

LEONARDO PRASETYO

Maincode application answers

Question 1 - Model training

Training production-grade models from scratch on our own clusters is demanding and requires you to learn how complex systems behave under load over long runtimes. Tell us about the hardest engineering problem you have taken on recently. Then tell us why you want to do this kind of work at Maincode.

Answer

One of the hardest engineering problems I've worked on recently is building and improving my AI portfolio assistant. It uses RAG and a vector database to allow easy retrieval for AI agent. The answers are related to the question, but they don't always give the detail I'm looking for. I suspect one chunk contain too many information, making it harder for the model's decision.

Since I can't train the model and it is a low context and low paramater model, I'm considering clearer agent instructions for different question types, guardrails to keep answers supported by the documents, and changes to how chunks are split and weighted during retrieval. My next step would be to test a set of questions while changing one thing at a time, so I can understand what actually improves the answers.

I want to work at Maincode because I want to build on my applied AI experience and learn how models like Matilda are trained. Working on an in-premise model helps me understand agent decisions, retrieval, and training works along with the infrastructure supporting it. I'm interested in the patient work of finding problems, understanding their causes, and making the systems behind AI more reliable.

LEONARDO PRASETYO

Maincode application answers

Question 2 - Frontend engineering

Building the front end for a production AI system means handling real users, real failures, and performance issues that only appear at scale. Tell us about the hardest engineering problem you have taken on recently. Then tell us why you want to build this kind of system at Maincode

Answer

One of the hardest engineering problems I've worked on recently was displaying live IoT device data in OPA, a SCADA platform at Outpost Ventures. The system uses MQTT and Modbus to gather device information. I chose polling for the frontend because the backend already handled data collection, so the interface could regularly request the latest readings. The challenge was deciding how to keep the display up to date while working within that setup. One improvement I would like to explore is connecting the frontend to AWS IoT Core using MQTT over secure WebSockets, allowing it to receive updates and display MQTT message logs as they arrive. I would also investigate how the interface should handle dropped connections, reconnecting, and showing when a reading is no longer current. These are changes I would like to test and compare with the existing polling approach. Maincode interests me because I want to bring this experience to a production AI frontend, where users need clear feedback and a responsive interface even when something goes wrong. I would like to deepen my understanding of connectivity and performance while helping build a system people can rely on

Question 3 - Backend and systems engineering

Running large scale model serving systems on our own infrastructure is demanding work. It requires understanding how distributed systems behave under load, how failures propagate, and how to keep services reliable over long runtimes. Tell us about the hardest back end or systems engineering problem you have worked on recently. What made it difficult, and how did you approach solving it? Then tell us why you want to work on building and operating systems like this at Maincode.

Answer

One of the hardest backend problems I've worked on recently was developing Konnect's agentic AI site analysis, which helps with site scoping for engineering projects. The difficult part was bringing together the site information, retrieval workflows, agent instructions, and output format so the analysis could be used by the application. I approached this by configuring agent personas and master prompts, setting up retrieval workflows, and using Azure AI Foundry's structured outputs to return the analysis in a consistent format. This required thinking about how each part contributed to the result and how information moved through the backend. Alongside this, I assisted with OPA's backend database work and MQTT communication using AWS IoT Core and Lambda. That experience made me interested in understanding how services behave when connections drop or data stops arriving, and how logs can help trace problems across different parts of a system. Maincode interests me because I want to build on this applied AI and cloud experience by working directly on model serving infrastructure. I want to understand how services behave under heavier workloads and over long runtimes, how a failure in one service affects others, and how to make problems easier to detect and recover from. I would bring experience connecting AI workflows to practical applications, alongside a willingness to learn the deeper systems work needed to operate them reliably.