Lendo product chat

AI Engineering Cohort | Design submission | 26 October 2026

I work at Lendo, a comparison site for consumer, business and mortgage loans in Sweden. Lending rules and product conditions can be hard to navigate. Customers currently contact us by phone or email. We have no chat service.

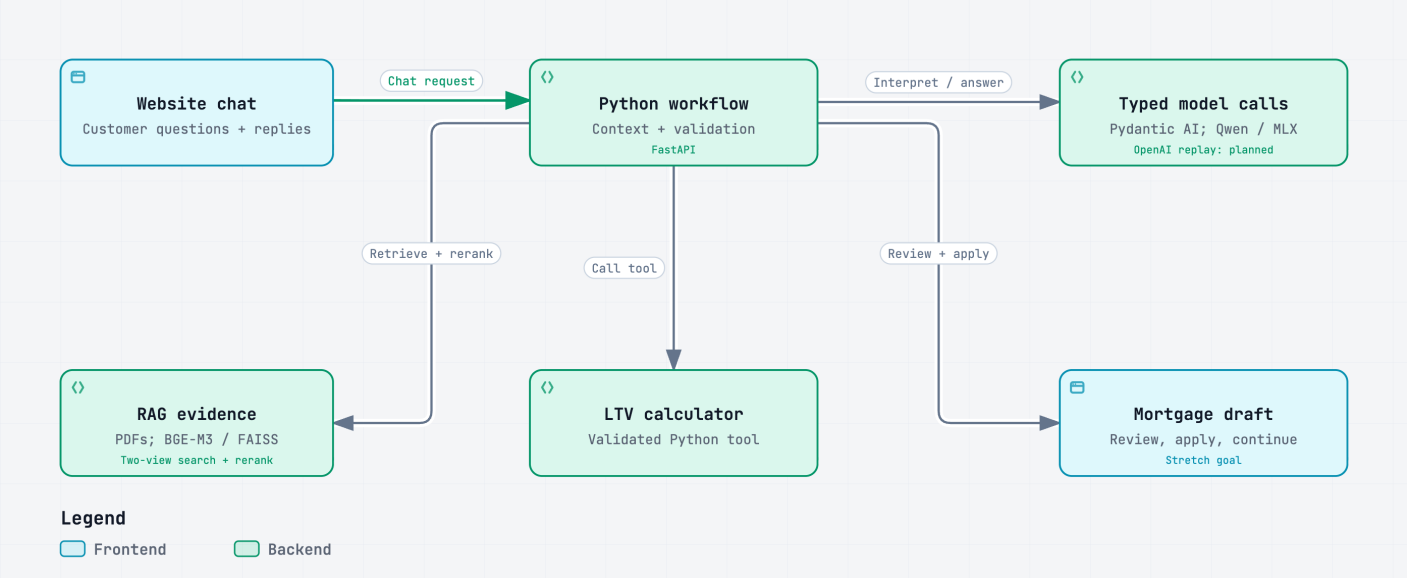
1. Problem statement and goals

I want customers to get clear answers to general product questions in Swedish and English, with citations they can check. This would give our customer relations agents more time to work on loan cases. Faster answers could also help customers resolve doubts and complete applications. I plan to propose the chat as a website feature.

The chat covers the three loan products, insurance and Kreddy, Lendo's personal finance management (PFM) app. Answers must preserve each source's product scope, conditions and exceptions. When evidence is insufficient, the chat asks for clarification or refers the customer to support. The local prototype uses public-source material and labelled capstone assumptions. It cannot access customer accounts, make individual lending decisions or submit applications.

2. Architecture

The existing frontend calls a Python service that controls routing, conversation context and validation. PDF ingestion retains source and page references before embedding and indexing. Retrieval uses versioned PDF evidence, BGE-M3 embeddings and FAISS. It searches two representations of the same records, then reranks the candidates. A Python tool calculates loan-to-value from validated user inputs. The engineering console shows evidence and model calls so I can trace incorrect answers.



3. Framework justification

I chose FastAPI for typed API contracts and Pydantic AI for validated model outputs. Keeping retrieval and tool execution in Python makes each step independently testable. FAISS supports exact and approximate search comparisons. MLX runs Qwen locally on Apple hardware.

4. Evaluation criteria and provider comparison

I ran a series of retrieval evaluations to compare chunking, embedding models, vector indexes, dense retrieval, BM25 and hybrid search. The selected dense pipeline recovered 92.9% of labelled evidence on held-out questions, compared with 89.3% for the baseline.

For answer quality, I will compare dense RAG with two-view retrieval and reranking using the same frozen cases, corpus and answer model. Swedish and English will have separate results. Targets for each language are 85% evidence recall@5, 80% task success and 80% supported-answer coverage. Success requires the appropriate answer, clarification or referral, supported facts and preserved conditions. All citation links and critical calculations must be correct. A rubric-based LLM judge, checked against human scores, will assess relevance and factual support. I will select by answer quality, then latency and cost, and report raw counts and p50/p95 latency. The README will document failures and subsequent changes.

I have mainly used local MLX inference. With the course OpenAI API key, I will rerun interpretation and answer-generation cases in both languages with retrieval held fixed, comparing factual support, citations, latency and token cost.

5. Stretch goal: mortgage form handoff

The mortgage form assistant would use details from the conversation to prepare an editable draft. The customer reviews the proposed fields, chooses whether to apply them and continues in the form.

Example: mortgage question, cited RAG answer, then a follow-up with a SEK 2.4m loan and SEK 3.0m property value. The calculator returns 80% loan-to-value. Form preview, apply and editable handoff are stretch steps; the demo cannot submit an application.

