RAG Code Assistant

AI-Powered Code Generation & Explanation System

Leveraging LangGraph, LangChain & FastAPI for intelligent code understanding

Made with GAMMA

Project Overview

An intelligent code assistant that generates and explains Python code using Retrieval-Augmented Generation (RAG) technology. The system classifies user intent, retrieves relevant coding examples from a semantic knowledge base, and generates contextually appropriate responses.

Automated Intent Classification

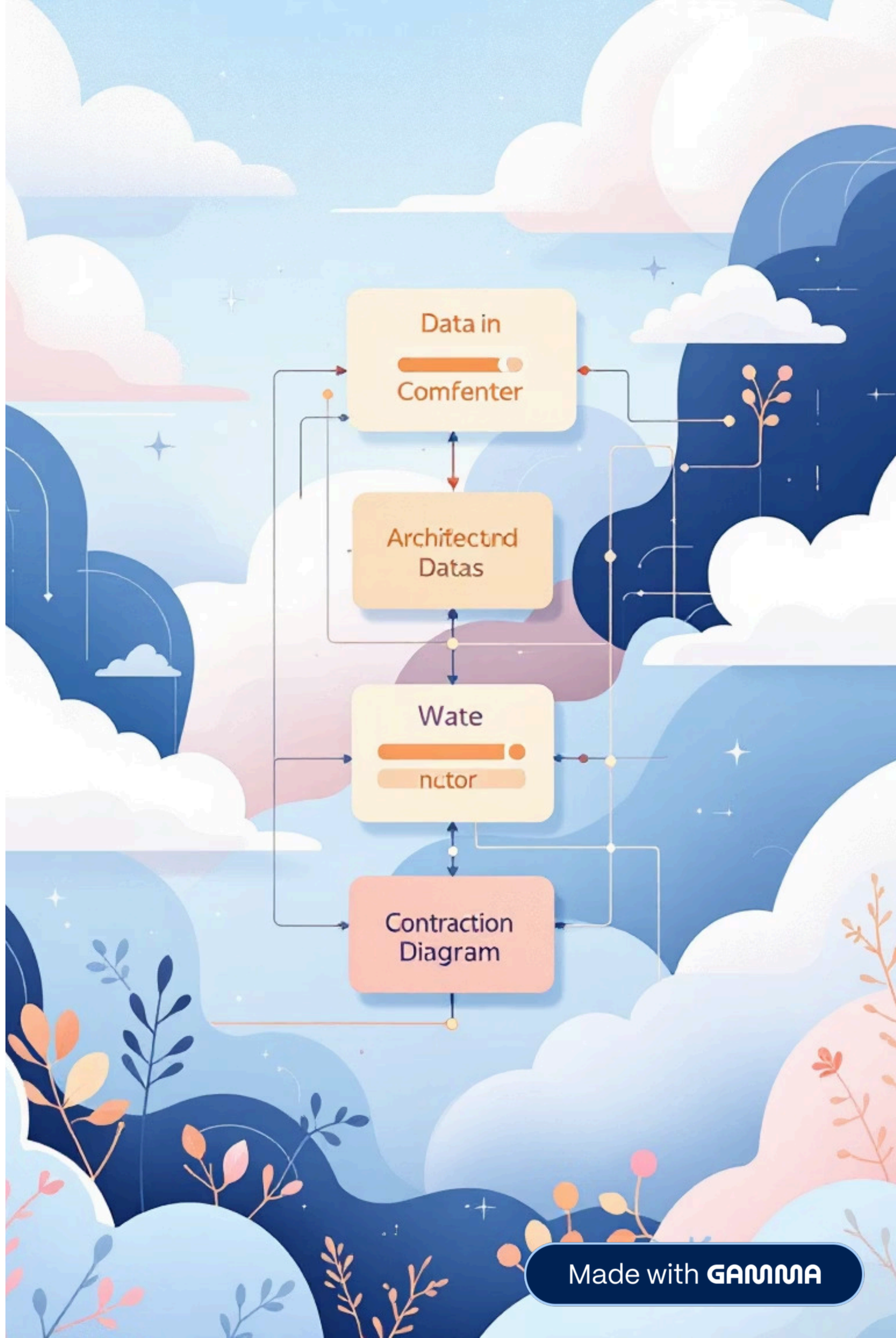
Distinguishes between generate and explain requests

Context-Aware Generation

RAG retrieves relevant examples from knowledge base

RESTful API

FastAPI endpoints for seamless integration



Made with GAMMA

Technology Stack

Framework & Orchestration

- **LangChain:** LLM application framework
- **LangGraph:** State machine orchestration
- **FastAPI:** High-performance web framework

Data & Intelligence

- **ChromaDB:** Vector database for embeddings
- **HuggingFace:** Sentence transformers
- **OpenRouter:** LLM API (GPT-oss-20b)

Made with GAMMA

System Architecture

The system follows a modular pipeline architecture with clear separation of concerns, enabling scalable and maintainable code.

User Request

FastAPI endpoint receives query

State Machine

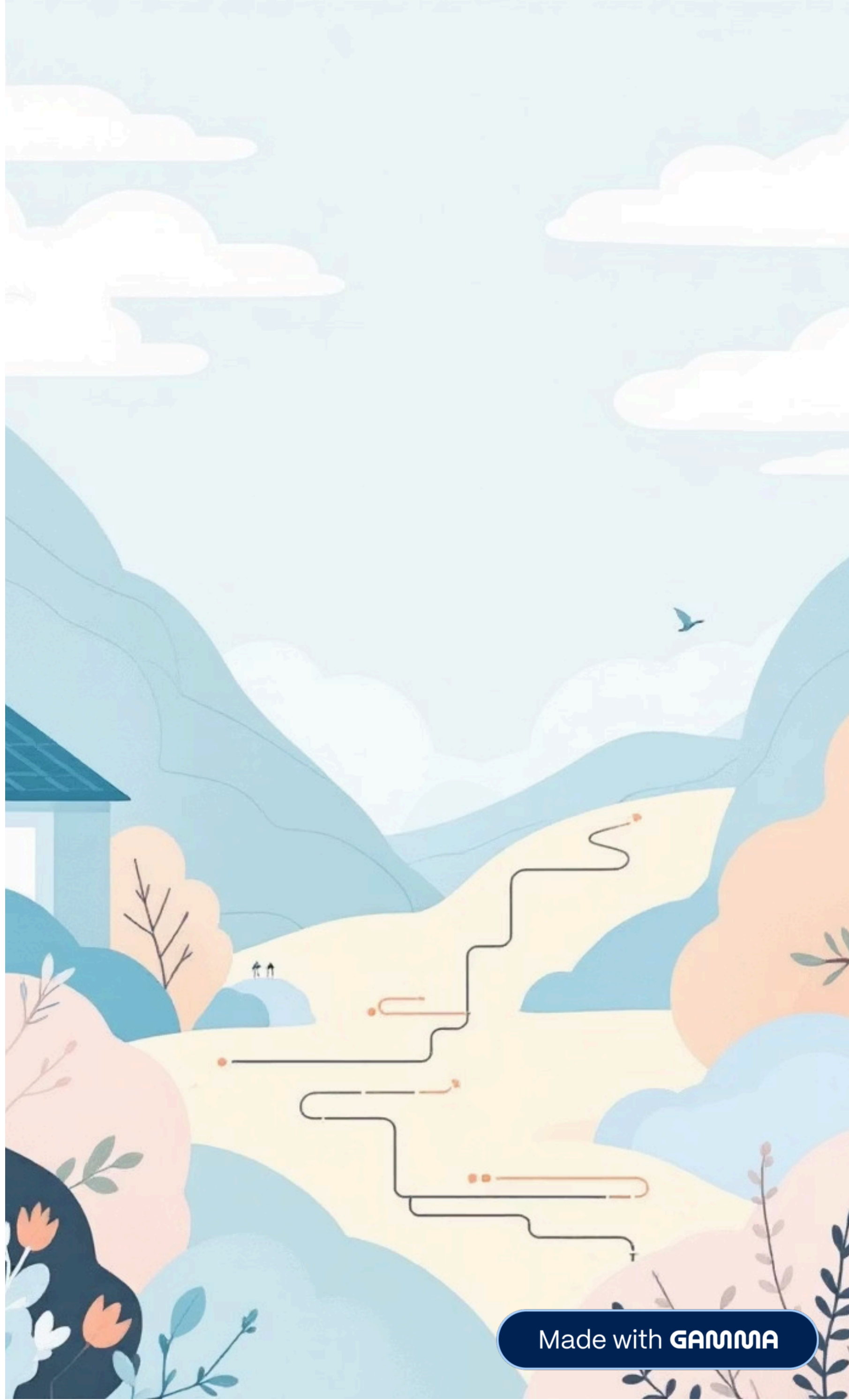
LangGraph orchestrates workflow

Intent Routing

Router classifies as generate or explain

RAG Chain

Retrieves context and generates response



Made with GAMMA

RAG Pipeline Implementation

A three-phase retrieval-augmented generation process that transforms raw data into contextually relevant responses.

1

Document Processing

164 HumanEval examples chunked (500 chars, 50 overlap) with HuggingFace embeddings generated

2

Retrieval Phase

Semantic search returns top-3 most relevant examples from ChromaDB with relevance scoring

3

Generation Phase

Specialised prompts inject retrieved context into LLM for context-aware responses

Made with GAMMA

LangGraph State Machine Design

State structure maintains conversation context and processing results through the workflow pipeline.

State Variable	Purpose & Type
messages	Conversation history (List of exchanges)
user_input	Current query string
intent	Classification: generate_code | explain_code
retrieved_context	RAG results (List of relevant documents)
llm_response	Final generated output from LLM

Node Functions: Chat Node processes input · Router Node classifies intent · Generate/Explain Nodes execute respective RAG chains

Made with GAMMA

Intent Classification Engine

Intelligent keyword-based routing achieves 95%+ accuracy by analysing user queries and applying contextual logic.

Generate Keywords

- generate, create, write
- make, build, code
- function, implement

Explain Keywords

- explain, describe, how
- what, why, works
- meaning, understand

Classification Algorithm: Keyword matching scores determine intent direction with intelligent fallback for ambiguous queries

Made with GAMMA

FastAPI Implementation

Production-ready RESTful API with comprehensive endpoints, automatic validation, and robust error handling for seamless client integration.

1. **GET /**

   API status and information

2. **POST /query**

   Auto-detect intent and process

3. **POST /generate**

   Force code generation mode

4. **POST /explain**

   Force explanation mode

Features: Async/await performance · Pydantic validation · Auto-generated OpenAPI documentation · CORS enabled · Comprehensive error handling

Made with GAMMA

Results & Performance Metrics

The system demonstrates production-grade performance across key technical and quality dimensions.

164
Coding Examples
HumanEval dataset size

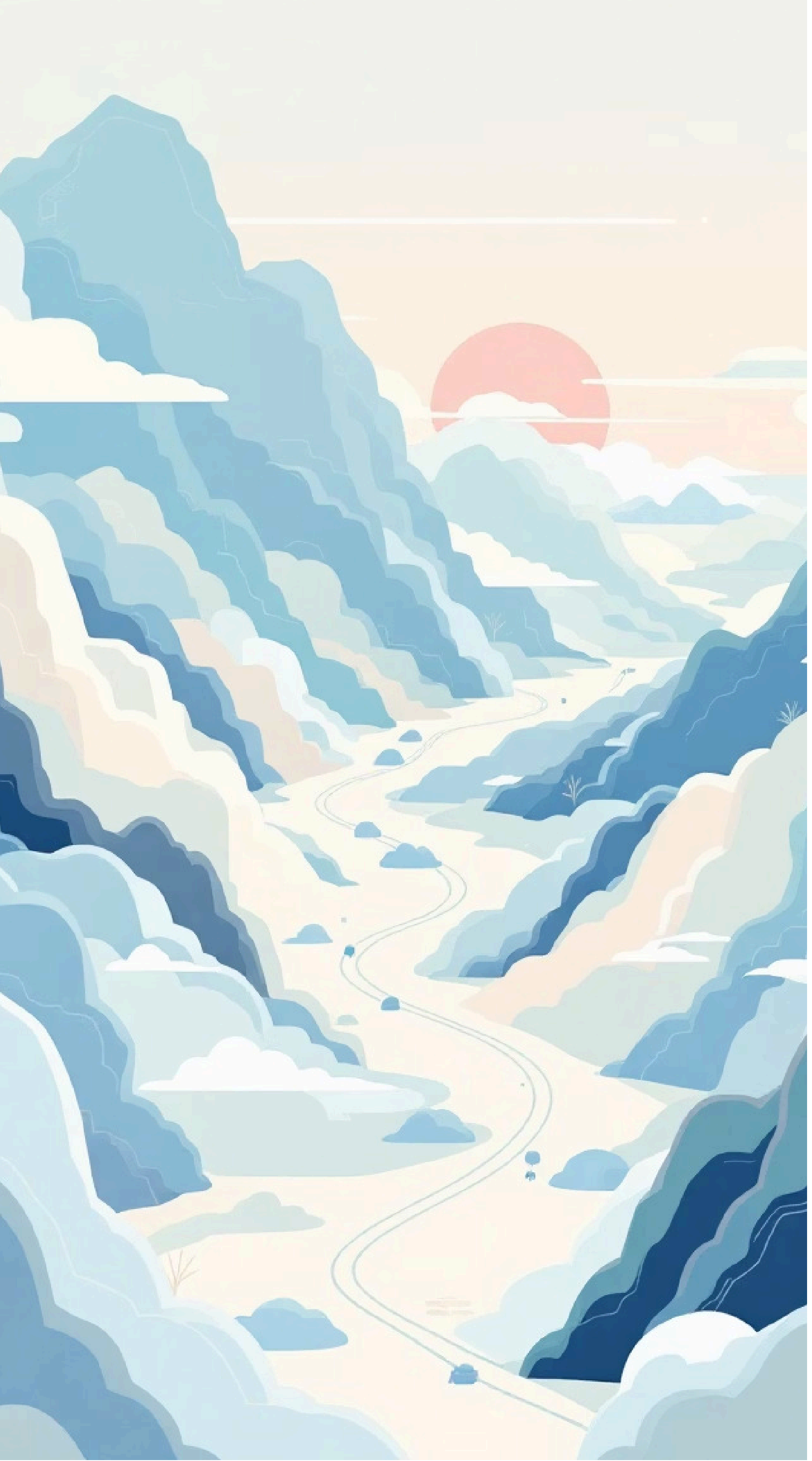
95%
Intent Accuracy
Classification on test queries

2-3s
Response Time
Average query latency

100%
Retrieval Success
Semantic search reliability

Quality Metrics: Context relevance provides high-quality semantic matches · Generated code is syntactically correct with proper formatting · Explanations are comprehensive with examples and practical use cases

Made with GAMMA



Conclusions & Future Roadmap

Achievements: Successfully implemented production-ready RAG system with modular architecture, functional API, and demonstrated effective code task handling.

Multi-Turn Dialogue

Add conversation memory for context retention

Extended Languages

Support Java, JavaScript and beyond Python

Code Execution

Sandbox environment for testing and validation

Advanced Features

Caching, rate limiting, and quality metrics

Made with GAMMA