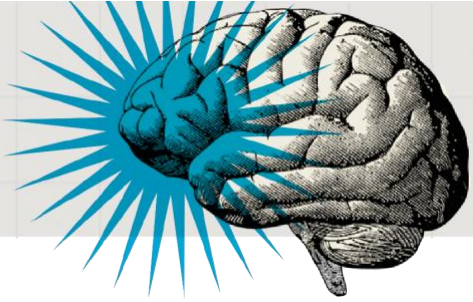


FCAT ARTIFICIAL INTELLIGENCE

A Guide to Designing a Conversational Digital Person

By: Supritam Sen, Murad Maayah, Racchit Thapliyal, Jason Meyersburg, Brandon Mino
August 2023

EXECUTIVE SUMMARY

Generative AI is a technology that can produce realistic and novel content, such as text, images, music, and more, by learning from existing data. The Fidelity Center for Applied Technology (FCAT) leveraged this technology to design and integrate a digital person, an experimental avatar that can interact with users in natural language. The project aimed to demonstrate how a digital person can help users learn more about FCAT's website and research content in an engaging and innovative way. This paper explains the design and technical aspects of a prototype that demonstrates the potential of generative AI and digital avatars for delivering information in specific domains.

We want to acknowledge both our internal Fidelity team members and our external collaborators, without whose support this would not have been possible. From Fidelity - Mark Heid, Hema Balasubramanian, Deepak Sivaraman Kuppa Sadhahari, Kerri Lanigan, Jenson Joy, Meagan Hove, Yogith Senapathy, Srikanth Yachamaneni, Brian Marianelli, Heather Hall, Sarah Abdelmessih, Monika Garofano, Marilou Rohrbach. From Slalom - Dev Worah, Trupti Sanikop, George White. From Soul Machines - Zara Jillings. And special thanks to the FCAT marketing team including Kristin Kanders and Melissa Calise for their support through the publishing process.

TABLE OF CONTENTS

Overview 3
Project Vision & Objectives 3
Solution Description 3
Digital Person 4
Appearance 5
Voice & Language 5
Personality & Interactivity 5
Context & Function 5
Integration with Custom Skills 6
Intent Routing 7
Semantic Search 8
Retrieval Augmented Generation (RAG) 8
Enterprise Search Engine vs. Vector Databases 9
Large Language Model (LLM) 9
Prompting 10
Orchestration 11
Traditional Orchestration 11
Advanced Orchestration 11
Orchestration Tool (AWS Step Functions) 12
Integrating Step Functions 12
Latency Improvements 14
The Future for AI Powered by Orchestration 15
Security considerations 15
Controlling hallucination and privacy 15
Future Direction 15

Overview

Advances in technology have proliferated the use of virtual characters that can be controlled using humans or software. These characters, often called avatars, are used by companies to engage, and serve their customer bases. The authors of the paper, "An Emerging Theory of Avatar Marketing"[1] have synthesized prior research and business practices to produce a 2x2 taxonomy which include behavioral realism and form realism. Behavior realism implies the intelligence of the avatar whereas form realism refers to the anthropomorphic appearances. In this paper, we discuss how the Fidelity Center for Applied Technology (FCAT) designed and created 'Katalina', FCAT's experimental brand ambassador. The FCAT team collaborated with Soul Machines (a leading deep-technology company that produces digital people and avatars) to design and capture Katalina's form realism and created an architecture combining both traditional and generative technologies for behavioral realism. Designing a digital person required several design considerations and best practices when determining appearance, voice and language, personality and interactivity, and context and function. Technical architecture included considerations like intent routing, usage of semantic search to understand the user's intent and prompting to guide the responses of the avatar. We also discuss orchestration patterns using AWS step functions to empower an AI-driven architecture that enables flexibility and scalability. Finally, the paper discusses the benefits of advanced orchestration techniques.

Project Vision & Objectives

The Fidelity Center for Applied Technology (FCAT) developed a proof of concept of a digital person powered by generative AI. While digital avatars backed by large language models (LLMs) that support open book/domain conversations, or even simple domain-specific conversations, are starting to appear on the market, this project set a goal to enable queries of FCAT's Priorities Report research content.

For this experiment, the following objectives were set:

- To create an intelligent and emotionally engaging digital person avatar that can interact with users.
- To enable responsiveness in its conversations, the avatar will be integrated with a large language model (LLM) in the backend. The experimental avatar will also answer domain-specific questions, those related to FCAT[2] and the FCAT Priorities Report, as well as non-domain specific questions that require knowledge outside of FCAT and priorities report.
- To create an architecture that would enable flexibility, scalability and is built on cloud technologies.

Solution Description

At a high-level, our solution consists of the following.

- An experimental digital person avatar-based interface created using Soul Machines Digital DNA Studio and Autonomously Animating Digital People technology that allows the user to interact with the system via voice or text.
- An intent classification system that determines the user's goal and helps route the execution flow downstream.
- An orchestration service that determines the action to be taken or appropriate services to be called, depending on the user intention.
- A knowledge base that serves as a repository of organizational content.
- A large language model that serves as a completion service and gives a natural language response to the user for domain-specific or non-domain specific questions.

Figure 1 lists the services and solutions used and Figure 2 provides a high-level architecture diagram, both used in this project, respectively.

CATEGORY	SOLUTION	RATIONALE
Web App	Soul Machines	Furnishes an emotive, multi-modal interface (chat and voice) that makes the interaction appear more natural.
Cloud Environment	AWS	Provides essential building blocks for our proposed solution and aligns with Fidelity's own environment preference.
API Gateway	API Gateway	Allows the Soul Machines Digital Person to communicate with our back-end solution hosted in the AWS environment.
Serverless Compute	AWS Lambda	Event driven services that allow us to logically and selectively execute "layers" based on user queries.
Natural Language Processor	Amazon Lex	Classifies user intent based on queries, to be used for downstream execution of relevant logic in the step and lambda functions.
Orchestration Service	Step Function	Allows us to maintain state and orchestrate the multiple lambda functions, and selectively execute code based on user intent.
Search Engine	Amazon Kendra	Crawls and indexes Fidelity's data, which can then be queried against to answer user questions. Reduces reliance on unpredictable LLMs, reduces opportunity for "hallucination," makes it easier to add/remove content (vs. fine-tuning LLMs), less of a black box.
Password Management	Secrets Manager	Stores our API keys in a secure manner.
Cloud Storage	Amazon S3	Storage for activity logs and Fidelity content (to be indexed).
Summarization	Amazon Sagemaker	Makes the response from AI shorter and, thus, improves the user experience when spoken by the Digital Person (Optional).
Large Language Model	Azure OpenAI	Leveraged as a "response generator" to make Kendra responses more conversational and serves as a fallback for the system to have (somewhat) open-ended conversations with the user, providing a more natural experience.

Figure 1: List of services and solutions used along with rationale.

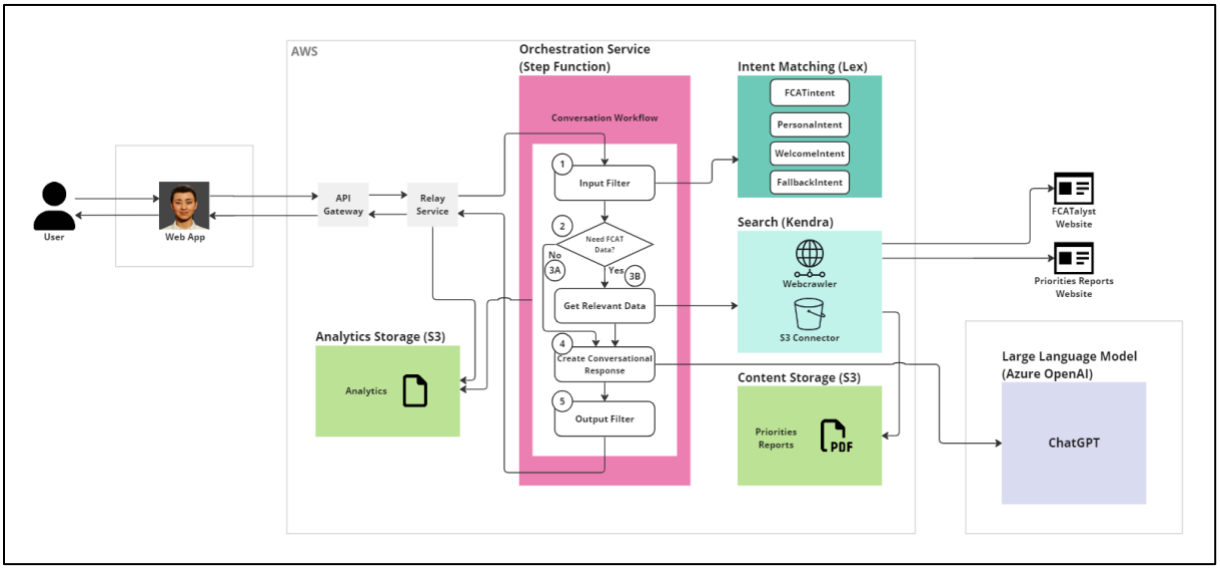


Figure 2: The architecture includes several services to create a plug and play model to reduce vendor locking

Digital Person

Designing a digital person requires several design considerations and decisions regarding appearance, voice and language, personality and interactivity, as well as context and function. These elements help create a compelling and effective virtual persona. See the supporting paper, "Building a Digital Person: Design Best Practices" for a more detailed explanation of the design process and principles. Soul Machines was selected as the Digital Person provider due to their hyper-realism, advanced autonomous animation engine enabling engaging and empathetic interactions, and flexible DDNA (Digital DNA) Studio creation and integration platform.

Appearance

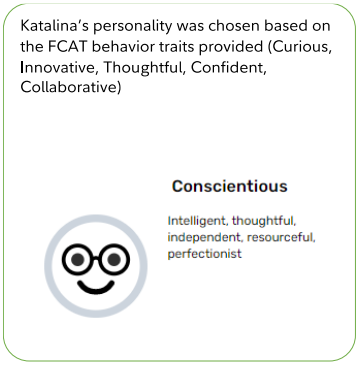
The digital person's appearance is the first thing users see. Its design should aim to be visually appealing and appropriate for the audience and context. A digital personality encapsulates the traits and characteristics that inform how people see your digital person. Age, gender, ethnicity, background, education, interests - all influence personality. One should consider the target audience, inclusivity and branding when designing a digital person. Consistent with the theme of the project, the baseline for Katalina's visual appearance was created using generative AI. A GPT model was prompted, "describe a persona and physical description of an employee for a financial services innovation lab." After some back and forth with the LLM, the physical description was fed to multiple AI image generators. Then, one image was chosen based on realism and the camera angle that matched the front-end solution specifications. The chosen image was manually recreated (i.e., "eyeballed") in Soul Machines Digital DNA Blender, enabling us to customize the physical representation from the face shape and facial feature to hair styles and colors, skin textures, makeup, clothing, and more. It should be noted that the team was cognizant of inherent biases of LLMs especially when it comes to physical descriptions of humans (e.g., race, ethnicity, gender, background). The output of the LLM was edited manually to help account for these biases.
A custom button-down shirt was designed using FCAT brand colors and the FCAT logo. Customized clothing can strengthen the digital person's representation of the brand.

Voice & Language

Voice, language, and accent are important aspects of designing a digital person, as they can impact how users perceive and interact with it. The language and accent should be tailored to the target audience, their language proficiency, cultural background, and preferences. Katalina will only speak English for the first release, and she is optimized for users speaking English with a US-dialect.

Personality & Interactivity

A digital human's personality should be consistent with its purpose and the emotions it is intended to convey. The digital person should be approachable, friendly, and engaging to encourage users to interact with it. It should be interactive and responsive to user input. Some digital people can see, hear, and reflect a user's emotional state. Katalina's personality is described as "conscientious", which was chosen based on the FCAT associates' general behavior traits[3] (curious, innovative, thoughtful, confident, and collaborative). Katalina can respond to not only what users are saying, but also the emotional sentiment they express throughout a conversation to help make the interaction more personalized, empathetic and engaging.



Context & Function

The digital person's context and function or "job role" should be considered when designing its characteristics. It should be designed with the intended use case in mind. Our solution has one key purpose:
Katalina is an FCAT ambassador, adept at discussing FCAT's research.

The functional goal for the proof-of-concept phase is for Katalina to:

- Respond to user inquiries about FCAT research with knowledge from fcatalyst.com.
- Respond to inquiries about FCAT research using knowledge from multiple years of annual Priorities Reports

- Use a broader large language model to supplement the FCAT research topic areas as well as respond to general knowledge inquiries.

To train and test her to complete this goal, four user personas were created (see Figure 3) which represented potential users with whom she would engage.

Figure 3: A set of sample user utterances was created for each of the personas, and these utterances were indexed to train the digital person's "brain" on the kinds of questions that might be asked of it.

Integration with Custom Skills

Soul Machines offers a range of built-in connectors called skills. These skills are represented by JSON files that contain important information about the services used to power conversations. While there are pre-existing skills available in other conversational platforms, we needed a custom skill to connect our system to API Gateway.
The skill file contains metadata that helps trigger the appropriate service for each conversation. In our case, the crucial metadata was the ExecutionEndpoint. This URL serves as the destination where the avatar service sends all the relevant details whenever a user inputs a new query to the digital human. For our implementation, the ExecutionEndpoint was set to the API Gateway, which then forwards the information to our backend services.
Figure 4 illustrates the implementation of our custom skill within the avatar ecosystem. By leveraging this custom skill and connecting it to API Gateway, we were able to seamlessly integrate our services and ensure smooth communication between the user and the digital person.

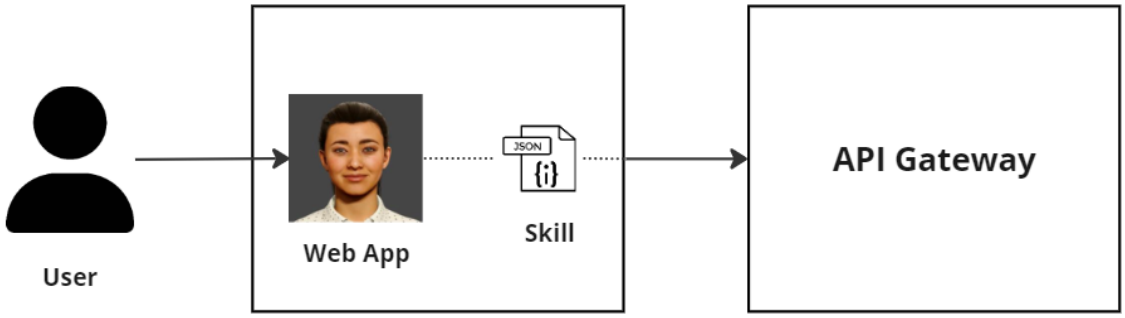


Figure 4: Custom Skill Implementation

Intent Routing

After identifying and training Katalina on the question types, the team needed to design her intent routing function. Intent routing refers to the process of determining the specific intent or purpose behind a user's input or query, and then guiding it down the appropriate path. Chatbot services analyze and understand the user's language to identify the intention or goal they have in mind.

Intent routing involves classifying user input into predefined categories or intents based on their semantic meaning. This classification enables the system to determine how to best respond to the user's request or route it to the appropriate action or functionality. It helps direct the conversation flow and enables the Natural Language Processing (NLP) system to provide relevant and accurate responses.

As shown in Figure 5, Katalina was designed to handle three intent types, enabled by four different sources, as part of the design/conceptual phase:

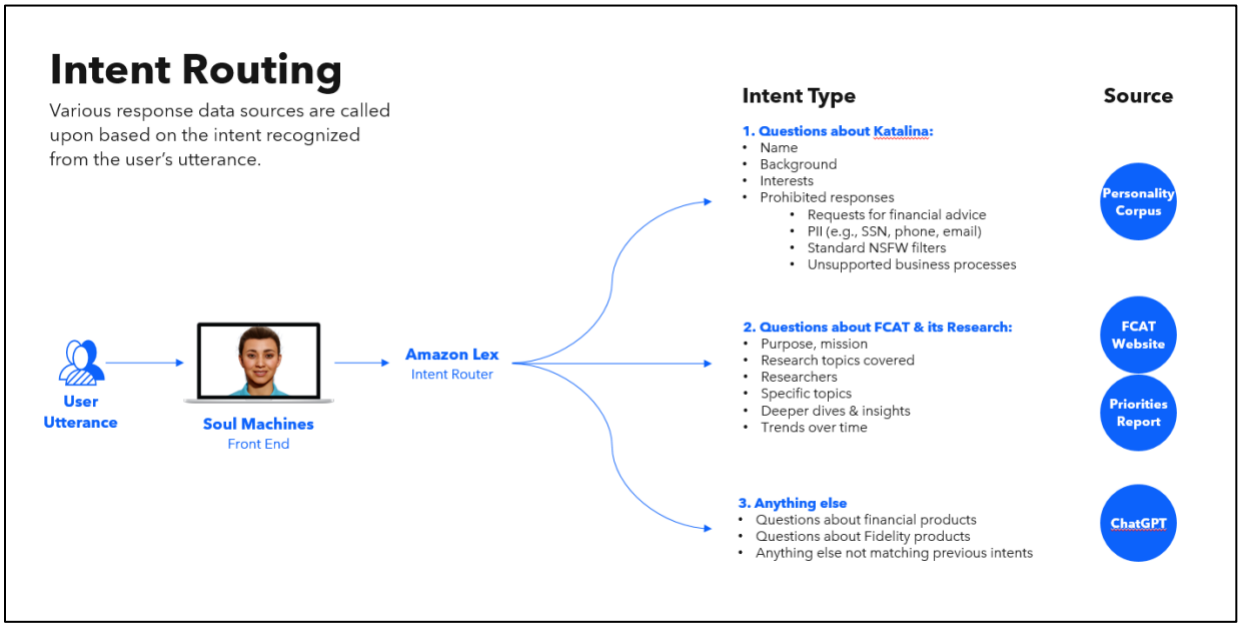


Figure 5: Katalina's intent routing enabled by different data sources as a concept. ChatGPT was used for internal experimental pilot and was designed to exclude prohibited responses to financial advice and Fidelity products. Implementation model will reassess the most appropriate LLM provider prior to deployment.

Intent Type	Source
Questions about Katalina (e.g., her name, background, interests)	Personality corpus created using generative AI and edited by the project team
Questions about FCAT *and its research* (e.g., purpose, mission, research topics covered, specific topics, deeper dives & insights into FCAT, trends over time)	Public-facing content (fcatalyst.com) Internal content (7 years of FCAT Priorities Reports)
*Anything else (e.g., questions about financial products, questions about *Fidelity* financial products, anything else not matching above intents)	ChatGPT API, to enable open book/domain questions, reflecting the data the underlying model has been trained on
Prohibited intents are captured by the NLP and responded to as required. ChatGPT is used to create the response sentences in natural language.	

Table 1: This table shows the mapping of the three intent types to sources that can be extended in the future.

Sample utterances for each intent type were defined and supplied to the intent router, and the system was "fine-tuned" based on Katalina's performance vis-a-vis user queries. The intent router, in essence, uses a Few-Shot Learning approach, wherein it generalizes from the sample utterances provided to it. This does not involve simple keyword matching, but rather, more sophisticated semantic understanding of user intent.

Katalina's modular architecture ultimately allows for simple testing and tuning based on intent routing and flexible content source selection.

Semantic Search

Semantic search goes beyond simple lexical search (keyword matching) to understand the user's intent and the holistic meaning of the query in a specific context. **This search technique helps computers understand the meaning of words and phrases based on context and relationships, rather than just their literal definition**, thus helping to provide more relevant and accurate results, even if the intent/query is phrased in an unusual way or contains misspellings.

For example, let's say you searched for the phrase "best restaurants near me". A search engine using semantic technology would understand that you are looking for restaurants that are physically located near your current location and have good ratings or reviews.

While designing Katalina, semantic search was enabled to provide more relevant responses to user queries. For example, the utterance "what's the latest in blockchain?" would not just search for articles that contain the keywords 'latest' and 'blockchain'. Rather, it would look for content in the knowledgebase that contains information on the recent developments in blockchain, thus reflecting an "understanding" of user intent.

Retrieval Augmented Generation (RAG)

While pre-training or fine-tuning LLMs on a desired domain space is a potential option, it poses several difficulties. It requires a team with sophisticated machine learning (ML) skills, expensive distributed/accelerated compute instances, a large amount of data, and a lot of time. If one gets beyond those challenges, they are faced with the specter of hallucination[4] (or, more correctly, confabulation). This is the phenomenon where the LLM (rather confidently) generates factually spurious content.

RAG provides an architectural paradigm which can effectively help alleviate the difficulties.

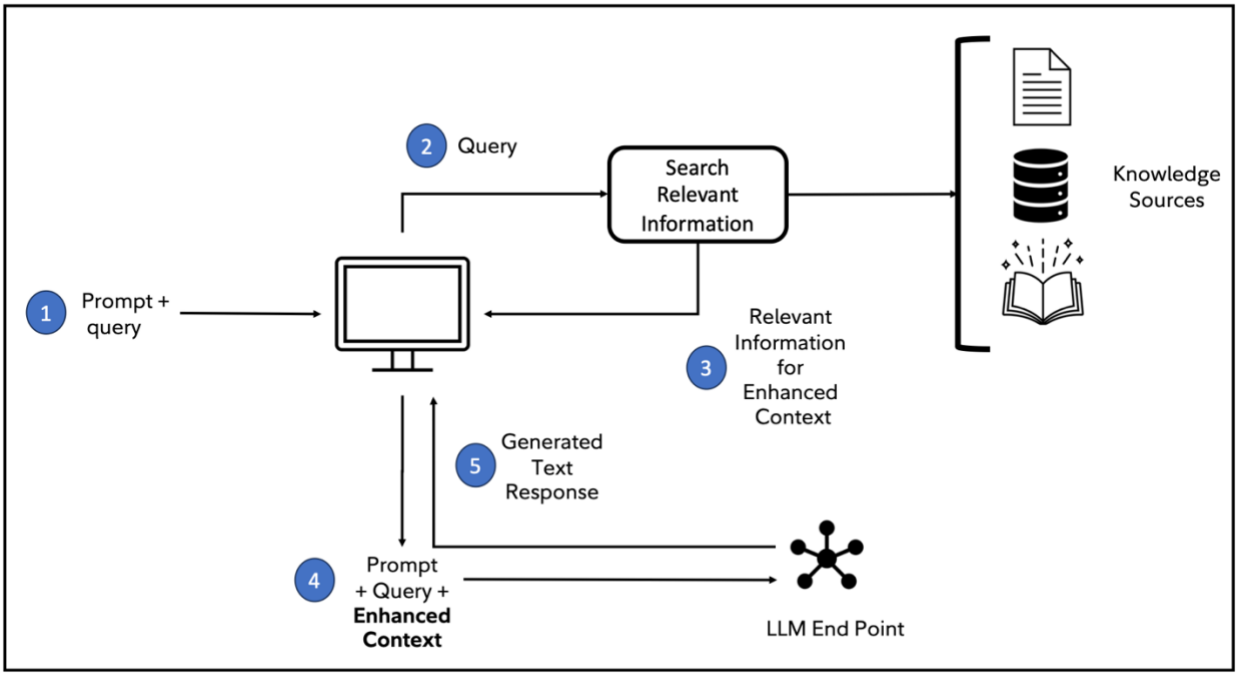


Figure 6: Retrieval Augmented Generation Architecture
(Source: Amazon Web Service's explanation of Retrieval Augmented Generation (RAG))

RAG retrieves data from outside the language model (non-parametric) and augments the prompts by adding the relevant retrieved data in context.

Enterprise Search Engine vs. Vector Databases

While it is common to use vector databases (DB)[5] as the knowledgebase, we proposed using an enterprise search engine for numerous reasons.

- First, due to its familiarity within developer communities
- Second, the process of onboarding documents (especially using a managed service, as we demonstrate below) is much easier than building and maintaining a vector embedding pipeline.
- Third, vector DBs and embedding algorithms require a specific skillset (data science) to build and maintain, and we wanted to avoid that bottleneck.
- Finally, since our goal was to quickly surface useful information in-situ (within a document or link) and give a short response connected with it, rather than purely rely on LLM generated response, we believe this approach provided for a better user experience.

After we began our implementation, our approach was validated by posts written by major cloud providers, and they are also recommending this paradigm for similar use-cases.

Large Language Model (LLM)

A large language model (LLM) is a type of artificial intelligence that is designed to understand and generate human-like language that is natural, like a person would. It's called a "large" language model because it works by analyzing a very large amount of data to learn and identify patterns and relationships between words and phrases when used in different contexts. Using this knowledge, LLMs predict the next word in a sentence or the next few words that are coherent and makes sense, given the context of what has come before it.

Prompting

Prompts are the text-based input to large language models and play a vital role in instructing and guiding the responses received. Prompts can contain different roles to specify their purpose and context. Beyond simplistic, monolithic prompts, sophisticated APIs to LLMs provide for a more fine-grained approach to generate better content. This may differ based on the provider. For the Azure OpenAI Service API, three roles are available to describe the input: system, user, and assistant [6].

- **System role:** This role involves sending a message to GPT to dictate how it should generate responses. One can think of this as an avenue to define the baseline behavior of the generative AI system. In this case, the system role was used to provide a description of our digital person and define the desired behavior for the responses. By doing so, it helps ensure that each response from GPT aligns with the persona of the digital person and fits the context of our use -case.
- **User role:** For the user role, text files were utilized and contained both static and placeholder content. The static content consisted of general rules and instructions, guiding GPT on how to handle the dynamic content injected into the prompt. The dynamic content, injected during execution, included contextual data (when necessary) and the user's query. By structuring the prompt in this way, we provided GPT with the necessary information to help generate relevant and contextually appropriate responses, no matter the user intent.
- **Assistant role:** In this specific use case, the assistant role was not used. The assistant role is tied to generated messages. As a result, this role typically encompasses previous messages generated by the LLM so it can create responses with context to its prior responses. However, for this scenario, the focus was on leveraging system and user role prompts to shape the responses.

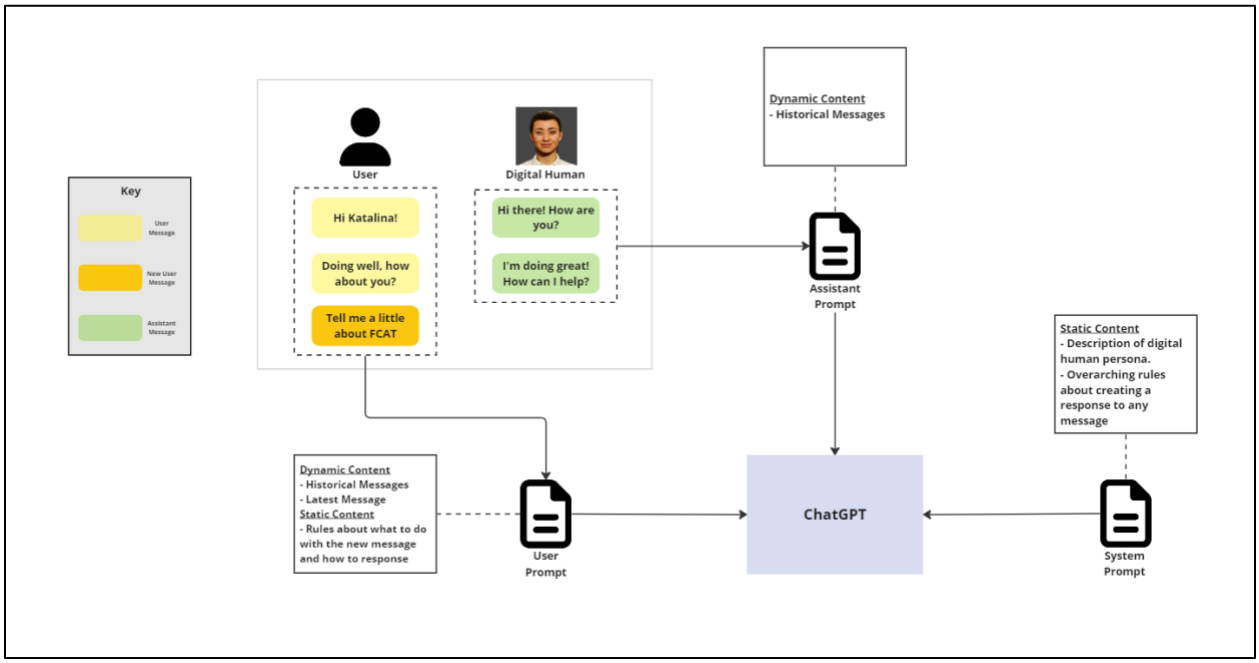


Figure 7: Prompting LLM with three roles.

By utilizing prompts in the system and user roles, we provided comprehensive instructions and context to GPT, enabling it to generate responses that more likely align with the desired persona and help meet the specific requirements outlined in the static content. This approach helped facilitate more accurate and contextually relevant answers from the language model, enhancing the overall quality of the conversation.

Orchestration

In the current state of generative AI, new tools and services are continuously emerging, enticing organizations with a wealth of possibilities to explore. With so many options, the quest to leverage these services and experiment freely while avoiding vendor lock-in remains a challenge. Moreover, seamlessly integrating generative AI services into existing systems is one of the largest challenges surrounding implementation. To overcome these hurdles and ensure long-lasting impact, orchestration pattern emerges as a compelling solution. **Orchestration pattern is an architectural approach where a central controller component coordinates and manages the execution of multiple interconnected tasks or services (known as a workflow) to achieve a specific goal** [7]. This section offers an in-depth overview of the orchestration pattern, shedding light on its application, potential benefits, considerations, and the future of conversational AI driven by orchestration.

Traditional Orchestration

Orchestration patterns have gained popularity in microservice systems due to their ability to enable interoperability. This diagram depicts a traditional orchestration architecture, where the orchestrator serves as the crucial component that connects various services together to achieve desired business outcomes.

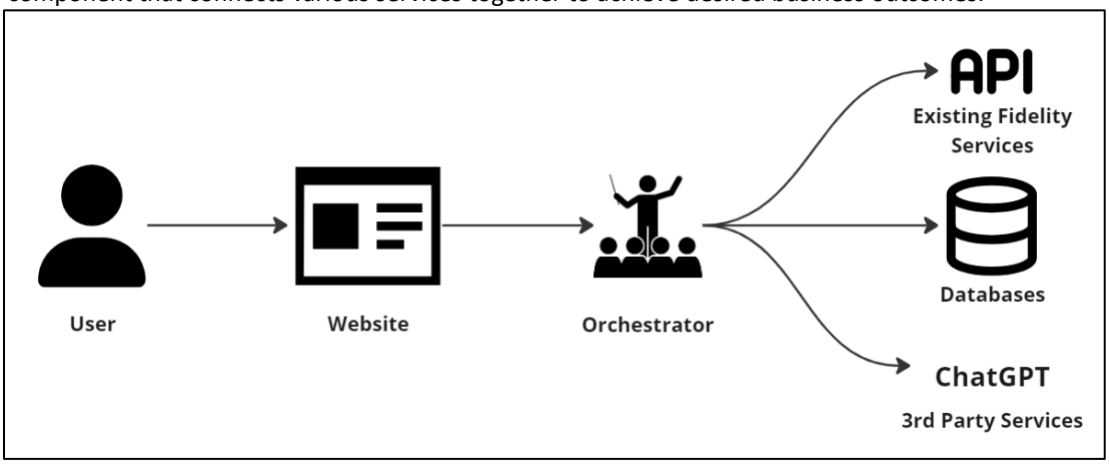


Figure 8: Traditional Orchestration Architecture

This setup offers flexibility, allowing services to be easily integrated to create customized business processes. It's important to note that while the example describes a single business process, orchestration has the capability to handle multiple processes concurrently.

Advanced Orchestration

Even with the benefits listed above, traditional orchestration architectures have certain pitfalls. One such drawback is the tight coupling of user experiences with the orchestrator. The orchestrator needs to have knowledge of all the activities it controls to provide a seamless user experience. To address these challenges, an advanced orchestration architecture was developed.

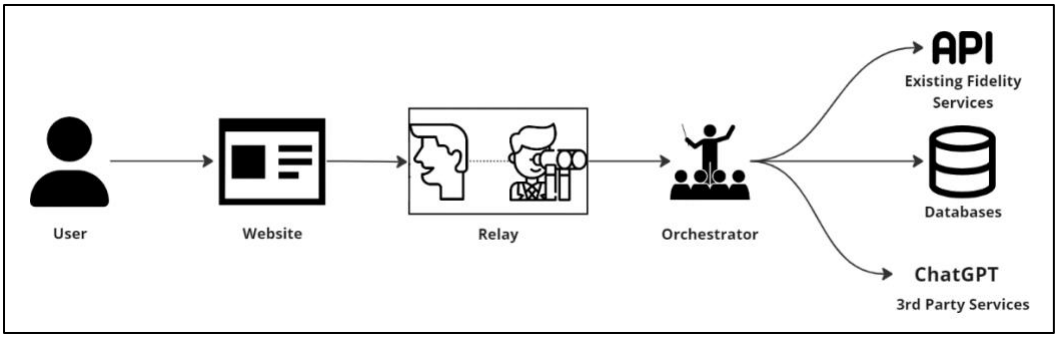


Figure 9: Advanced Orchestration Architecture

The advanced architecture shown in Figure 9 introduces an abstraction layer called the relay service, positioned between the user experience and the orchestrator. The relay service acts as an observer, monitoring the orchestrator's activities and reporting its findings back to the user. This decouples the user experience layer from the orchestrator, allowing the orchestrator to work in isolation and perform tasks without having to provide detailed information to the experience layer.

Orchestration Tool (AWS Step Functions)

To understand the power behind orchestrators, let's zoom in on their inner workings. In this specific use case, AWS Step Functions were employed as the orchestration tool. Step Functions offer an accelerated yet stable approach to orchestration, and their AWS-specific nature made them a suitable choice over traditional and non-traditional orchestration tools. With Step Functions, users can create and coordinate workflows using a visual interface. Workflow steps, decisions, and conditions are defined graphically, with each step representing an action like code execution or API invocation. Step Functions execute and track the workflow's progress, handling retries and error scenarios. Additionally, they can integrate with other AWS services, leveraging their capabilities. AWS Lambdas can also be utilized to connect with services not available in AWS. Overall, Step Functions help simplify the management and monitoring of complex workflows, thus accelerating the development of robust and scalable applications.

Integrating Step Functions

Now, let's explore how step Functions were utilized to influence OR engage in digital person conversations.

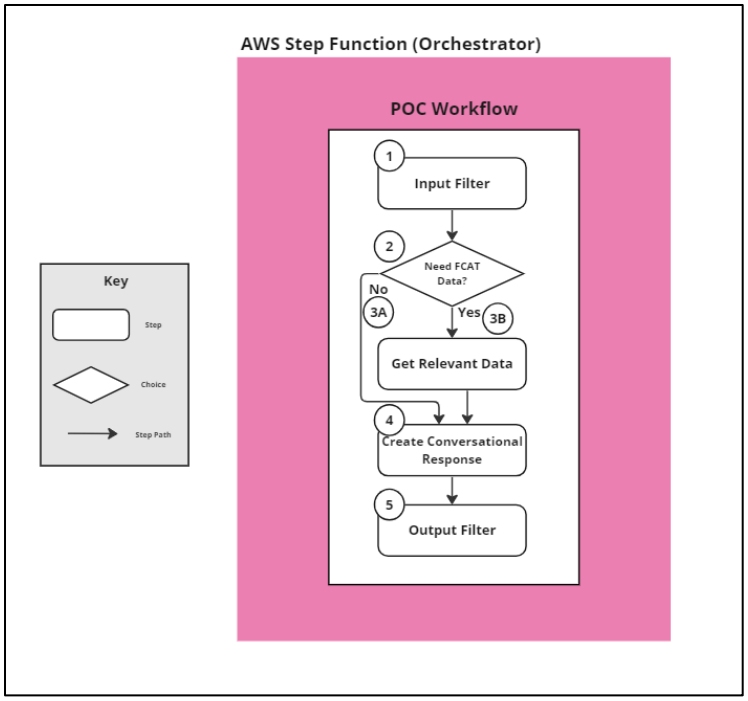


Figure 10: Step Function POC Workflow

The workflow depicted in Figure 10 consists of logical steps:

1. Filter the input from the user and determine what the user is trying to get out of their input.
2. Determine if FCAT-specific data is required. In this case, FCAT data is required if the input is related to FCAT.

3a. If FCAT-specific data is required, get the data relevant to the user's input from FCAT data sources before moving on.

3b. If FCAT-specific data is not required, skip to the next step.

4. Create a conversational response to the user input using FCAT-specific data if the input requires it.
5. Filter the output from the conversational response.

These logical steps represent the process executed by the orchestration service whenever a user inputs a new query. Once the logical steps were defined, decisions were made regarding the services that could fulfill each step, enabling their integration into the workflow, as shown in Figure 11.

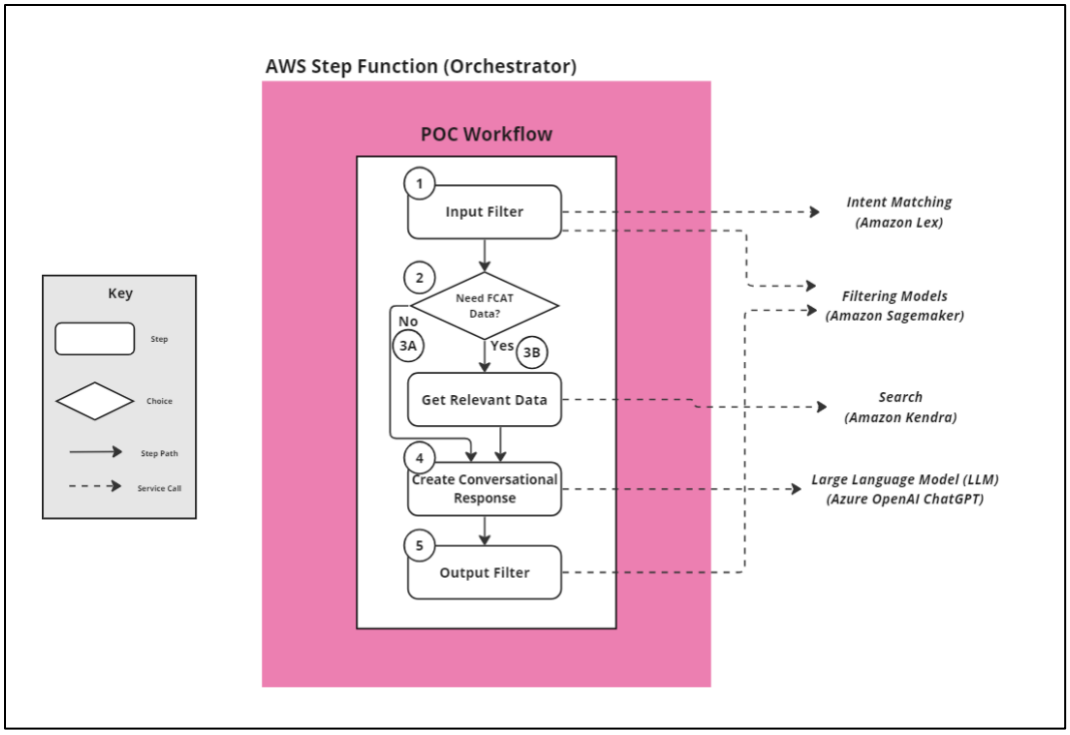


Figure 11: Step Function POC Workflow with Services

The strength of this orchestration approach lies in its flexibility to incorporate different services into the logical steps. This allows for experimentation with new technologies, such as generative AI tools, by plugging them into the corresponding steps. Moreover, the orchestration architecture facilitates the addition of new steps to the workflow in the future, enabling further expansion and experimentation with minimal effort.

Latency Improvements

Upon implementation, the system operated smoothly, but one significant issue was evident: latency. **Latency refers to the time delay between the initiation of an action and the corresponding result.** The steps in the Step Functions used a serverless service, to execute logic. However,
using a serverless service can introduce challenges for sporadic use real-time applications due to the time required to start up the underlying machine before it can execute code (known as a cold start). As a result, there was a noticeable delay (Figure 12) in the overall execution time, causing the digital person's responses to be delayed.

Conductor Lambda	Input Filter Lambda (SF-1)	Data Handler Lambda (SF-2)	Conversation Generator Lambda (SF-3)	Output Filter Lambda (SF-4)
0.542	0.13825	0.16825	0.87075	0.0685

*Figure 12: AWS CloudWatch metrics for average execution time of each Lambda **before** optimization (seconds)*

To mitigate the latency issue, provisioned concurrency was introduced. This involved enabling provisioned resources to keep at least one machine continually running, eliminating the need for cold starts. By doing so, the system helped ensure that there was no significant drop in latency caused by the delay in spinning up machines. This improvement (Figure 13) allowed for smoother and more responsive interactions with the digital person, resulting in a more natural experience.

Conductor Lambda	Input Filter Lambda (SF-1)	Data Handler Lambda (SF-2)	Conversation Generator Lambda (SF-3)	Output Filter Lambda (SF-4)
0.00225	0.04325	0.00225	0.00525	0.0065

*Figure 13: AWS CloudWatch metrics for average execution time of each Lambda **after** optimizations (seconds)*

The Future for AI Powered by Orchestration

Currently, the system utilizes a single workflow within the orchestration service. However, future use-cases can leverage multiple workflows. For instance, additional workflows could be employed to store and retrieve conversational history, providing further capabilities to enhance the digital person experience.

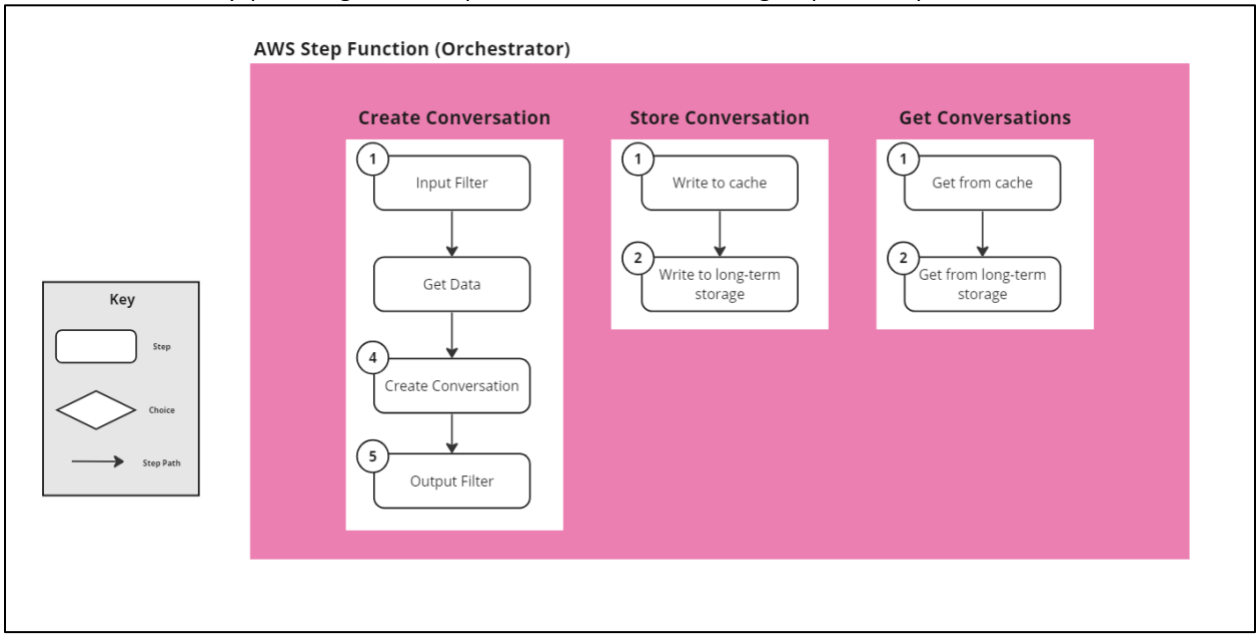


Figure 14: Simplified Example of Conversation Orchestration in a Future State

In conclusion, the orchestration pattern, particularly when implemented using tools like AWS Step Functions, empowers AI-driven microservice systems by enabling more flexible and scalable coordination of services. The advanced orchestration architecture, with its relay service abstraction layer, mitigates tight coupling between the user experience and the orchestrator. This approach offers numerous benefits, including the ability to incorporate different services, experiment with new technologies, and expand workflows in a straightforward manner.

Security considerations

Controlling hallucination and privacy

- Using a knowledge base software hosted in our own environment helps address privacy concerns about proprietary information leaving a secure environment.
- Not training the LLM directly on our data controls for hallucination/confabulation as there is a much lower chance of the LLM generating fallacious knowledge due to the use of response augmented generation (RAG).

Future Direction

- Maintaining context in a chatbot is important because it allows the chatbot to understand users' follow-up questions, relate those to previously served information and provide users with relevant and valuable

answers[8]. Moreover, context helps increase user engagement as it makes the conversation feel more natural and the intent in each message is identified and carried forward across multiple messages.

- Another area of future exploration is usage of vector databases[9]. A vector database is designed to store and retrieve embeddings. It can store the content of your documents in a format that can be easily compared to the user's prompt. Benefits include faster processing, scalability, and precise similarity matching.
- Many vendors provide capabilities to create custom user interfaces that help drive conversational experience. This includes creation using their software development kits for mobile and web development like adding charts, videos, and graphics to the user experience.
- Another area we are keenly interested in is for the LLMs to fetch data from factual sources. This may circumvent the problem of LLMs hallucinating.

---

[1] Miao, F., Kozlenkova, I. V., Wang, H., Xie, T., & Palmatier, R. W. (2021). EXPRESS: An Emerging Theory of Avatar Marketing. Journal of Marketing, 86(1), 002224292199458.
[2] https://www.fcatalyst.com/overview
[3] The behavior traits are a list of compiled information on an avatar.
[4] Hallucination (artificial intelligence). (2023, May 26). Wikipedia.
[5] What is a Vector Database? (n.d.). Pinecone.
[6] https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/advanced-prompt-engineering?pivots=programming-language-chat-completions
[7] https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/saga/saga#orchestration
[8] The Importance of Context in a Chatbot Conversation – Loyal. (n.d.). Loyalhealth.com.
[9] 8 Best Vector Databases to Unleash the True Potential of AI (geekflare.com)