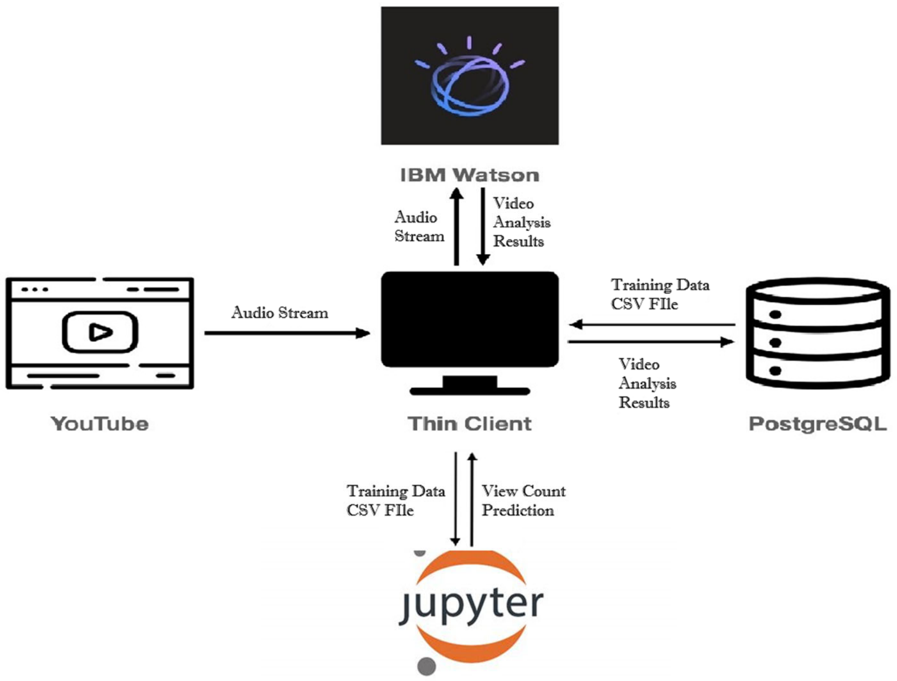

The Architecture of jiNx

Model-driven software engineering was implemented in the architecting of jiNx. We used the 4+1 Architecture view to develop baseline models from the perspectives of the development, logical, physical and process views. This MDSE approach informed our decision to engage iterative cycles of prototyping in our product development lifecycle. At the beginning of this process, models provided a high-level overview but with each consecutive iteration, added insight leant complexity to the prototypes developed from the models. Our first models allowed us to develop prototypes for simple transcription of Youtube videos. Our final models allowed us to capture data sufficient data to finetune our machine learning model and utilize that model to predict views of submitted Youtube videos.

In Domain Driven Design by Eric Evans, there is a discussion of the idea of conceptual contours. These are depicted as divisions within the domain. These divisions are intended to be highly cohesive while simultaneously remaining highly independent. Such fluidity between divisions facilitates more efficient refactoring within the domain and greater stability within the system in direct response to refactoring. Conceptual contours are a characteristic necessity of supple design in architecture. Architecting jiNx reflects this approach via the implementation of standalone classes within the module. As an example, the frontend employs a flask app which calls a databuilder class. When initialized, the databuilder class implements methods to get the video transcription, get the tone analysis of the transcription, store the tone analysis in a database record, send the record to the model to make a prediction, and finally display the result of the prediction. Each of these classes runs independently and to completion prior to the call for the next activity routed using the flask app. As Evans explains, this is an exceedingly useful pattern in supple design as it results in an overall reduction of interdependency and lends a richness to the system.

Strategy is a key component of supple design. Knowing the context, or limits of the model is imperative to the consistency of the tested performance of the prototypes conceived by the model. Unambiguous function of the conceptual contours through expediency of the rules, eliminates overlap and contradiction within the interface. A unified model is the infrastructure upon which a prototype should ideally be built. The context in which jiNx was architected began with REST API HTTP Get and Post requests and evolved to include a containerized environment for a flask app.