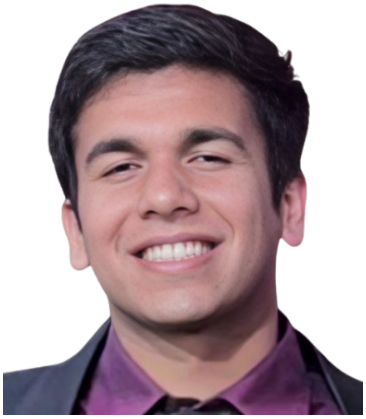
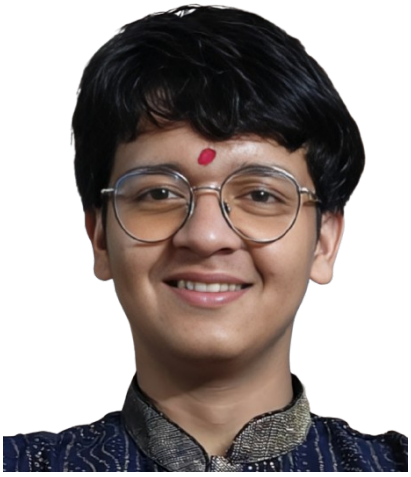
Graphic Era deemed to be University DEHRADUN

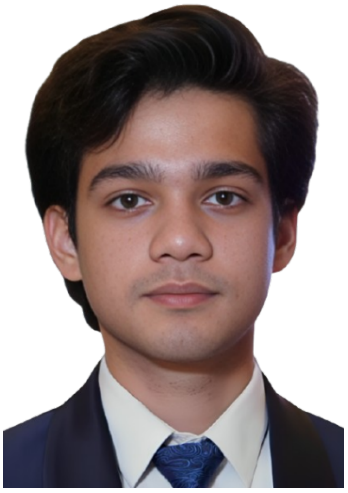
PROJECT AND TEAM INFORMATION

Project Title

VectorSearchDB: A Content-Based Image Search Engine in PostgreSQL

Student / Team Information

Team #DBMS-V-T028	Team Name: Mighty Bytey
Team member 1 (Team Lead)	Maini, Gunottam – 230211521 gunottammaini@gmail.com 
Team member 2	Dhyani, Dishant – 23021797 dihantdhyani01@gmail.com 

Team member 3	Gusain, Abhinav Singh – 23021580 abhinavgusain5108@gmail.com 
Team member 4	Rawat, Satyam Singh – 230211536 rawatsatyam058@gmail.com 

Tasks Completed

Task Completed	Team Member
1. Developed the Data Ingestion Pipeline	Gunottam Maini , Dishant Dhyani , Satyam Singh Rawat
2. Implemented AI-Powered Embedding Generation	Dishant Dhyani , Abhinav Singh Gusain , Gunottam Maini
3. Validated Initial Data Insertion	Satyam Singh Rawat , Abhinav Singh Gusain

Challenges/Roadblocks

The primary challenge lies in **configuring and integrating the pgvector extension** with PostgreSQL. While the setup was successful, ensuring compatibility with our Python-based embedding pipeline (using CLIP) required multiple iterations to establish a seamless data flow between the embedding generation script and the database.

Another challenge was **embedding consistency and dimensionality management**. Generating uniform vector embeddings from diverse images using the CLIP model demanded careful preprocessing and validation to avoid dimension mismatches or corrupted vectors during insertion.

We also faced **performance bottlenecks** while inserting large batches of image embeddings into the database. The insertion time and memory usage increased significantly with dataset size. To overcome this, we plan to implement **batch-wise insertion and indexing** strategies to maintain both efficiency and scalability.

Additionally, managing the **dataset preprocessing pipeline**—including resizing, encoding, and normalization of images—required optimization to ensure embedding quality without sacrificing speed.

We are addressing these challenges through systematic debugging, incremental testing after each component integration, and regular code profiling. Our upcoming steps will focus on optimizing data ingestion speed and preparing the database for Phase 3: **Search Implementation and Indexing**.

Tasks Pending

Task Pending	Team Member
Implementation of vector similarity search (image-to-image and text-to-image) in Flask backend	Gunottam Maini
Creation and optimization of HNSW vector index in PostgreSQL	Dishant Dhyani
Performance benchmarking and latency analysis using EXPLAIN ANALYZE	Satyam Singh Rawat
Frontend development for user interface (HTML/CSS) and result visualization	Abhinav Singh Gusain

Project Outcome/Deliverables

The *VectorSearchDB* project aims to deliver a unified content-based image search system built on PostgreSQL with pgvector integration. By the end of the project, the key outcome will be a **functional prototype** capable of performing **image-to-image** and **text-to-image similarity searches** directly within a relational database.

The main deliverables include:

1. **PostgreSQL Database with pgvector Extension** – Configured schema for storing image metadata and AI-generated embeddings.
2. **Data Ingestion Pipeline** – A Python-based script that processes image datasets, generates embeddings using the CLIP model, and populates the database efficiently.
3. **Backend Search Module** – Flask-based implementation enabling hybrid queries combining metadata filters and vector similarity.
4. **User Interface Prototype** – A simple web UI allowing users to upload an image or enter text queries for content-based retrieval.
5. **Performance Benchmark Report** – Evaluation of search latency and efficiency before and after vector indexing.
6. **Final Report & Presentation** – Comprehensive documentation of system design, implementation, and experimental results.

Collectively, these outcomes demonstrate how vector search can be seamlessly integrated into traditional databases, reducing the dual-system dependency of modern AI applications.

Progress Overview

As of the completion of **Phase 2**, the *VectorSearchDB* project is progressing well according to the planned timeline. The **Phase 1 objectives**, including environment setup, PostgreSQL and pgvector configuration, and database schema design, have been successfully completed on schedule. The database is fully functional, and the schema for storing image metadata and embeddings has been finalized.

Currently, the team is in **Phase 2**, focusing on developing the **data ingestion pipeline**. The embedding generation process using the CLIP model has been implemented, and initial tests for inserting vector data into PostgreSQL have shown positive results. Minor performance optimizations are underway to handle large datasets efficiently.

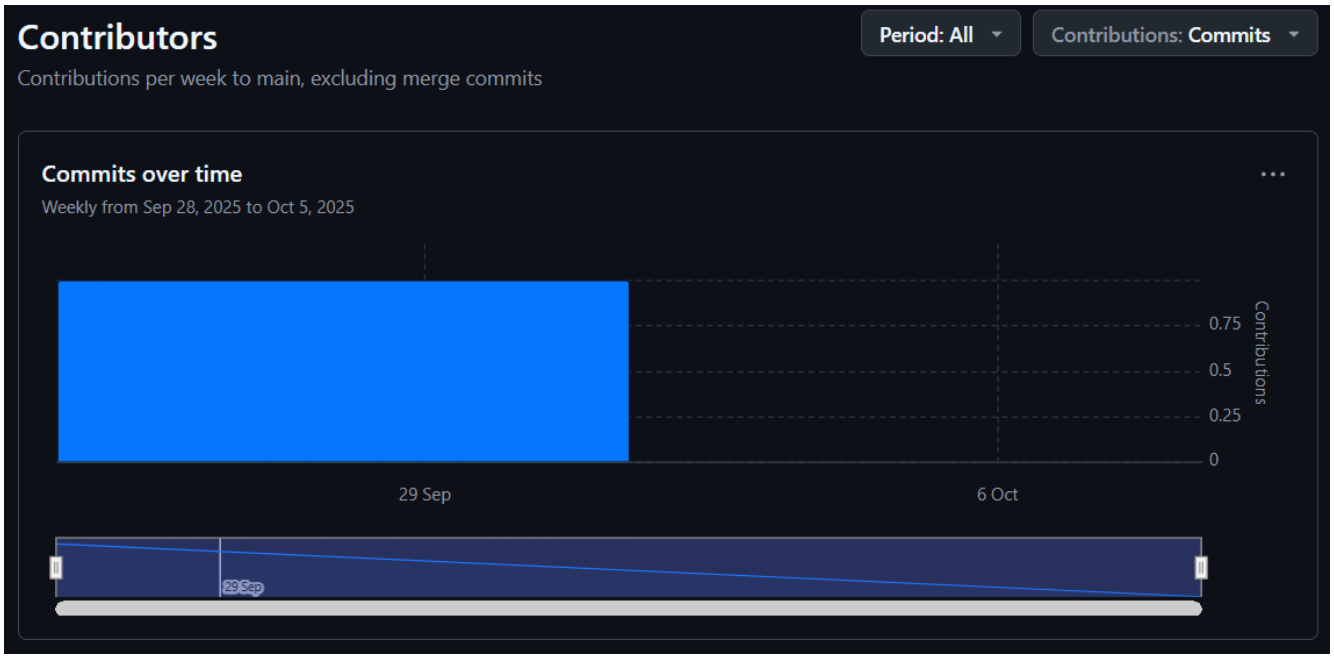
The project is **on track overall**, with slight delays only in optimizing batch insertion speed, which is being addressed through parallel processing and indexing improvements. Phase 3 tasks, including similarity search and HNSW index creation, are planned to begin soon.

Overall, the project is progressing smoothly, with foundational components completed and integration between the AI model and database well established, setting a strong base for the upcoming search and optimization phase.

Codebase Information

The project codebase is maintained on **GitHub** under a private team repository to ensure collaborative development and version control.

- **Repository Link:** https://github.com/ABHINAVGUSAIN10/VectorSearchDB
- **Primary Branch:** main – contains the stable version of the project with verified code and documentation.



Testing and Validation Status

(Provide information about any tests conducted)

Test Type	Status (Pass/Fail)	Notes
1. PostgreSQL & pgvector Configuration Test	✅ Pass	Verified installation, vector data type creation, and similarity operator (<=>) functionality.

2. Database Schema Validation	✅ Pass	All required tables and fields for metadata and embeddings successfully created and tested with sample data.
3. Data Insertion Test	⚙️ Partial Pass	Basic insertion successful; optimization required for large batch uploads to improve performance.

Deliverables Progress

Deliverable	Description	Status
PostgreSQL Database with pgvector Extension	Database successfully configured with pgvector enabled; schema for metadata and vector storage finalized.	✅ *Completed*
Data Ingestion Pipeline	Python script developed for generating embeddings using CLIP and inserting them into PostgreSQL. Optimization for batch processing in progress.	⚙️ *In Progress*
Backend Search Module	Flask integration framework created; vector search logic and HNSW indexing scheduled for next phase.	⏳ *Pending*
User Interface Prototype	Frontend design planned for Phase 4 using HTML/CSS to visualize search results.	⏳ *Pending*
Performance Benchmark Report	Preliminary insertion time tests conducted; full benchmarking to be done after HNSW index implementation.	⚙️ *In Progress*
Final Report & Presentation	Structure prepared and documentation initiated; to be completed in the final phase.	⏳ *Pending*