Project Documentation and Report

Project Title: Visual Search System using VLLMs

Team Name: Team Shamrocks

Problem Statement

In today's digital world, users expect intelligent search systems that understand natural language and visual inputs. Traditional image search engines rely heavily on metadata or tags, which limits their effectiveness in handling semantic queries or untagged datasets.

Problem: Design a system that enables users to perform a visual search using either text input or image upload, providing semantically relevant image results. If no local match is found, the system should seamlessly retrieve data from external APIs.

Solution

Our project introduces a Visual Search System powered by Vision-Language Large Models (VLLMs) like CLIP, integrated with pgvector and backed by a fallback to external APIs.

Key Features:
- Text-to-Image Search: Users enter natural language queries to fetch matching images.
- Image Upload & Match: Users can upload an image to find visually similar images.
- Local Search Using pgvector: Fast semantic similarity search using vector embeddings.
- External API Fallback: If no local match is found, it calls external APIs for results.
- Built with Flask, React, Vite.js, Tailwind CSS, and PostgreSQL.

Team & Mentor Details

Team Members:
- Vigneshwar S – Full Stack Developer
- GokulNath – Backend Developer
- Marishwaran – Frontend & UI/UX Designer

Mentor:
- S.H.Annie Silviya
  Designation: Assistant Professor
  Email: anniesilviya.sh@ritchennai.edu.in

Logic & Implementation Strategy

1. Text or Image input is taken from the user.
2. CLIP model converts input into embeddings.
3. Local dataset is searched using pgvector in PostgreSQL for similarity.
4. If no relevant match is found, the system triggers an external API to fetch images.
5. The response is returned to the frontend with the top most similar results.
6. The system ensures semantic relevance using Cosine Similarity on vector embeddings.

Code Description & Output

Backend (Flask):
- Endpoint /search handles both text and image queries.
- Vector embeddings are generated using the CLIP model.
- Uses pgvector for similarity search within the database.
- If no local match, it calls the Unsplash/Google/Bing image API.

Frontend (React + Vite.js):
- Users can either type a query or upload an image.
- Axios fetches results from Flask backend.
- Images are displayed in a responsive UI using Tailwind CSS.

Output:
- Relevant images fetched and displayed based on query or upload.
- Smooth fallback between local DB and external APIs.

Sample Demo Video

You can view the demo videos here:https://drive.google.com/file/d/10QXP-tAAf4WTF66EnV3CdLTr4lhobyGy/view?usp=sharing

Individual Contributions

Team Member Role & Contribution

Vigneshwar S - Built the backend logic with Flask, CLIP integration, PostgreSQL, and pgvector.

Gokulnath V - Developed the frontend UI using React and Tailwind CSS. Integrated backend APIs.

Marishwaran A - Handled image processing, embedding logic, and contributed to database management.