadata**, such as file names, dates, or locations, instead of the actual content inside the image. This means the system does not truly understand what the screenshot contains. If the file name or date is unknown, finding the correct image becomes very difficult.

Another major problem is the **keyword limitation**. Most existing tools work only with exact word matching. For example, if a user searches for the word *"Food,"* the system may not show a receipt that contains words like *"Momo"* or *"Pizza."* This happens because the system does not understand that these items are types of food. It lacks the ability to understand meaning and relationships between words.

There is also a **visual–textual gap** in current tools. Operating systems are not good at connecting visual information with text. For example, they cannot easily tell whether an image is a graph, a meme, or a document. Because of this, search results are often inaccurate or irrelevant. This makes searching for screenshots inefficient and frustrating for users.

3. PROJECT OBJECTIVES

1. **To develop** a Multi-Modal Semantic Information Retrieval System that allows users to index, organize, and retrieve screenshot data using natural language queries, transforming static images into a searchable knowledge base.

2. **To enable** semantic search by understanding the meaning of text and linking related concepts, such as connecting the query *"Health expenses"* with receipts containing words like *"Medicine."*

3. **To develop** a hybrid search mechanism that combines semantic understanding with metadata such as date and time, allowing queries like *"screenshots from last night"* or *"chat images from 2024."*

4. **To design** a user-friendly web-based interface that displays search results clearly, shows confidence scores, and highlights relevant text regions within the original screenshots.

3. SCOPE

3.1 Functional Scope

- **Data Ingestion**
  - The system will accept static image formats such as **JPG, PNG, and JPEG**.
  - It will mainly focus on **screenshots**, including:
    - Digital payment receipts
    - Social media conversations
    - Code error screenshots and snippets
    - Lecture slides and notes
- **English Text Processing**
  - The system is designed to work **only with the English language**.
  - OCR models optimized for **Latin scripts** will be used.
  - Embedding models trained on **English text data** will ensure accurate text understanding.
- **Semantic Search**
  - The system will support **meaning-based (semantic) search** instead of simple keyword search.
  - Users can search using conceptual queries such as:
    - "Travel expenses" to find receipts like **Uber rides or flight tickets**.
- **Temporal Filtering**
  - The system will extract image metadata such as **creation date**.
  - Users can filter results based on time, for example:
    - "Last week"
    - "2024"

3.2 Technical Scope

- **Programming Language**
  - The entire project will be developed using **Python**.
- **Optical Character Recognition (OCR)**

        - OCR will be implemented using:
            - **Tesseract OCR** (English configuration), or
            - **PaddleOCR** (English configuration).
            - Or Deepseek OCR
- **Vector Database**
    - A local vector database will be used for efficient similarity search.
    - Possible options include:
        - **ChromaDB**
        - **FAISS**
- **User Interface**
    - A **web-based dashboard** will be developed using **Streamlit**.
    - The interface will allow users to:
        - Upload images
        - Enter search queries
        - View and manage search results

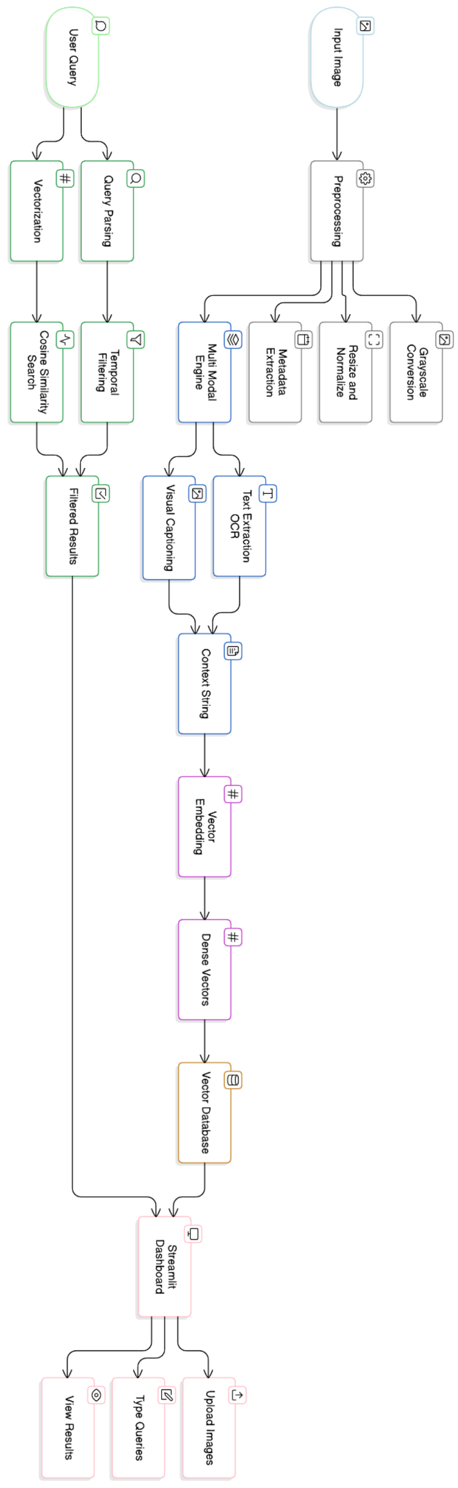
4. LITERATURE REVIEW

The system is based on CLIP proposed by Radford *et al.* [1], which enables joint understanding of images and text. Although CLIP performs well for general visual concepts such as common objects, it struggles to accurately recognize fine-grained textual information in screenshots, including receipts and system error messages. To address this limitation, PP-OCR developed by Du *et al.* [2] is employed to extract textual content from images. PP-OCR is a lightweight and efficient optical character recognition system, making it suitable for deployment on standard computing devices without relying on cloud-based processing. Furthermore, for semantic retrieval of screenshots, Sentence-BERT introduced by Reimers and Gurevych [3] is utilized to convert the extracted text into dense vector representations. This enables meaning-based search rather than exact keyword matching, thereby improving the overall effectiveness and usability of the system.

5. METHODOLOGY

The system is designed to **search screenshots and images** using text and visual information.

It works in **steps or phases** to make unstructured images searchable.



5.1 System Architecture

- The system has **three main layers**:
    - **Processing Layer**: Reads text (OCR) and describes images (Visual Captioning).
    - **Knowledge Layer**: Stores processed data as vectors in a database.
    - **Application Layer**: Lets users search and see results.

5.2 Phase 1: Data Collection & Preprocessing

- **Data Collection**: Accepts **JPG and PNG images** through a web page.
- **Preprocessing** (using OpenCV):
    - Convert images to **grayscale** to help OCR.
    - **Resize / normalize** images for consistency.
    - Extract **creation date** for time-based search.

5.3 Phase 2: Feature Extraction (Multi-Modal Engine)

- **Text Extraction (OCR)**:
    - Read text in the image using **PaddleOCR or Other**.
    - Extracts receipts, chat messages, error codes, etc.
- **Visual Context (BLIP)**:
    - Describes non-text parts of the image in natural language.
    - Example: "A screenshot of a line graph showing growth."
- **Combine Data**:
    - Merge OCR text and visual description into a single **Context String**.

5.4 Phase 3: Vector Embedding & Indexing

- **Convert text to vectors** using **Sentence-BERT** (384-dimensional).
- Similar words/concepts (e.g., "Supper" and "Dinner") are close in vector space.
- **Store vectors** in the ChromaDB database with metadata like file path and date.
- Use the HNSW **algorithm** for fast search.

5.5 Phase 4: Query Processing & Search

- **Hybrid search** process:
    1. **Parse query for date** (e.g., "last night") using `dateparser`.
    2. **Convert query text to vector** with Sentence-BERT.
    3. **Compare vectors** to stored image vectors using **cosine similarity**.
    4. **Filter results** by date or other constraints.
- Example query: *"Show me the sushi bill from last night."*

5.6 Phase 5: User Interface & Visualization

- Web dashboard made with **Streamlit**.
- Features:
    - Upload multiple screenshots at once.
    - Enter natural language search queries.
    - View results with:
        - Image preview
        - Extracted text
        - Confidence scores

6. TOOLS AND TECHNOLOGY USED

The system uses tools chosen for **performance, open-source availability, and local execution**.

6.1 Programming Language & Environment

- **Python (v3.9+)**: Main programming language; popular for AI, Computer Vision, and NLP.
- **IDE (VS Code / Jupyter/colab)**: Used for writing, testing, and debugging code.

2. Computer Vision & OCR (The "Eyes")

- **OpenCV (cv2)**:
    - Preprocesses images by converting them to **grayscale**, removing noise, and highlighting text.
- **PaddleOCR**:
    - Detects and reads English text in screenshots.
    - Uses a **lightweight neural network (PP-OCRv3)** for high accuracy on screen data.
    - Preferred over older tools like Tesseract.

3. Natural Language Processing (The "Brain")

- **Sentence-Transformers (SBERT)**:
    - Model used: **all-MiniLM-L6-v2**
    - Converts text from screenshots into **384-dimensional vectors** for semantic search.
    - Fast and accurate, works well even on CPU.
- **PyTorch**: Powers the deep learning models for OCR and embedding.

4. Database & Storage (The "Memory")

- **ChromaDB**:
    - Stores vectors instead of text.
    - Uses **HNSW algorithm** for fast similarity searches.
- **Pandas**:
    - Handles metadata like filenames, timestamps, and confidence scores.

5. User Interface (The "Face")

- **Streamlit**:
    - Web dashboard for uploading screenshots and typing queries.
    - Shows search results and bounding boxes quickly without complex web coding.
- **Dateparser**:
    - Converts natural language time queries (e.g., “last night”) into machine-readable

7.EXPECTED OUTPUT

This project addresses the problem of “Data Dark Matter”, where important information in screenshots is hard to find. By combining Optical Character Recognition (OCR) with semantic vector embeddings, the system allows users to search images intelligently using natural language. Unlike traditional keyword search, it understands the meaning behind queries, saving time and effort.

8.CONCLUSION

The methodology uses PaddleOCR to extract text from screenshots, Sentence-BERT to convert the text into vectors for semantic understanding, and ChromaDB to efficiently retrieve relevant results. This multi-modal approach ensures that users can quickly locate receipts, documents, or chat messages even without knowing the exact filenames or dates.

The system is designed to be flexible. While the current plan uses specific OCR models, embedding models, and databases, the technology stack can be updated if needed to improve performance, accuracy, or compatibility with local hardware. The main goal remains the same: accurate and fast semantic retrieval of information from unstructured screenshots.

REFERENCES

[1] A. Radford et al., “Learning Transferable Visual Models From Natural Language Supervision,” in Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR, vol. 139, pp. 8748–8763, 2021.

[2] Y. Du et al., “PP-OCR: A Practical Ultra Lightweight OCR System,” arXiv preprint arXiv:2009.09941, 2020.

[3] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992, 2019.