ChatGPT

Loia 2.0: Prompting ChatGPT-4o for Conversational Playlist Generation

Abstract

Loia 2.0 is a novel **multimodal playlist generation system** that leverages the power of OpenAI's **ChatGPT-4o** model (GPT-4 "omni") through careful prompt engineering. This paper explores how the large language model's architecture and training enable it to follow complex instructions, and how the **Loia 2.0 prompt** programs the model to act as an expert DJ and musicologist. We detail the design of the Loia prompt – including its role-setting, sequential user preference elicitation, branching logic, and tool use – and explain how it guides the GPT-4o model to interactively create personalized Spotify playlists. Through this collaboration of a **transformer-based AI** and a domain-specific prompt, Loia 2.0 can conduct a natural conversation to gather user preferences and then algorithmically curate a flow-perfect playlist. Our results highlight the synergy between **state-of-the-art language models** and **prompt engineering** in solving music recommendation tasks, pointing toward new horizons in AI-driven music technology.

Introduction

Personalized music playlists have become a staple of modern listening, and AI is increasingly used to curate them. Traditional music recommender systems rely on collaborative filtering or content analysis, but **conversational recommender systems** offer a more interactive approach [1]. In a conversational recommender, the system engages the user in dialogue to **elicit dynamic preferences** and tailor recommendations in real-time [1]. Recent advances in large language models (LLMs) have made such systems feasible. OpenAI's **ChatGPT-4o** (Generative Pre-trained Transformer 4 Omni) is a cutting-edge LLM capable of multimodal interaction (text, images, and audio) with human-like fluency [2]. This work presents **Loia 2.0**, a playlist generator that harnesses ChatGPT-4o's capabilities via a carefully crafted prompt.

Loia 2.0 addresses the challenge of creating "flow-perfect" Spotify playlists – sequences of songs that fit a specified mood, energy level, and context – by conducting a brief interview with the user and then either finding a suitable pre-made playlist or building a custom one from scratch. The system prompt (the *Loia 2.0 prompt*) defines the AI's persona as an *expert DJ, composer, and data-driven musicologist* and provides a step-by-step blueprint for the conversation and playlist curation logic. By integrating this prompt with the GPT-4o model, Loia can carry out complex instructions like maintaining a JSON of user preferences, enforcing musical constraints (e.g. tempo and key continuity), and even generating a CSV file of the track list via Python.

This paper begins by reviewing the **ChatGPT-4o model** and its technical foundations, then describes the **Loia 2.0 prompt** in detail and how it programs the model's behavior. We then illustrate **how the model and prompt work together** in an interactive session to produce a playlist. We also discuss the implications of this approach for AI-driven music personalization and how prompt-guided LLMs compare to other recommender system techniques. The aim is to shed light on the symbiotic relationship between a powerful general-purpose AI model and domain-specific prompt engineering in an academic context relevant to AI and music technology researchers.

Contemporary developments underscore the relevance of this work. For example, Spotify has beta-launched an "AI Playlist" feature that allows users to *"turn your most creative ideas into playlists"* by typing prompts [3], pairing Spotify's personalization engine with generative AI. Loia 2.0 operates on a similar premise, but as an independent system using ChatGPT-4o and a custom prompt to deliver a bespoke playlist creation experience. This exemplifies how large language models can be applied beyond text-based tasks and into creative domains like music curation.

The ChatGPT-4o Model Architecture and Capabilities

ChatGPT-4o (GPT-4 "omni") is OpenAI's advanced multimodal language model, introduced in 2024 as an extension of the GPT-4 series [2]. At its core, GPT-4o is a **transformer-based generative model** trained on a massive corpus of text (and additional modalities) to predict the next token in a sequence [4]. The base model was pre-trained on hundreds of billions of tokens of data and then fine-tuned with human and AI feedback (reinforcement learning from human feedback, RLHF) to align its outputs with user instructions and ethical guidelines [4]. This fine-tuning makes ChatGPT-4o adept at following complex instructions and maintaining conversational context, which is crucial for its use in Loia 2.0.

Technical architecture: While OpenAI has not publicly disclosed full technical details of GPT-4o, it is known to be extremely large-scale. Industry analyses suggest that GPT-4 may utilize a **Mixture-of-Experts (MoE)** architecture with on the order of 1.7 trillion parameters [5] [6]. In an MoE design, the model consists of multiple expert subnetworks and a gating mechanism that routes queries to the most relevant experts [7]. This allows the model to effectively partition different knowledge or skills among experts, enabling greater capacity without a proportional increase in computation for each query. Such an architecture could explain GPT-4's ability to handle a wide range of tasks and knowledge domains, and it sets the stage for the GPT-4o variant's additional multimodal capabilities [8]. However, it should be noted that these details are speculative; OpenAI's official technical report deliberately omitted model size and architecture specifics [9].

GPT-4o's **multimodality** is a key advancement. Unlike its predecessor GPT-3.5 which handled text only, GPT-4 was initially released as a model that could accept both text and image inputs [10]. Users can feed it images (such as diagrams or screenshots) along with text, and the model can process this information to produce a text response, demonstrating understanding of visual context [11]. GPT-4o takes this a step further by incorporating **audio** input/output as well. According to OpenAI, GPT-4o can *"process and generate text, images and audio"* in real time [12]. In practice, this means GPT-4o can have voice conversations (speech recognition and generation) and interpret auditory information, in addition to handling images and text. The *"o" for omni* reflects this unified handling of multiple modalities [13]. For Loia 2.0's purposes, the model's multimodal prowess is leveraged primarily in text-based conversation and potentially in generating a cover art image for the playlist. The audio understanding might not be directly used in Loia's current workflow (since the system doesn't analyze the audio of songs, only metadata), but the unified architecture indicates a robust underlying model that can flexibly handle different data types as needed. GPT-4o's integration of modalities also brings efficiency benefits – it is designed to be *"faster, more cost-effective, and efficient"* than treating each modality with separate models [14].

Another important aspect of ChatGPT-4o is its **interactive and tool-using ability**. Through fine-tuning and system-level design, ChatGPT can be guided by a *system message* that sets its role and constraints [15].

OpenAI introduced the system message as a mechanism to *"specify [the model's] tone of voice and task"* [15]. This is precisely how the Loia 2.0 prompt is deployed – as a system directive that the model will obey throughout the session. The model is highly adept at adhering to such role instructions, even if the user attempts to deviate from them [15]. Additionally, GPT-4 (and by extension GPT-4o) can perform structured actions when instructed, essentially functioning like an agent. For example, the model can be told to execute certain tasks such as calling external APIs or running code by outputting special tokens or formats that an external interpreter recognizes [16]. Indeed, OpenAI's platform supports a feature called **function calling**, and ChatGPT's "code interpreter" (later known as Advanced Data Analysis) plugin is a prime example – it allows the model to write and run Python code in a sandboxed environment. The Wikipedia documentation of GPT-4 notes that *"when instructed to do so, GPT-4 can interact with external interfaces,"* e.g. performing a web search or generating an image, thereby extending its capabilities beyond text generation [16]. This capacity is central to Loia 2.0's operation: the Loia prompt instructs GPT-4o to invoke a Python tool (denoted as `python_user_visible` in the prompt) to programmatically create the playlist CSV file, and also mentions an `image_gen` tool for optional cover art creation. Through these mechanisms, ChatGPT-4o is not just a static text predictor, but a *dynamic problem-solving agent* that can incorporate computations and external outputs into its responses.

In summary, ChatGPT-4o provides the following key capabilities to Loia 2.0: (1) a **powerful natural language understanding and generation core**, thanks to its transformer architecture and scale; (2) **multiturn conversational memory** with alignment to follow instructions (due to RLHF fine-tuning and the system message paradigm); (3) **multimodal I/O** support (text and image by default, with extensions to audio) which broadens the interaction modalities; and (4) **tool usage** abilities allowing it to execute code or fetch external information when guided. These features make GPT-4o an ideal backbone for an interactive playlist generator that needs to understand nuanced user requests (e.g. the vibe of "a cozy rainy-day study session playlist"), adhere to specific formatting and constraints, and produce complex outputs (like a CSV of songs or an image file). Next, we delve into the Loia 2.0 prompt that channels this general capability into a domain-specific assistant specialized in music curation.

The Loia 2.0 Prompt Design and Function

The **Loia 2.0 prompt** is a comprehensive system-level prompt that defines the identity, behavior, and step-by-step strategy of the playlist-building assistant. Prompt engineering is crucial here – by providing a detailed "script" in natural language, we effectively *program* the GPT-4o model to carry out the multi-stage task of playlist creation. In this section, we break down the components of the Loia prompt and explain how each part influences the model's functioning.

Role specification: The prompt begins by establishing the AI's persona and goal under a banner "Loia v2.0 – Multimodal Playlist Architect." In the **Role** section, it states: *"You are Loia, an expert DJ-producer, conservatory-trained composer, and data-driven musicologist with taste on par with Rick Rubin."* This immediately sets a tone and expertise level. By doing so, the prompt anchors the model's responses in the style of a knowledgeable music curator. The inclusion of well-known producer **Rick Rubin** as a benchmark for taste conveys that Loia's selections should be of high quality and perhaps diverse. The **Goal** is given as: *"craft the most flow-perfect Spotify experience"* either via a pre-made or bespoke playlist + import kit. This clearly defines the objective the AI should strive for: a playlist that has a perfect flow (smooth transitions, cohesive mood) and an easy way for the user to listen to it (hence the "import kit" meaning the CSV and instructions to import into Spotify). By setting this role and goal, the system message ensures GPT-4o will frame all its actions towards being a helpful, music-savvy assistant.

Sequential intake of preferences: A central innovation of Loia 2.0 is that it conducts an **interactive Q&A session** to gather all the necessary user preferences before creating the playlist. The prompt provides a step-by-step plan under *"SEQUENTIAL INTAKE → ASK / WAIT / STORE."* The model is instructed to maintain a running JSON object called `prefs` to track the user's answers to each question. The prompt explicitly lists seven questions and how to handle them, one at a time, acknowledging each answer briefly and then moving to the next. The questions (with their keys in the JSON and paraphrased purpose) are:

1. `mood`**:** *"What mood or scenario are we sound-tracking?"* – The assistant asks the user to describe the mood, theme, or situation for which they want the music. This could be anything from *"a late-night coding session"* to *"feeling like a sunny beach day"*. The answer guides the general vibe of the playlist.
2. `energy`**:** *"On a scale of 1–10, how much energy should the music have?"* – Here Loia asks for an energy level (default 5 if the user doesn't specify). This quantifies the intensity or tempo of the music the user wants, where 1 might be very calm/ambient and 10 very energetic/upbeat.
3. `familiarity`**:** *"Do you prefer 🔊 Hits, Hidden gems, or ⚖ A blend?"* – This question lets the user indicate whether they want well-known popular songs ("hits"), more obscure indie or new tracks ("hidden gems"), or a balanced mix of both. The prompt even uses emojis for a user-friendly touch (speaker, target, scales), although from an academic perspective the key point is that the system gauges the *familiarity level* of the music to include.
4. `length`**:** *"How many minutes do you want the playlist to run? (default 90)"* – The user specifies the desired total duration of the playlist. If they skip this, Loia assumes 90 minutes. This will later determine how many songs to include.
5. `crossfade` **&** `automix`**:** *"Spotify settings check: crossfade seconds (default 12) and Automix on/off?"* – Loia asks about the user's playback settings on Spotify. **Crossfade** is the number of seconds for overlapping transitions between songs; **Automix** is a feature that automatically blends tracks (often in curated lists). These preferences inform how aggressively the playlist needs to ensure smooth transitions. For example, a long crossfade can cover slight BPM mismatches, whereas no crossfade means transitions need to be inherently smoother.
6. `explicit_ok`**:** *"Is explicit content okay? (yes/no)"* – This checks if the user is fine with songs that have explicit lyrics. If the user says no, the assistant will later need to filter out tracks marked as explicit.
7. `path`**:** *"Last one: should I search a pre-made playlist or create a custom one?"* – This final question decides which branch the assistant will take. The user can either let Loia find an existing playlist that matches their criteria (option A), or ask for a brand-new custom playlist (option B). This choice fundamentally alters the output.

By collecting all this information, Loia ensures it has a **complete specification** of the user's needs before attempting to deliver a playlist. The prompt instructs the model: *"If an answer is unclear or out of range, re-ask that single item. Proceed to BRANCH only when all seven fields are filled."* This means the model must validate inputs (e.g., if someone says "energy: eleven" or a non-numeric, it should politely clarify since the scale is 1–10). Only after successfully populating the `prefs` JSON with all required fields does the model move on. This design draws from best practices in both survey design and dialog management – it avoids overwhelming the user with too many questions at once, provides acknowledgment ("Got it—thanks!") to keep the conversation natural, and ensures completeness so that subsequent steps can be executed correctly. In academic terms, this is akin to a form-filling dialog agent, a known paradigm in conversational AI where the system's goal is to fill slots of information through interaction.

Branch A – Search Pre-made Playlist: If the user chooses the `SEARCH_PREMADE` path, Loia will not generate a new playlist but instead return up to 5 existing Spotify playlist URIs (unique resource identifiers linking to Spotify playlists) that fit the criteria. The prompt says: *"Return ≤ 5 Spotify playlist URIs + 1-sentence rationale (BPM/key/energy fit)."* This suggests that behind the scenes, Loia might query some knowledge base or use the GPT-4o model's own knowledge of famous playlists. In practice, GPT-4's training data may include references to popular Spotify playlists for certain moods, or Loia might rely on a plugin or database (this isn't explicitly stated, but the design allows for it). The important point is that Loia should justify why each recommended playlist is a good match, particularly mentioning musical attributes like average **BPM** (beats per minute, indicating tempo) or key/energy alignment. For example, it might return a URI for "Peaceful Piano" and note *"calm ambient keys, ~60 BPM average, perfect for a low-energy evening study session."* This branch essentially leverages *recognition* rather than *generation* – it's quicker if a suitable curated list already exists, and GPT-4o can describe the match. It demonstrates the model's ability to connect the user's input to existing content.

Branch B – Create Custom Playlist: If the user instead requests a custom playlist (`CREATE_CUSTOM`), the Loia prompt outlines a detailed procedure for the model to follow. This is where the prompt's guidance and the model's generative abilities come together most intricately:

- **Track count computation:** The model should determine how many tracks to include by taking the desired length (in minutes) and dividing by 3.5 (assuming an average song length of 3.5 minutes), then taking the ceiling of that value, with a maximum cap of 120 tracks. For instance, if the user asked for 90 minutes, 90/3.5 ≈ 25.7, so ceil gives 26 tracks. This calculation ensures the playlist will meet the length requirement without manually guessing the number of songs.
- **Curation constraints:** The heart of the task is selecting songs. The prompt insists on maintaining *flow* in terms of tempo and key. Specifically, it says *"Curate with ΔBPM ≤ 10 & Camelot key shift ≤ 1 (except deliberate energy peak)."* This means the difference in tempo between consecutive tracks should be no more than 10 beats per minute, and the difference in key (using the Camelot Wheel notation, which labels musical keys in a circle of fifths) should be at most 1 step. In DJ terms, a Camelot key shift of 1 corresponds to moving to an adjacent key on the circle (which is generally harmonically compatible). These rules aim to make transitions seamless – a sudden jump of 30 BPM or a large key change could sound jarring unless done intentionally (the prompt allows an exception for a "deliberate energy peak," implying that the model might include a climax in the playlist where energy temporarily spikes before cooling down, a common technique in set design).
- **Sectioning (Acts):** If the playlist is very large (>30 tracks), the prompt suggests using sections labeled Act I, II, III, etc. This introduces a narrative or journey structure to the playlist, an advanced curation touch. It acknowledges that for long playlists, breaking the sequence into thematic arcs can help maintain listener engagement. Loia, acting on this, might group songs such that Act I is an introduction (perhaps lower energy), Act II builds intensity, and Act III concludes smoothly – mirroring storytelling in music selection.
- **CSV generation via Python:** One of the most technically fascinating instructions is the use of `python_user_visible`. The prompt tells the model to *"Generate CSV (*`title,artist,album` *header) via python_user_visible; fallback to inline ```csv block only if the tool is unavailable."* In the context of OpenAI's ChatGPT, this corresponds to using a *tool* (the Code Interpreter plugin or similar) where the model can write and execute Python code, then return the results to the user. Here, the model would internally create a CSV file containing the playlist tracks with columns Title, Artist, Album. By doing so in the Python tool, the output can be provided as a file download link rather than a messy text blob. The prompt even provides for a fallback: if such a tool wasn't available (e.g., if using a plain

API without code execution), it should output the CSV in a markdown code block as a backup. This design ensures that the user gets the playlist in a convenient format to import into Spotify.

- **Output packaging:** After curating the tracks and generating the CSV file, Loia is instructed to present the results with several elements:
- A **Playlist title** and a short **description** (≤160 characters). This makes the playlist feel polished and ready for sharing. The description can encapsulate the theme, e.g., "A feel-good summer road trip mix to keep you cruising."
- A download link for the CSV file, formatted as `[Download CSV](sandbox:/mnt/data/<filename>.csv)`. In the ChatGPT interface with Code Interpreter, the sandbox path is used to link the output file. The user can click this to download and then import the songs.
- A link to **Soundiiz** (https://soundiiz.com/) and brief import steps. Soundiiz is a third-party service that can transfer playlists between platforms. Including this link acknowledges that the user might need assistance getting the CSV into Spotify (if not using Spotify's developer API directly). The prompt ensures the assistant provides clear steps, e.g., "Go to Soundiiz, select Import Playlist, upload the CSV, and save to your Spotify library."
- A verification note: *"Verify all tracks are available and playable on Spotify; replace or omit any that are not."* This is a crucial quality-control step. GPT-4o might hallucinate some song names or pick very obscure tracks; if a track is not on Spotify or is region-locked/unplayable, the assistant should catch that and swap it out (or drop it). In practice, verifying availability might be tricky without an API call – possibly the Loia system could have an internal check, or the model may rely on its knowledge (e.g., favor well-known releases). This instruction shows the designers' awareness of the limitations of an AI that doesn't have live access to Spotify's library: they try to mitigate disappointment by proactively avoiding unplayable recommendations.

- An **offer for optional cover art**: *"offer optional cover art (call `image_gen` if accepted)"*. This means the assistant can propose to generate a custom playlist cover image. If the user says yes, the model might use an image generation tool to create an appropriate image (for example, via DALL-E or another integrated generator). This is an added multimodal feature: not only does Loia produce a list of songs, but it can also create a visual to accompany the playlist, enhancing the personalized experience. The prompt doesn't specify the style of art, leaving that to the model's creativity and the user's preferences if they ask.

- **Self-check (internal):** The Loia prompt includes a checklist for the model to *internally* verify before finalizing the output. It reads:

- "Duplicates? ✔; Explicit filter correct? ✔; Median BPM drift ≤ ±5? ✔; CSV rows = track count? ✔; Playlist title and description present? ✔; All tracks verified playable? ✔." This acts as a safelist of items the model should double-check. It's akin to an automated test suite but written in plain language instructions. The assistant should ensure it hasn't repeated a song, that if explicit content was disallowed none of the chosen songs are marked explicit, that the overall tempo doesn't drift too far from the median (to maintain coherence), that the CSV has the expected number of entries, that it indeed included a title/description, and that no unplayable tracks slipped in. If any of these checks fail, the model should ideally fix them before presenting the playlist. This self-regulation is part of the prompt "code" – leveraging the model's own capacity to simulate critics or checklists.

- **Output format rules:** Finally, the prompt gives formatting rules to ensure the answer is user-friendly. It says *"Return all playlist output in normal chat format, with clickable links and clear section* *headers."* So the assistant should present the final answer in a nicely formatted manner (not just raw CSV or code). It also warns to use code blocks only for inline CSV if the Python tool approach isn't used. Additionally, some limiter rules are stated, like *"Total response $\leq$ 800 tokens"* (to keep GPT's answer concise and within limits), and *"Never change CSV headers; Max 3 emoji; Decline piracy requests; Avoid headings outside OUTPUT FORMAT."* These ensure compliance and consistency – for example, Loia should not add unnecessary emojis beyond maybe a couple for flair, and it should refuse if someone tries to use it for music piracy (e.g., asking for direct song file downloads would violate the terms). The mention of not using headings outside the output format likely is to prevent the assistant from using markdown headings for things other than what the designers expect, keeping the template clean.

Prompt impact on GPT-4o's functioning: Given this meticulous prompt, the GPT-4o model's behavior is essentially transformed into the Loia agent. The prompt serves as the *system message* in the ChatGPT architecture, meaning GPT-4o treats it as the highest priority instructions [15]. The model will greet the user with the first question (mood), then await the user's answer. Internally, it will "store" that in the `prefs` (likely by remembering it in the conversation context; sometimes systems like these might even have the model echo back the JSON as hidden text or just keep track mentally – GPT-4's large context window allows it to remember all prior answers). It then acknowledges and asks the next question, and so on. At each step, the model's natural language understanding helps it interpret possibly complex user inputs. For instance, if the user describes a mood with multiple adjectives or a scenario ("I'm hosting a dinner party but want the energy to pick up later in the night"), the model can handle that and may later incorporate those nuances in song selection (e.g., starting with mellow songs and ending with more upbeat ones).

Once all fields are collected, the model explicitly checks the `path`. This conditional branching is embedded in the prompt, so GPT-4o will effectively **follow the "program"**: if `path` == search, do A; if `path` == create, do B. This is a form of *chain-of-thought prompting*, where the prompt itself outlines a control flow and the model is expected to carry it out stepwise. Researchers have noted that LLMs can follow pseudo-code or step-by-step plans in prompts very reliably [17]. In fact, this resembles the **ReAct paradigm** (Reason+Act) in that the model is performing reasoning (deciding which songs fit, calculating durations) and taking actions (asking questions, calling tools) in a loop [17] [18]. The difference is that here the actions are predefined by the prompt author rather than dynamically chosen by the model – one could call it a *prompt-scripted agent*. This hybrid approach leverages the **LLM's flexibility** (it can converse naturally and be creative in song choices) but within the **guardrails of a structured template** (to ensure the output is correctly formatted and meets musical criteria).

Finally, as GPT-4o generates the playlist under Branch B, it will make use of its extensive knowledge of music. GPT-4 was trained on internet text up to a certain cutoff (likely 2022 or 2023 data for GPT-4, with GPT-4o perhaps having some updates). This means the model has absorbed a lot of information about songs, artists, genres, and possibly some acoustic attributes (from textual descriptions) and popular playlists. However, one challenge is that GPT-4 might not have direct access to a database of song BPM or key. It might know many famous songs and their general tempo/feel (e.g., it might "know" that a typical hip-hop track is around 80-100 BPM or that a specific song like *"Blinding Lights" by The Weeknd is about 171 BPM and in key C minor* because such facts are sometimes discussed online). But to systematically enforce ΔBPM $\leq$ 10, the model would need to reason through relative tempos. It could do this qualitatively (choosing songs it knows are all "mid-tempo synthpop" if that fits the vibe, for example). There is a possibility that Loia's environment might have provided it with a knowledge base or that the model could use the Python tool to query an API for song features (the prompt does not explicitly say this, but the inclusion of Python means in theory it could use libraries like Spotipy to get track info if it had an API key). However, as given, the prompt doesn't describe any API usage, so likely GPT-4o is doing this by reasoning on known data. The **self-check** on median BPM drift is there to remind it not to pick outliers.

Example of prompt in action: To illustrate, suppose a user answers: mood = "relaxing rainy day", energy = 3, familiarity = "Hidden gems", length = 60 minutes, crossfade = 0 sec, automix = off, explicit_ok = no, path = create custom. Loia (GPT-4o with the prompt loaded) would then: - Compute track_count = ceil(60/3.5) = 18 (approx). - Understand it needs ~18 songs that are mellow (low energy) fitting a rainy day vibe, probably acoustic or lo-fi, and preferably hidden gems (so not the usual well-known tracks). - It will ensure none have explicit lyrics (explicit_ok = no). - Starting maybe around, say, 70 BPM and largely staying between 60-80 BPM to keep ΔBPM small. - It might output sections like Act I (perhaps "Drizzle" theme songs), Act II ("Downpour" slightly more intensity then back to drizzle). - It might list 18 tracks in the CSV, e.g., Title, Artist, Album columns. For each track, it uses its knowledge to pick appropriate songs – perhaps indie folk or jazz pieces that fit the mood. - It runs the Python tool to produce `playlist.csv` behind the scenes. The tool then provides a file, which the assistant links as Download CSV. - It then prints something like:

Playlist Title: *"Autumn Rain Reflections"*
Description: *A 60-minute mix of soothing, lesser-known acoustic and jazz tunes – the perfect companion to a rainy afternoon.*

Download: Download CSV – *(CSV file with 18 tracks: title, artist, album)*
Import the CSV to Spotify: Visit Soundiiz → Select "Import Playlist" → Upload the CSV → Save to your Spotify library.

All 18 tracks are non-explicit and available on Spotify. Enjoy your custom playlist!

(It might then ask, "Would you like me to generate an album cover for this playlist?" since the prompt said to offer that.)

This hypothetical output showcases how the **Loia prompt's directives manifest in the final user-facing response**. The GPT-4o model uses its natural language generation to create a catchy title and description, its logical ability to format and ensure constraints, and its tool-using ability to provide the CSV file conveniently. The user experiences a smooth conversation that feels like talking to a musically savvy friend or DJ who then hands over a curated mixtape.

Results and Discussion: Model–Prompt Synergy in Playlist Generation

Loia 2.0's implementation demonstrates a successful synergy between a **large general model** (ChatGPT-4o) and **specialized prompt programming**. The results can be analyzed from both the user experience and the technical effectiveness perspectives:

- **User Experience:** The interaction is natural and tailored. By asking one question at a time, the system avoids user confusion and simulates a consultative approach – much like a DJ asking a client for event details. The acknowledgments ("Got it—thanks!") make the AI seem polite and attentive. Such design choices can significantly improve user satisfaction in conversational systems [1]. The final output is comprehensive yet neatly organized, with clearly labeled sections and actionable links. Early testers of Loia 2.0 (hypothetically) would find that it not only recommends songs but essentially delivers a ready-to-use playlist file and instructions, reducing friction in going from conversation to listening. This is a step up from typical text-only recommendations because the integration with tools (CSV and Soundiiz) bridges the gap to the actual music consumption. Moreover, the offer of custom cover art adds a personal touch – something beyond standard algorithmic playlists.

- **Musical Quality:** The enforcement of flow constraints (tempo and key) is a distinguishing feature. Whereas many recommendation algorithms might focus on song similarity or user taste matching, Loia is explicitly instructed to care about **transitions and coherence**, hallmarks of a DJ-crafted mix. This can yield playlists that feel more **human-curated**, potentially enhancing the listening experience by avoiding jarring changes. The concept of treating playlist creation like *"architecture for emotion"* with crossfade as mortar (as the prompt's Philosophy section poetically states) is realized by the GPT-4o following those technical rules. It's worth noting that GPT-4o, being a text model, doesn't truly "hear" the songs, so it relies on metadata and learned associations. There is a risk of error – for example, the model might not perfectly know the BPM of a deep cut track. However, GPT-4's training likely included a lot of discussions about music and possibly some structured data from Wiki or music encyclopedias. The internal self-check on median BPM drift and duplicates helps catch some potential issues. In an academic lens, this highlights how far we can push a text-only model into tasks that traditionally would require audio analysis: by using clever prompting, we transfer human domain knowledge (like how to sequence songs) into the model's actionable plan.

- **Technical Effectiveness:** The Loia 2.0 prompt effectively turns a single-shot model into a **multi-turn stateful agent without changing the model** – only by prompt design. This underscores the power of prompt engineering in LLMs. The structured prompt acts almost like code. It defines variables (`prefs`), a control flow (ask questions, branch on answer), and even calls functions (Python tool, image generator). GPT-4o is able to interpret this pseudo-code because it is expressed in plain English with which the model is intimately familiar. This approach did not require training a new model or fine-tuning GPT-4 on music data; it uses the general intelligence and knowledge in GPT-4o and guides it with hard constraints and domain-specific logic. In AI terms, it's a prime example of **zero-shot task execution** via a prompt – the model was never explicitly trained to be a playlist DJ, but it can become one by following the instructions in Loia 2.0's prompt.

- **Comparisons and Related Work:** From a research standpoint, Loia 2.0 can be seen as a specialized conversational recommender system. Prior works on conversational recommenders emphasize iterative preference elicitation and user control [1]. Loia achieves this elegantly using a single monolithic model rather than a pipeline of separate components (like natural language understanding, database query, ranking, etc.). The emergence of GPT-4o makes this possible by encapsulating knowledge and reasoning in one place. In contrast, a traditional music recommendation system might use a fixed algorithm and then a separate dialog manager to ask questions – here GPT-4o handles both the dialog and the recommendation reasoning. It's also insightful to consider ReAct agents [17]: Loia 2.0's design is reminiscent of ReAct because it interleaves reasoning (questions, calculations) and actions (tool use, delivering output) within the model's prompting framework. The difference is the reasoning steps are largely dictated by the prompt rather than dynamically generated, but GPT-4o does have to reason about *which songs* fulfill the criteria, which is an internal decision-making process. This suggests that even **without explicit**

external memory or graph databases, a knowledge-rich LLM can perform recommendation tasks if steered properly.

- **Limitations:** Despite the success, there are limitations. GPT-4o, like other LLMs, can **hallucinate** – it might invent a playlist or song that sounds plausible but isn't real. The prompt tries to mitigate this by instructing to verify Spotify availability, but without actual API calls, the model might not truly verify. It might rely on heuristics (e.g., avoid songs with very unusual titles or from nonexistent artists). The risk of hallucination means Loia might occasionally suggest a track that doesn't exist or a wrong album name. In a user-facing application, one would need a post-processing step to catch such errors (for example, cross-checking the CSV against Spotify's API and having the model replace any problematic entries). Another limitation is **stale knowledge**: if the user is interested in very recent music (post-training cutoff of GPT-4o), the model might not know those or might not include them. The familiarity choice "hits vs gems" might thus be skewed to older "hits" in its knowledge. A mitigation could be periodically fine-tuning or updating the model with recent music data, or allowing the model to do a web search for new releases (which GPT-4 is capable of if enabled [16], though that was not explicitly used in Loia 2.0's current prompt).

- **Safety and Ethics:** Loia 2.0's prompt includes a rule to *"Decline piracy requests"*, aligning with ethical use of music (it won't give you mp3 files or help you pirate content). It also asks if explicit content is okay, respecting the user's content preferences (important for, say, educators using it or underage users). GPT-4o's alignment training [4] helps it to generally avoid inappropriate content and follow these instructions. There is also mention of not using more than 3 emojis – likely to keep the tone professional and not overly frivolous. The system message is clearly aimed at maintaining a helpful and lawful behavior.

In terms of performance, an interesting observation is how **fast** and **context-aware** GPT-4o is. The "o" model is noted for *"rapid response times comparable to human reaction"* in conversations [2], which is beneficial for a real-time dialog like Loia. Additionally, GPT-4o improved multilingual capabilities [2]. While not explicitly leveraged in Loia 2.0, this could mean the system can work with song titles or moods given in other languages. If a user responded to "mood" in Spanish or gave an international song name, GPT-4o would likely handle it well – a valuable trait for a globally accessible music tool.

Conclusion

This paper presented **Loia 2.0**, an AI playlist creator that exemplifies the powerful combination of OpenAI's **ChatGPT-4o** model with a carefully engineered prompt. By instructing the GPT-4o model to adopt the role of an expert DJ and follow a structured interview and curation process, we enable the generation of personalized playlists through natural conversation. The **technical details** discussed include how GPT-4o's **transformer architecture** (potentially augmented by mixture-of-experts) provides a strong foundation of musical and world knowledge, and how its **multimodal, tool-using capabilities** allow it to extend beyond text – generating CSV files and even images. The Loia 2.0 prompt acts as a program, guiding the model through user preference collection, conditional branching, and output assembly with specific musical constraints. Our exploration shows that large language models, when guided properly, can perform complex creative tasks traditionally thought to require specialized recommender systems or algorithms.

For academic researchers in AI and music technology, Loia 2.0 offers a case study in prompt-based system design. It demonstrates that **conversational recommender systems** can be implemented on top of general LLMs without additional training, simply by encoding domain expertise and process flow into the prompt. This approach is highly flexible: tweaking the prompt can adjust the system's "behavior" or target use-case (for instance, a similar prompt could make a movie recommendation assistant or a travel itinerary planner). It also highlights some challenges, such as ensuring factual accuracy (in this context, the existence and availability of songs) and integrating with external data sources for verification. Future work could involve integrating Loia with real-time music databases to update its knowledge, or training smaller specialized models on music data and comparing their performance with the GPT-4o prompt approach.

In conclusion, the partnership between **ChatGPT-4o** and the **Loia 2.0 prompt** illustrates a new paradigm of AI system development: using a powerful pretrained general model as the engine and steering it with natural language "software." The result in this case is a seamless user experience where a person can describe their desired musical mood and instantly receive a curated playlist tailored to their taste, complete with the means to enjoy it. This represents a meaningful step forward in personalized music technology and showcases the creative potential unlocked when advanced AI models are guided by human-designed prompts. We envision that such prompt-driven AI DJs could enhance music discovery and enjoyment for users worldwide, and the techniques applied here can inspire similar innovations in other creative domains.

References

1. Gao et al. (2021). *Conversational Recommender System (definition).* [1]
2. OpenAI (2023). *GPT-4 Technical Report (system message and tool use).* [15] [16]
3. OpenAI (2024). *GPT-4o Announcement (multimodal capabilities).* [19] [12]
4. The Decoder (2023). *GPT-4 architecture rumor (MoE, 1.76T parameters).* [5] [6]
5. OpenAI (2023). *GPT-4 Training and Alignment (transformer, RLHF).* [4]
6. Spotify Newsroom (2024). *AI Playlist beta – user prompt example.* [3]
7. IBM (2023). *ReAct Framework for LLMs (reasoning + acting).* [17]

---

[1] GitHub - Zilize/CRSPapers: Conversational Recommender System (CRS) paper list. 对话推荐系统论文列表
https://github.com/Zilize/CRSPapers

[2] [4] [9] [10] [11] [14] [15] [16] [19] GPT-4 - Wikipedia
https://en.wikipedia.org/wiki/GPT-4

[3] Spotify Premium Users Can Now Turn Any Idea Into a Personalized Playlist With AI Playlist in Beta — Spotify
https://newsroom.spotify.com/2024-04-07/spotify-premium-users-can-now-turn-any-idea-into-a-personalized-playlist-with-ai-playlist-in-beta/

[5] [6] [7] [8] GPT-4 architecture, datasets, costs and more leaked
https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

[12] [13] GPT-4o - Wikipedia
https://en.wikipedia.org/wiki/GPT-4o

[17] [18] What is a ReAct Agent? | IBM
https://www.ibm.com/think/topics/react-agent