# **🧠 RAG-Based Developer Assistant System — Team Responsibilities & Architecture**

## **🔍 Overview**

This document outlines the architecture and team ownership for building a Retrieval-Augmented Generation (RAG) system designed to assist developers directly inside the IDE by intelligently responding to code, task, and documentation-based queries using LLMs.

## **🔗 Architecture Flow**

[Dev Prompt]  
 ↓  
[IDE Plugin]  
 ↓  
[Embedding Generator (Mukesh)]  
 ↓  
[Context Collector (Chirag)]  
 ┌────────────┬─────────────┬────────────┬───────────┐  
 │ Git Module │ DB Module │ Task API │ Docs API │  
 └────────────┴─────────────┴────────────┴───────────┘  
 ↓  
[Semantic Search + Prompt Builder (Survil)]  
 ↓  
[LLM (GPT-4o / Claude / Local)]  
 ↓  
[IDE Panel → Review → Apply]

## **👥 Team Responsibility Breakdown**

### **🧩 Part 1: Prompt & Embedding Generation**

**🔧 Owned By:** *Mukesh*

#### **🗂 Responsibilities**

* Capture the developer’s prompt from the IDE using plugin or extension.
* Convert prompt into vector embeddings using an embedding model:
  + text-embedding-3-small (OpenAI)
  + BGE-Large (BAAI)
  + Cohere-Embed
* Send the embedding vector to the Context Collector module.

#### **🛠 Suggested Tools:**

* IDE Plugin SDK (e.g., VSCode Extension API, JetBrains Plugin)
* Embedding API: OpenAI, HuggingFace Transformers

### **🧠 Part 2: Context Collector & Vector Retrieval**

**🔧 Owned By:** *Chirag*

#### **🗂 Responsibilities**

* Preprocess and embed content into vector store:
  + Git commit history, diffs, PRs
  + DB schemas, ER diagrams
  + Tasks from Jira/ClickUp
  + Docs from Notion/Confluence
* Store embeddings using:
  + Pinecone, Qdrant, Weaviate, or Chroma
* On prompt query, perform semantic search to retrieve top-N relevant chunks.

#### **🛠 Suggested Tools:**

* Vector DB: Qdrant / Weaviate / Pinecone
* LangChain / LlamaIndex for chunking & search
* Scheduled sync for Git/DB/Docs APIs

### **🧠 Part 3: Prompt Builder, LLM, and IDE Integration**

**🔧 Owned By:** *Survil*

#### **🗂 Responsibilities**

* Combine developer’s prompt and retrieved context into a structured prompt using template engine.

##### **🧾 Prompt Template Example:**

Developer Intent: {user\_prompt}  
  
Relevant Context:  
- Code Snippet: {function}  
- Schema Info: {table\_definitions}  
- Related Tasks: {task\_summaries}  
- Git Changes: {last\_commits}  
  
Instructions:  
Return clean, tested, and commented code ready to be merged.

* Forward final prompt to LLM:
  + GPT-4o, Claude 3, or local models via vLLM / Ollama
* Display response inside IDE panel with:
  + ✅ Code diff viewer
  + 💬 Explanation
  + 🔘 Apply / Edit / Reject buttons
  + 📝 Optional: Auto-commit, PR creation, AI co-author tagging

#### **🛠 Suggested Tools:**

* LLM Gateway: OpenAI API, Anthropic, vLLM
* Prompt templating: LangChain, custom Python engine
* IDE Integration: CodeMirror, Monaco Editor SDK

## **⚙️ Optional Enhancements**

|  |  |
| --- | --- |
| **Feature** | **Status** |
| Chunking by context type | ✅ Recommended |
| Metadata tagging (source/line) | ✅ Recommended |
| Instruction tuning for LLM | Optional |
| Support for offline models | Optional |
| PR Generation with summary | Optional |

## **📌 Summary Table**

|  |  |  |
| --- | --- | --- |
| **Module** | **Owner** | **Key Tools/Tech** |
| Dev Prompt → Embedding | Mukesh | IDE Plugin + Embedding API |
| Context Collector & Vector DB | Chirag | Qdrant / Weaviate + Git/DB/Docs Sync |
| Prompt Builder & LLM Response | Survil | Prompt Engine + GPT-4o/Claude + IDE Panel |

.