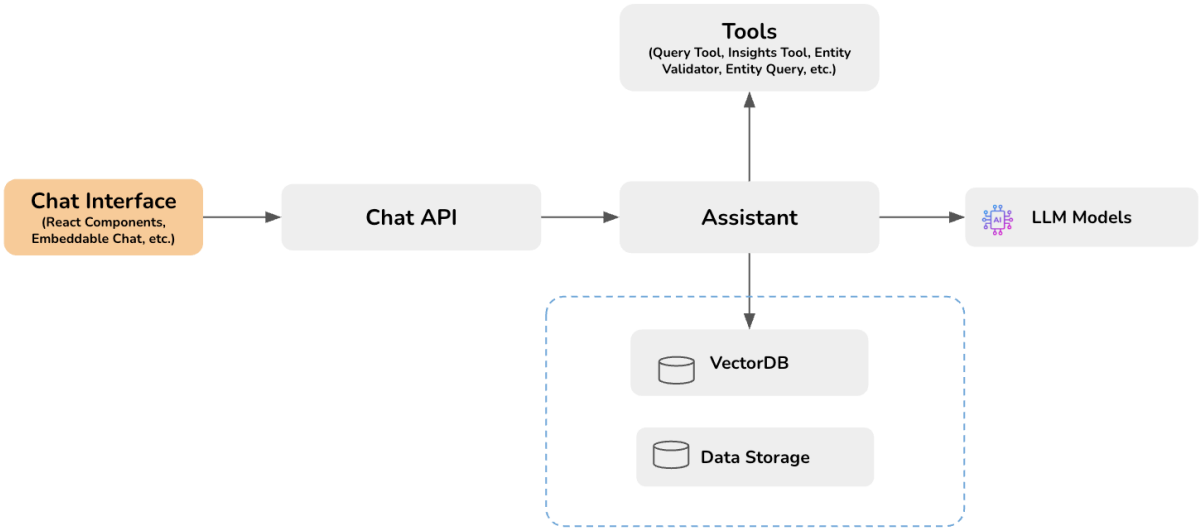
opsloom

Menu On this page

At the heart of Opsloom are assistants, AI-powered agents that can be supplied with knowledge through retrieval-augmented generation (RAG) and seamlessly connected to your organization's workflows. In the following diagram, you can see the architecture we've built around these assistants:



In addition to empowering assistants, we have followed the state of the art wherever possible. This is why we chose to build with Python, the current lingua franca of artificial intelligence. Elsewhere, we looked for technologies that were reliable, flexible, and lightweight. This led us to use uv as our build tool and package manager, PostgreSQL as our database, FastAPI as our API, and React as our frontend, with Tailwind for styling and Zustand for state management.

On this page we'll explore Opsloom's technology stack, tracing a path through the diagram above.

Chat Interface

The frontend for Opsloom was built in React, using Tailwind for styling and Zustand state management. These choices have provided us with speed: Tailwind's reusable types make development blazingly fast, while Zustand's lightweight nature provides a major performance boost over Redux.

The chat interface can also be embedded on other domains, either taking up the full size of the page or acting as a chat interface in the corner. For more details, take a look at the guide on embedding.

Chat API

Python, once the language of machine learning and big data, has now become the language of artificial intelligence. As such, it was particularly important to write the API in Python—not only because it is well suited for artificial intelligence applications but also to facilitate communication with third-party tools, most of which are written in Python.

Assistants are supported by a variety of external tools, the most important of which is the large language model (LLM). By default, assistants use the foundational model from OpenAI, but they can be modified to use any model through Amazon Bedrock. During the setup of a RAG AI assistant, we use Amazon Textract for optical character recognition (OCR), Bedrock to generate embeddings, and Cohere for reranking. We also make use of LangChain to simplify calls to and from the third-party LLM vendors.

Assistants can also leverage tools to interact with your organization's workflows. For example, they can use the text-to-SQL tool to translate natural language questions into SQL queries to fetch data from databases. Tools like these allow the assistants to access real-time information, automate tasks, and integrate with existing systems.

Database

Opsloom uses Postgres to store sessions, assistants, and chat history. In addition, we use the PGVector extension for Postgres to perform vector-based storage and similarity search, which is essential for creating RAG AI assistants.

Database startup and migration is handled by Alembic, a lightweight migration tool written in Python.

Previous page
Introduction

Next page
Setting Up Locally