The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University

Project Number: 23-0-211

Team: Ariel Hedvat, Eitan Bakirov, Yuval Bakirov

Mentor: Jossef Harush Kadouri

Checkmarx

CodeMatch

We detect stolen* code using LLM

Useful when **copy-pasting snippets** from ChatGPT or similar to ensure the code isn't **"inspired" by a protected project.**

INTRODUCTION AND MOTIVATION

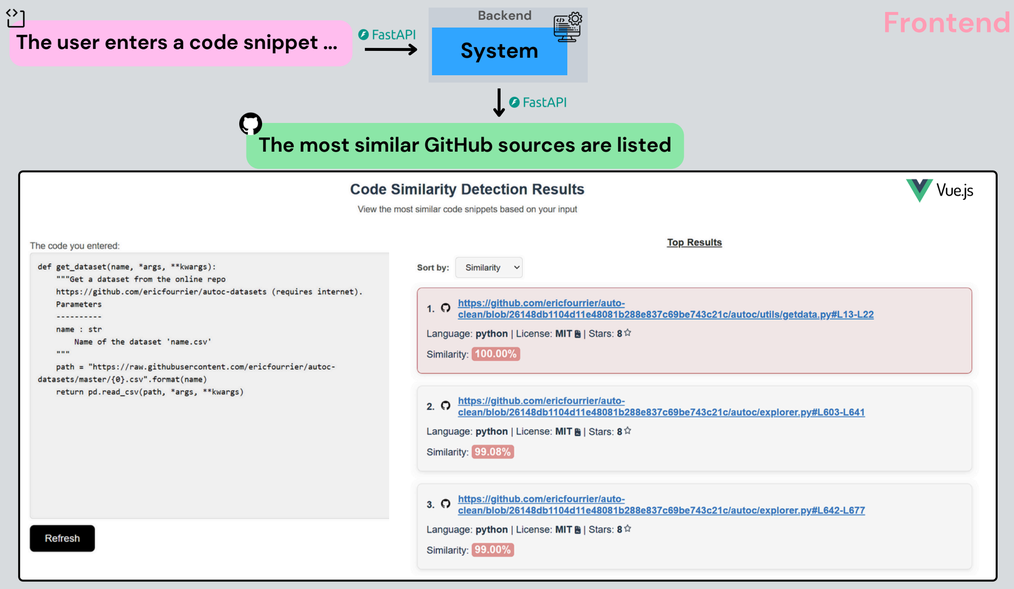
The rise of open-source software and automated code generation tools like LLMs has increased the risk of unintentional code cloning, leading to challenges in licensing compliance, intellectual property protection, and security. Tools like GPT and Copilot streamline development but raise legal and ethical concerns. This project uses LLMs to create code embeddings, store them in a vector database, and detect code clones across repositories, helping developers ensure compliance, protect intellectual property, and address security risks.

METHODOLOGY

Our methodology leverages LLMs and embedding-based similarity searches to create a scalable system for detecting code clones.
Key components include:

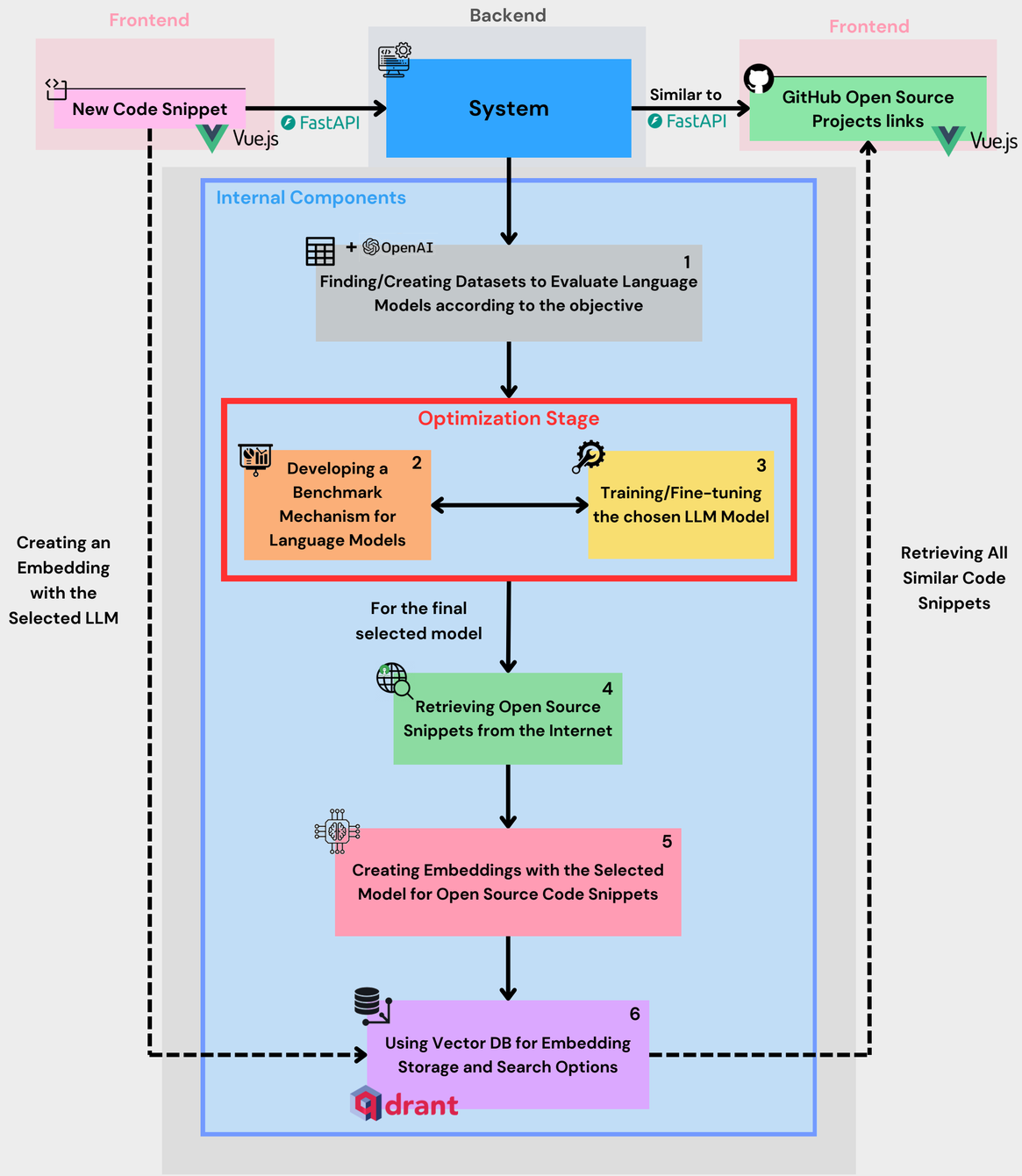
- **Choice of LLMs and Embedding Creation:**
  We select code-oriented LLMs that generate embeddings capturing the semantic essence of code, enabling detection of functional similarities despite differences in style or structure.
- **Using a Vector Database:**
  We store these embeddings in a vector database optimized for rapid similarity searches across large datasets, allowing efficient, large-scale retrieval of similar code snippets.
- **Benchmarking Approach:**
  To select the best LLM, we benchmark models against a custom dataset with various clone types (e.g., exact, renamed, semantic). This ensures we choose a model that excels in identifying nuanced code similarities.
- **Fine-Tuning/Prompt-Engineering the Best Model:**
  We use prompt engineering to optimize the top-performing model, improving its ability to identify subtle code similarities and perform well on real-world tasks.
- **Similarity Matching and Retrieval:**
  The system generates embeddings for new code snippets and retrieves similar ones from the vector database, enabling effective clone detection and linking to matching code or repositories.
- **Interface and Final Design:**
  Creating a user-friendly and well-structured interface, carefully combining it with all other components to ensure a seamless integration and build a fully functional system.

USER INTERFACE



WORKFLOW

1. **Code Snippet Input and Embedding Generation:**
   The user submits a code snippet, and the system generates an embedding using a code-oriented LLM to capture its functionality.
2. **Similarity Search in Vector Database:**
   The system searches a vector database of code embeddings to find and rank the most similar snippets.
3. **Retrieval and Display of Results:**
   Similar code snippets are presented to the user with links to their sources for review.
4. **User Review and Decision:**
   The user assesses the results to ensure originality, compliance, and proper licensing.



FUTURE WORK

Language Support: Expand from Python, JavaScript, Java, PHP, Go, and Ruby to languages like C++, Swift, and Rust for broader flexibility.
Custom Model: Train a tailored model to improve accuracy in detecting cross-language and partial code clones.
Vector DB: Enrich with diverse code from open-source projects and user contributions for better precision.
Public Access: Enable a public URL with cloud scalability for broader and easier access.
Benchmarking: Add diverse datasets and evaluation methods to enhance performance testing.