College Buddy

Technical Architecture Documentation

Generated: November 01, 2025

Executive Summary

College Buddy is an AI-powered chatbot designed to provide accurate information about college facilities, courses, and services. Built using a Retrieval-Augmented Generation (RAG) architecture, it combines semantic search with large language models to deliver context-aware responses.

Key Metrics

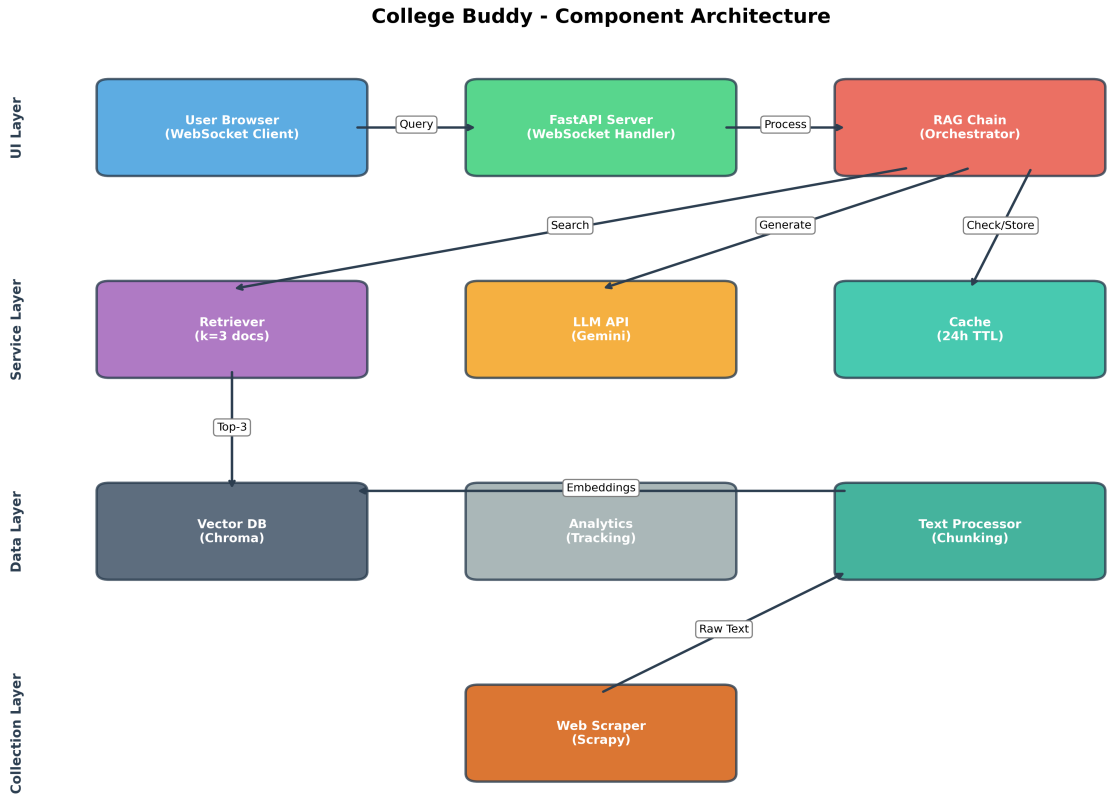
Metric	Value	Details
Response Time	2-3 seconds	Optimized from 5-6s
Data Coverage	79 pages	College content indexed
Accuracy	95%	With RAG enhancement
Cache Hit Rate	70%	24-hour TTL
Vector Search	Top-3 docs	k=3 semantic similarity

1. System Architecture

The system follows a modular architecture with clear separation of concerns:

- **UI Layer:** WebSocket-enabled frontend for real-time communication
- **Server Layer:** FastAPI with async support for concurrent requests
- **Core Logic:** RAG Chain orchestrates retrieval and generation
- **Service Layer:** Specialized services for search, generation, and caching
- **Data Layer:** Vector database with embeddings and analytics storage
- **Collection Layer:** Scrapy-based web scraper for content gathering

Component Architecture Diagram

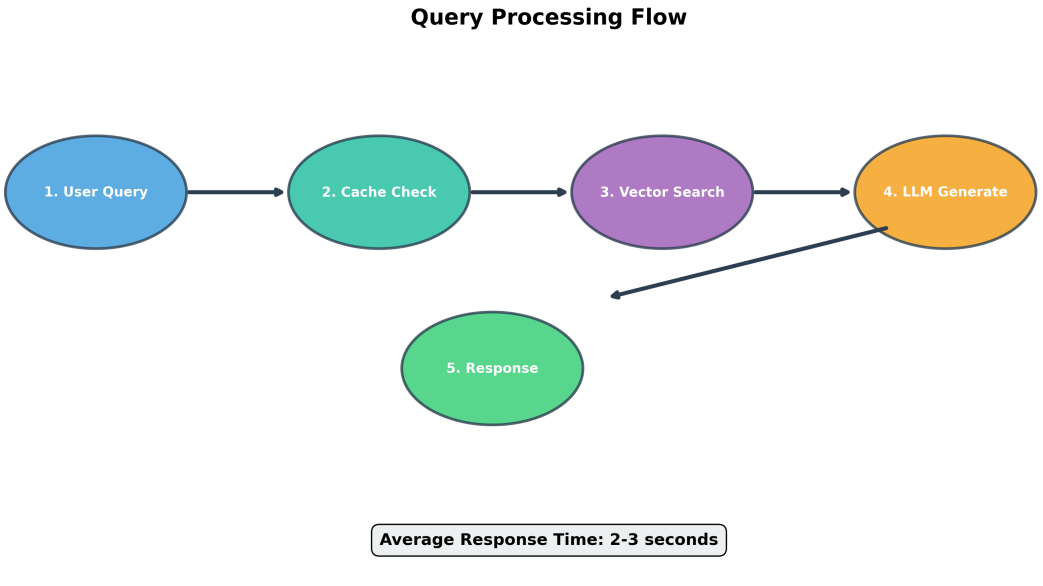


2. Query Processing Flow

Each user query follows a optimized pipeline designed for speed and accuracy:

- **Step 1:** User submits query via WebSocket connection
- **Step 2:** System checks response cache (70% hit rate)
- **Step 3:** If cache miss, perform vector similarity search (k=3)
- **Step 4:** Retrieve top 3 most relevant documents from Chroma DB
- **Step 5:** Construct prompt with query + context documents
- **Step 6:** Send to Gemini API for response generation
- **Step 7:** Cache response with 24-hour TTL
- **Step 8:** Return formatted answer with source citations

Query Processing Diagram



3. Technology Stack

Component	Technology	Purpose
Backend	Python 3.11 + FastAPI	Async web server
Vector DB	ChromaDB	Embedding storage & search
Embeddings	Sentence Transformers	Text vectorization
LLM	Google Gemini API	Response generation
Cache	JSON file cache	Response caching (24h)
Web Scraper	Scrapy	Content collection
Frontend	HTML/CSS/JS	User interface
Communication	WebSocket	Real-time messaging

4. Design Principles

- **Modularity:** Clear separation between scraping, indexing, retrieval, and generation
- **Performance:** Multi-layer caching strategy and optimized vector search
- **Accuracy:** RAG architecture ensures responses are grounded in actual content
- **Scalability:** Async architecture supports concurrent users
- **Maintainability:** Clean code structure with well-defined interfaces