ChatGPT

Miles AI Tweet Processor – Product Requirements Document

Background & Motivation

Miles Deutscher is a prominent crypto analyst and influencer, known for his engaging Twitter threads and educational content on YouTube [1] [2]. With over 600,000 followers on X (Twitter) and a reputation for breaking down complex crypto concepts for his audience [1], Miles has a distinct writing style and persona. He emphasizes education over hype, aiming to provide well-researched insights to his followers [3]. For example, Miles often posts long Twitter threads (sometimes 25+ tweets) to explain topics in depth, and he frequently reminds readers that his content is not financial advice [4] [5].

Motivation: Given Miles's large audience and unique voice, there is a need to scale up his content output on X without sacrificing authenticity or quality. Recent demonstrations (e.g. by AI expert Harper Carroll) have shown it's possible to fine-tune an AI model on a person's own tweets to make it *"tweet exactly like you"* [6]. Inspired by this, the **Miles AI Tweet Processor** is envisioned as a tool to capture Miles's digital essence and expedite the process of creating tweets (or tweet threads) in his style. In essence, this product will allow Miles or his team to paste in raw text (ideas, news, analysis) and receive a polished tweet or thread that **reads as if Miles wrote it** – maintaining his tone, phrasing, and persona. This will enable ramping up daily posts and engaging content for his followers, while saving time and ensuring consistency with Miles's brand.

Objectives and Goals

- **Authentic Style Emulation:** Develop an AI-powered system that can **perfectly emulate Miles Deutscher's tweeting style and persona** in generated content. The output should be indistinguishable from Miles's own tweets, mirroring his tone, vocabulary, and formatting nuances.
- **Streamlined Content Creation:** Provide a simple interface for Miles or his content team to **paste or input text** (e.g. news snippets, rough analysis, or bullet points) and quickly obtain a well-crafted tweet or Twitter thread in Miles's voice. This drastically reduces the manual effort and time required to compose tweets, allowing more frequent posting and ideation.
- **Preserve Brand and Credibility:** Ensure the AI-generated tweets **maintain the trust and credibility** Miles has with his audience. The content should adhere to Miles's principles of educating rather than simply hyping [3], include disclaimers or cautious wording when appropriate (e.g. "not financial advice" reminders [4]), and engage the audience in a manner consistent with his past interactions.
- **Scalability:** Enable Miles's team to **ramp up daily posts** reliably. The system should be able to handle multiple generations per day and adapt to various topics (crypto market updates, project analyses, personal updates) while staying on-brand. Over time, the model can be retrained or updated with new data so it evolves with Miles's style and current events.

User Scenario & Use Case

Primary User: Miles Deutscher himself, or his content management team. They have knowledge of the subject matter (crypto, DeFi, AI topics) and possess raw insights or data they want to share. However, they want those insights presented in Miles's familiar Twitter voice to maximize engagement and consistency.

Use Case Example:
- A team member has drafted a quick analysis of a recent crypto market dip and some top altcoin picks, but it's written in a plain, generic tone. Using the Miles AI Tweet Processor, they paste the analysis into the tool. They select the option for a Twitter thread, since the content has multiple points. The tool then generates a threaded tweet series in Miles's style: the first tweet might start with an attention-grabbing hook and a confident statement about the dip (just as Miles often does), followed by numbered tweets breaking down each altcoin pick with concise explanations, and a concluding tweet that invites the audience's opinion (e.g. "Which coins are you watching?") and perhaps a friendly reminder ("Remember, this isn't financial advice – just my personal take!"). The language, formatting, and tone all sound exactly like something Miles would post. The team reviews the output, finds it on-point, and can then directly publish the thread to Miles's X account with minimal editing.

Another Scenario:
- Miles is about to release a new YouTube video on "Top 5 AI Robotics Crypto Projects". He wants to post a teaser tweet to drive viewers to the video. He enters a brief note into the tool: *"DePAI robotics narrative heating up, my top picks, huge opportunity, include link to video."* The AI outputs a tweet that Miles could have written himself, for example:

"The Decentralized AI (DePAI) robotics narrative is about to take center stage. I'm very *bullish on this sector – it could be the next big trend in coming months. Just dropped a new video with my top 5 picks to ride this wave. If you're positioning early, this is a must-watch☞ [YouTube link]"*

This output is formatted with Miles's typical line breaks, an emoji to convey excitement, a confident yet measured tone, and a call-to-action with the link – matching how Miles usually teases his video content 【20†】. Miles can then post this directly, knowing it sounds like his genuine voice.

The above scenarios show how the tool will be used to **transform raw input into Miles-styled content**, enabling him to engage his audience more frequently without individually crafting every tweet from scratch.

Scope

In Scope:
- Developing the AI model (or pipeline) that learns Miles's writing style from his past content (Twitter and YouTube).
- Creating a user interface (web or desktop app, or even a simple command-line tool initially) where users can input text and obtain the styled tweet output.
- Integration with X (Twitter) API for posting or scheduling tweets is a desirable feature for convenience (so that after generation, the content can be posted with one click).
- Content filtering or review step to ensure the AI output is appropriate and on-brand before posting (an

approval workflow).
- Ongoing training updates: the ability to feed new Miles content into the model periodically to refine or adjust the style mimicry as needed.

Out of Scope (for initial version):
- Generating content **without** user-provided input (the tool is not meant to autonomously come up with wholly new topics or ideas – it's focused on rewriting/formatting given ideas).
- Emulating **other** platforms or formats (e.g. long-form articles, direct video scripts) – however, the model's knowledge from YouTube transcripts will be used to inform tone on Twitter, but the product won't directly generate YouTube scripts in this version.
- Real-time trend analysis or suggesting topics to tweet – the tool will not itself decide what Miles should tweet about; it assumes the user provides the base content or topic.
- Fully automated tweeting without human oversight – while we aim to streamline posting, initially a human should review each AI-generated tweet (to ensure factual accuracy and appropriateness) before it goes live.

Data Sources & Training Strategy

To **capture Miles's digital essence**, we will leverage two primary data sources that comprehensively represent his communication style:

- **Twitter Data:** A complete archive of Miles Deutscher's tweets and threads on X (Twitter). *Status:* We have scraped approximately **39,000** of Miles's tweets (including both standalone tweets and threads, as well as replies) from his account history (2013–2025) as JSON data. This includes his famous multi-tweet threads, one-liner insights, casual "gm" posts, engagement questions, and any repeated phrases or emoji usage patterns. This dataset will teach the AI model the **format, tone, and brevity** of Miles's Twitter communications. It captures how he structures threads (often with numbering or " " thread emojis [5]), how he uses crypto terminology (e.g. tickers like $BTC, slang like "hodl" or "100x"), and his habit of injecting personality (emojis, informal greetings, etc.).

- **YouTube Transcripts:** A collection of transcripts from Miles's YouTube channel (Miles Deutscher Finance) covering all videos up to Aug 3, 2025 (over 200K subscribers on YouTube【12†】). These transcripts provide insight into Miles's spoken tone, explanatory style, and the content topics he covers in long form. While tweets are concise, the transcripts reveal **key phrases and the depth of explanation** Miles uses when discussing topics. For instance, Miles often emphasizes not being fooled by market dips and highlights "huge opportunities" with a balanced perspective【12†】. Such transcripts will help ensure the AI understands context and can maintain factual accuracy or reasoning in tweets (especially for threads that require a logical flow). However, the training process will be careful to align the style more with his *writing* style on Twitter (which is punchier), using the YouTube data mainly to enrich the model's knowledge of his perspective and common themes.

Training Approach: We will follow the blueprint demonstrated by Harper Carroll's reel, which showed how fine-tuning an AI on one's own tweets yields authentic results [6] [7]. The plan is as follows:

1. **Data Preparation:**
2. Clean and preprocess the tweet dataset. This involves removing any retweets or quotes that are not Miles's own words, filtering out tweets that are pure hyperlinks or very context-specific (if not useful

for style learning), and possibly splitting long threads into individual tweet examples for training. Emoji and punctuation will be preserved, as they are part of his style. Each tweet or thread segment will be treated as a training sample. We will also label or separate the tweet text from any metadata.

3. Prepare training pairs if needed. Since our goal is style emulation (not question-answering), we can fine-tune in a **Language Modeling** fashion: feed the model Miles's tweets so it learns to *generate text in that style*. Alternatively, we might create an instruct dataset where the "instruction" is a generic prompt like "Tweet about `<topic>` in Miles's style" and the "response" is an actual Miles tweet about that topic (if we find such pairs). However, an unsupervised fine-tune on all his tweets might be sufficient for style imitation [8].

4. Integrate YouTube transcript data selectively. We may extract key sentences or typical phrases Miles uses in videos that would also make sense on Twitter (for example, how he phrases warnings or excitement). These can augment the training data so the model has a broader sense of his lexicon. We will be cautious to avoid the model becoming too verbose from transcripts; tweet-style brevity is priority.

5. **Fine-Tuning the Model:**

6. Choose a suitable base language model for fine-tuning. Options include OpenAI's GPT-3.5 Turbo or GPT-4 (via fine-tuning API if available by 2025) or an open-source LLM like Llama-2 or GPT-J that can be fine-tuned on our data. The model should be large enough to capture nuances (likely 6B parameters or above) and support the short-form output length we need.

7. Perform the fine-tuning using the prepared dataset. This process will involve feeding the model the tweets (and selected transcript segments) so it adjusts its weights to produce outputs similar in style. According to Harper Carroll's experience, fine-tuning on personal tweets significantly improves the mimicry of tone and quirks compared to a standard model [8]. We expect the same: after fine-tuning, the AI should produce content that has Miles's signature *feel*, which a generic model previously could not replicate well.

8. **Validation during training:** We will set aside a portion of Miles's tweets as a validation set. Throughout training, we'll monitor how well the model generates Miles-like text for unseen examples. We'll avoid overfitting (so it doesn't memorize exact tweets, but rather the style). The success criterion is that the model's outputs are fluent and *sound like Miles*, even for new topics or combinations of ideas not explicitly in the training set.

9. **Testing & Iteration:**

10. After fine-tuning, we will rigorously test the model. This includes **side-by-side comparisons** with a baseline model as demonstrated in the reel [7]. For example, we'll prompt both the fine-tuned model and an untrained model with the same query (like "Tweet about Ethereum's latest upgrade in Miles's style") and observe the difference. We expect the fine-tuned model to use language and formatting that closely matches Miles's past tweets, whereas the baseline would be more generic.

11. We'll generate sample tweets on various topics (market updates, educational threads, casual greetings, etc.) and have Miles's team verify that the voice is accurate. If some aspects are off (too formal, or incorrect jargon usage), we will adjust the training (e.g. by refining the dataset or using prompt techniques in conjunction with the fine-tune). The goal is **surgical precision** in capturing his voice – meaning the details like his frequent use of certain emojis or phrases should appear naturally

in the AI outputs. For instance, if Miles often starts the day with “Gm.” or uses a “ ” to mark achievements【20†】, the model should learn this and replicate it when contextually appropriate.

12. **Deployment of the Model:**

13. Depending on the model choice, deployment might involve hosting the model on a cloud service or calling an API. If using OpenAI’s fine-tuned model, we’ll deploy via API integration (ensuring we handle rate limits and costs for each call). If using a local model, we’ll host it on a server or locally on a machine that the team can run, ensuring inference time per tweet is quick (ideally just a few seconds).
14. The final fine-tuned model will be locked for the initial launch of the product. Future updates (re-training) can be planned every few months or when Miles’s style shifts or he wants to add new vocabulary (for example, if a new crypto term becomes popular and Miles adopts it, we’d want the model to learn that).

Functional Requirements

1. User Interface & Input/Output

- **Text Input Field:** The product shall provide a clear text box where the user can paste or type the content that needs to be “Miles-ified.” This could be a rough draft of a tweet, a list of points, or even just a topic phrase. The interface should accommodate at least a few hundred characters of input (to allow multiple bullet points or a short paragraph if the user is describing a thread).
- **Generate Tweet Button:** There will be a prominent action (a button labeled e.g. “Generate Tweet”) which, when clicked, sends the input to the AI model and returns a tweet or thread in Miles’s style.
- **Output Display:** The resulting tweet text (or thread) should be displayed in an easy-to-read format, mimicking how it would appear on Twitter. For threads, each tweet could be shown on a new line or separate card, numbered or connected with a line to indicate sequence. The output area should preserve the formatting the model produces (line breaks, emojis, punctuation). **Citations or references** (if any were in the input) should be handled carefully or removed, as Miles’s tweets typically don’t include formal citations.
- **Copy & Edit Options:** The interface will allow the user to quickly **copy the generated text** to clipboard for manual posting on X if desired. Additionally, the user can make minor edits in the output field if they want to tweak phrasing. (For example, if the AI writes “In my opinion…” but Miles/team wants to remove first-person, they can edit it before posting.)

- **Regeneration & Variations:** There should be an option to “Regenerate” the tweet if the output isn’t satisfactory or if the user wants an alternate take. Because AI outputs can vary, this allows getting multiple phrasing options for the same input. We may also allow a setting for the user to specify the **tone or format** if needed (for instance, a checkbox for “Make this a thread” or “Include an engaging question at the end”). By default, the system should intelligently decide if a thread is needed based on input length, but giving the user some control ensures flexibility.

- **Thread Mode Handling:** If the input text contains multiple points or is long, the system should consider generating a thread. In thread mode, the AI should split content into coherent tweet-sized chunks, each with a logical breakpoint. It should also automatically prepend numbering or a thread emoji if that’s part of Miles’s typical thread format. *Example:* For an input with distinct bullet points,

the output might be:
Tweet 1: "1. Thinking in bets – Treat every trade or investment as a probability-based decision..."【21†】
Tweet 2: "2. Diversify your conviction – Even your best idea can fail, so..." and so on.
Each tweet in the thread should still sound complete and in Miles's voice, and the sequence should flow logically. The tool should ensure the thread doesn't exceed Twitter's thread length limits (if any) and that each tweet stays within the character limit (280 chars for standard tweets, or 4k if extended tweets are available to Miles's account).

- **Image/Media Attachment (Future):** If Miles often attaches images (charts, event photos, etc.), the PRD notes this as a future enhancement: allow the user to attach an image and have the AI incorporate a reference to it in the tweet (e.g. "(see chart below)" or appropriate alt text). However, for v1, handling text content is the priority; media integration can be manual.

2. Style Emulation & Content Quality

The core functional requirement is that **the output text must capture Miles's persona and style with high fidelity.** Key style elements derived from analysis of Miles's content:

- **Educational yet Conversational Tone:** Miles's tweets strike a balance between *informative* and *approachable*. The AI should mirror his objective, fact-driven commentary style [9] while maintaining a conversational tone (using first person "I" when appropriate, or "you" to address the audience, which Miles does frequently). Overly formal or verbose language is not desired – the phrasing should be **crisp and accessible** as in Miles's own threads [1] .
- **Brevity with Impact:** Each tweet (especially in threads) should usually be concise and to the point. Miles often uses short sentences or sentence fragments separated by line breaks for emphasis. The AI should use line breaks to create a logical and impactful flow, similar to how Miles formats his tweets for readability (often one or two sentences per paragraph). The model should avoid run-on sentences or dense paragraphs.
- **Emojis and Slang:** Incorporation of emojis and crypto slang should reflect Miles's actual usage patterns. For example:
- Miles uses emojis like (check mark for accomplishments or status updates)【20†】, (handshake, when meeting people or greeting)【20†】, (drooling, for food or exciting news)【20†】, ☞ (pointing finger for call-to-action links)【27†】, etc. The AI should insert such emojis in similar contexts (but not overuse them).
- Common crypto terms in Miles's vocabulary (from tweets and transcripts) include *"bullish"*, *"narrative"*, *"100x"*, *"alpha"*, *"DeFi"*, *"hodl"*, etc. The AI should feel free to use these terms when relevant, as Miles's audience is familiar with them. It should also use ticker symbols with a dollar sign for major coins (e.g. $BTC, $ETH) as Miles does【27†】.
- **Engagement and Persona:** Miles often engages his audience with questions or invites their opinions. The AI should replicate this by occasionally ending a tweet (or thread) with a question or prompt when the context suits it. *For instance*, after sharing his "3 top altcoins," Miles asked followers "what would yours be?"【34†】. The tool should know to include a similar audience prompt if the input suggests a list or personal picks, etc. Likewise, Miles uses inclusive language like "we" or speaks directly to the audience ("here's what you need to know..."). These rhetorical techniques should be present in the AI's outputs to keep the **community feel**.

- **Disclaimers & Caution:** To maintain credibility, the AI must include **Miles's cautionary tone** when appropriate. Miles is careful not to promise financial gains – often reminding that a thread is *"purely based on personal opinion and not financial advice."*【22†】 When the input is about price predictions, investment picks, or anything that could be construed as financial advice, the output should contain a brief disclaimer or at least temper statements with phrases Miles uses (e.g. "I believe…", "IMO," "not financial advice," or "do your own research"). This ensures the AI's tweets won't inadvertently sound like outright financial directives that Miles himself would avoid.
- **Consistency with Facts/Context:** The AI should use the information from the input accurately and **not hallucinate facts**. If the input says "Project X just partnered with Y," the output should not invent additional partnership details not provided. It should express an opinion on it in Miles's style, perhaps referencing similar past events if known (drawing from his trained knowledge), but it **must not introduce false information**. Maintaining Miles's reputation for well-researched content [9] is crucial. This may mean the model, if unsure, uses conditional language ("If X succeeds, it could…") rather than confident false statements.

- **Thread Cohesion:** For multi-tweet outputs, ensure that the **thread has a logical flow**. The model should ideally produce an outline of points that make sense in sequence (Miles's real threads often enumerate points or walk the reader through a narrative or argument). The tool should avoid repeating the same point in different words or going off on an unrelated tangent mid-thread. Each tweet in the thread should build on the previous or cover a next point. We might enforce this by guiding the model during generation (e.g. using the input structure or bullet points to map the thread outline).

- **Personal Touch and Updates:** Miles occasionally tweets about personal experiences (moving to Dubai, attending conferences, daily greetings). The AI should be capable of switching to a **more casual, first-person tone** when the input is of that nature. For example, if the user inputs something like "Met with followers at Token2049 conference, great experience!", the output might be: *"Met so many of you at Token2049 today – what an experience . The energy here in Dubai is incredible. Excited for Day 2!"* This sounds genuine and personal. While most usage will be for educational content, maintaining the ability to do these personal-style tweets ensures a well-rounded emulation of Miles (so the audience doesn't sense a sudden change in personality on certain posts).

3. Integration with X (Twitter) Platform

- **Twitter API Integration:** The product should integrate with Twitter's API (X API) to allow for seamless posting of generated tweets. After generating the content, a user who has authenticated Miles's Twitter account with the app can click "Post" to immediately publish the tweet or thread to his timeline. This removes the step of copy-pasting and logging into Twitter separately. The integration must use secure OAuth and comply with Twitter's automation rules (ensuring we do not violate rate limits or spam policies).
- **Scheduling Feature:** In addition to instant posting, the tool could offer scheduling. For instance, if Miles wants to queue up tweets for times he's not actively online (to maintain a global engagement), the interface can provide date/time pickers to schedule the tweet/thread. The system will then post at the designated time via the API. This helps *"ramp up daily posts"* by planning content in advance.
- **Draft Management:** The tool can keep a log of generated tweets. Not all generated content will be posted immediately; some might be saved as drafts for later. A draft management screen would list

recent generations with options to edit, post, or discard. This is useful if the team generates multiple tweet ideas in a brainstorming session – they can store them and decide later which to publish.

- **Collaboration:** (Optional future feature) If multiple team members use the tool, having a shared workspace (with version history or notes on each draft) could help collaboration. For v1, this may be as simple as not deleting outputs and allowing manual copy-edit, but later a multi-user support could be formalized.

- **Privacy & Security:** Since the tool will have access to Miles's Twitter account (for posting) and his proprietary content, we need to ensure data security. API keys and tokens should be stored securely (encrypted) and the application should only be accessible to authorized persons (Miles and his team). There should be a confirmation step before any tweet is posted (to avoid accidental or malicious posts – e.g., a stray click should not immediately broadcast without confirmation unless explicitly intended).

4. Performance & Reliability

- **Response Time:** The AI model should generate the tweet or thread output **within a few seconds** (ideally under 5 seconds for a single tweet, under 10 seconds for a thread). The user experience should be smooth – if generation is slow (due to large model), we will incorporate a loading indicator but aim to optimize for speed (via efficient model hosting or using a smaller fine-tuned model if necessary).
- **Uptime:** The system should be reliably available whenever the team needs to generate content. If using an external API (OpenAI, etc.), we must monitor for downtime or quota issues. If hosting ourselves, the server should have high uptime. In practice, tweets are often time-sensitive (commenting on breaking news or daily market moves), so the tool must be dependable at those moments.
- **Scalability:** While the primary user base is small (just Miles's team), the model should handle bursts of usage (e.g., generating 10 threads in an hour during a busy news day) without crashing or significantly slowing. The architecture should allow scaling the backend if needed (like containerization or cloud functions for generation on demand).
- **Error Handling:** In case the AI fails to generate a coherent result or returns an error, the UI should handle this gracefully. For example, if the model returns nothing or gibberish, the tool can show a message like "Generation failed, please try again" and log that event. We will implement basic validation on outputs – e.g., if the output is unusually long (beyond Twitter limits) or empty, the system will auto-regenerate or prompt the user to adjust input. Also, if the Twitter API post fails (due to network or an auth issue), the user should be notified and prompted to retry or re-authenticate.

Technical Implementation Details

(This section translates the above requirements into a plan that developers or an AI coding assistant like Claude can follow to build the system.)

- **Model and Fine-Tuning:** We will likely use OpenAI's fine-tuning capabilities on the GPT-3.5 Turbo model (which by 2025 supports fine-tuning for style adaptation), as it offers a good balance of capability and speed. Alternatively, a local LLaMA-2 model (~7B or 13B parameters) fine-tuned on our dataset with Low-Rank Adaptation (LoRA) is an option if we need on-premise processing. The fine-tuning process will follow the steps outlined in the Data & Training section. We should allocate time for experimentation to get the model output right (possibly fine-tune iteratively and evaluate).

- **Prompt Engineering:** In addition to model training, we will utilize prompt engineering for optimal results. For instance, when sending the input to the model, we might format it as: `"You are Miles Deutscher, a crypto analyst and influencer known for objective, educational tweets (not financial advice). Read the following input and rewrite it as a tweet in Miles's style:\n[User Input]\nTweet:"`
This system prompt primes the model to stay in character and only then the model generates the tweet. The fine-tuning will make this easier, but a well-crafted prompt can further reduce errors and ensure the model knows it **must use Miles's tone**. We will embed such a prompt in the backend generation code.
Additionally, if the user chooses "thread mode", the prompt can instruct the model accordingly, e.g.: `"Format the output as a Twitter thread with each point numbered."`

- **Application Architecture:** The product will have a front-end (possibly a simple web app) and a back-end. The back-end will handle calls to the AI model and to Twitter's API. We'll use a Python-based stack for ease of integrating machine learning (e.g. a Flask or FastAPI server to expose an endpoint for generation). The front-end could be a lightweight React or HTML/JS interface that communicates with the backend. Given this is an internal tool for now, we might even run it as a local app (with a simple GUI or Jupyter notebook interface) for initial testing. The Claude Code generation will be used to assist writing this code, following the PRD guidelines.

- **Steps to Execute (Development Tasks):**

- *Data ingestion:* Write a script to load the provided tweet JSON files and the YouTube transcript JSON. Transform these into the needed training format (e.g., a CSV or JSONL with either just the tweet text per line, or prompt-response pairs if doing instruct fine-tune). **(Status: data already scraped and available)】**.
- *Fine-tuning pipeline:* Set up the fine-tuning job (either through OpenAI API or using a library like HuggingFace's transformers for a local model). Run experiments and evaluate outputs on validation prompts. Adjust as necessary and finalize the model.
- *Backend implementation:* Develop the API that loads the fine-tuned model (or calls it externally). Implement the endpoint for "generate_tweet" which accepts input text and parameters (thread or single, etc), and returns the generated text. Ensure to handle splitting into multiple tweets if needed (the model might output a full thread as one string with separators, which we then split on "\n2." etc., or we generate iteratively tweet by tweet).
- *Twitter API integration:* Set up OAuth for Miles's account (the team will provide access tokens). Implement endpoints or functions for posting a tweet/thread. For threads, this means posting the first tweet, then replying with subsequent ones in order. Build in a check to ensure the tweet length is within limit (split if model overshoots, or truncate with an ellipsis and warning in UI).
- *Frontend UI:* Create a user-friendly interface. This could involve designing a form with an input text area, a few options (checkboxes or toggles for thread mode, etc.), and a generate button. After generation, show the output in a stylized way (maybe mimic a Twitter card layout for realism). Include buttons for "Copy text" and "Post to X". If posting, add confirmation modals. The UI should be clean, with Miles's branding subtly (maybe his profile pic or name to remind user this is Miles's voice generator).

- *Testing the end-to-end flow:* Use real examples from Miles's recent content to test. For instance, feed in the gist of a video or a news event and see if the tweet matches Miles's actual tweet (if one exists) or at least sounds coherent. Have testers who know Miles's style review several outputs. Fine-tune the prompt or retrain the model if issues are found (e.g., model still too generic in some cases).
- *Iteration and polish:* Once the core functionality is working, refine the UI/UX (make sure it's easy to use, maybe add a dark mode if Miles's team prefers, etc.), and tighten any loose ends (like ensuring the model doesn't accidentally produce content that Miles wouldn't – e.g., profanity or unrelated topics – if our dataset was clean this should not happen).

- **Execution of Harper's Video Steps:** To explicitly tie back to the Instagram reel that inspired this project [6], our implementation covers all the steps shown in that "Here's how" tutorial: we have collected tweets, we will fine-tune a model on those tweets, and we will compare the standard model's output to the fine-tuned model's output to confirm the improvement (just as Harper showed side-by-side [7]). The PRD has detailed these steps to ensure we follow that proven recipe. Fine-tuning the model is identified as the critical step that gives the AI the ability to *"tweet exactly like"* Miles [6]. We have also noted that fine-tuning is a relatively straightforward process with high impact on quality [8] – thus we have allocated effort to do it properly rather than relying only on prompts. In summary, every requirement needed to replicate the reel's outcome (data prep, fine-tune, style testing, integration) is captured above for the development team to implement with precision.

Evaluation Metrics

To determine success and ensure the output meets expectations, we will use the following metrics and tests:

- **Stylistic Accuracy (Human Evaluation):** Miles and a few close collaborators will review a set of AI-generated tweets (without knowing which are AI vs which are his past real tweets) to judge if they feel authentic. Success is defined by a high rate of "indistinguishable" ratings – i.e., the evaluators cannot reliably tell which content was human-written vs AI-written, or they explicitly say the AI tweets *sound exactly like Miles*. This qualitative check is the ultimate benchmark since the goal is to *perfectly emulate* his style.
- **Engagement Metrics:** Once some AI-generated tweets are posted on Miles's actual account (with his approval), we can compare their performance to historical tweets. We will look at metrics like average likes, retweets, replies, and engagement rate. If the AI tweets maintain similar or better engagement than typical Miles tweets on similar topics, that indicates the audience is finding them just as relevant and authentic. A significant drop in engagement or an increase in comments like "did someone else write this?" would be red flags to address. Our expectation, however, is that by keeping content quality high and on-brand, followers won't notice a difference except that Miles is posting more frequently (which should be a positive).
- **Linguistic Analysis:** We can perform some automated checks on style: compare the distribution of certain key phrases, emoji usage frequency, sentence length, etc., between the AI's outputs and Miles's real tweets (using the tweet dataset as a baseline). For example, if Miles starts 30% of his educational threads with a question, but the AI never does, we know to adjust. We might use embedding similarity or custom classifiers to ensure the AI outputs cluster closely with Miles's known content in vector space.

- **Error Rate:** Monitor how often the AI outputs require manual correction. For instance, does it ever produce a factual error or a weird phrasing that Miles wouldn't say? Each time the team has to edit an AI tweet, log the issue. If in daily use the team finds they almost never need to rewrite anything, that's a big success. Our aim is to minimize edits to maybe just tweaking for preference rather than fixing mistakes.
- **Throughput & Usage:** Track how many tweets/threads are being generated and posted via the tool per week. The goal of ramping up content can be quantified: e.g., if previously Miles posted 3-4 times a week and now with the tool he can comfortably post 10-12 times, that's a measurable win. We should ensure the system supports that volume without degradation.
- **Feedback from Audience:** Indirectly, we might gauge if followers respond well. Perhaps conduct a poll or gather a few trusted community members to see if they notice any changes in tone. Ideally, the feedback should just be about the content itself (which is as it should be), not about style changes. If someone who knows Miles's style can be shown an AI-generated tweet and they think Miles wrote it, that's confirmation of success.

Future Enhancements

Once the core system is in place and functioning, we can consider additional features and improvements:

- **Multi-Platform Expansion:** Extend the AI style emulation to other platforms. For example, generating LinkedIn posts or blog snippets in Miles's style, or summarizing his YouTube videos into Twitter threads automatically. Since we have YouTube transcripts, a future feature could take a full transcript of a new video and generate a corresponding Twitter thread summary to promote it. This would further streamline Miles's content production across channels.
- **Other Personas:** The framework could be reused to fine-tune models for other team members or influencers. In the future, "Miles AI Tweet Processor" could become a more general "Persona Tweet Processor" where new datasets can be plugged in. Our success with Miles can be a case study for potentially offering similar AI styling for other content creators (with their consent).
- **Real-time Trend Integration:** Incorporate real-time data feeds (e.g., crypto market data or news) so that the AI can automatically include the latest numbers or trending topics in the generated tweets. For instance, if Bitcoin price is surging, the tool might automatically include the price or percentage change if relevant to the input. This would require API integrations and ensuring the model or a post-processing step can handle factual inserts.
- **Sentiment and Risk Controls:** Implement more controls on the tone – e.g., if Miles wants to deliberately make a tweet more optimistic or more cautionary, an option could adjust that. Similarly, a "compliance" filter could be added to double-check that no output could inadvertently violate regulations (since crypto is a sensitive area) – e.g., not making explicit investment promises, etc. Currently we handle that by training and style, but additional rule-based checks might be good as volume scales.
- **Continuous Learning:** Set up a mechanism where any tweet that Miles manually writes (or heavily edits from the AI output) can be fed back into the training data to continually improve the model. Essentially, the model could periodically retrain on the latest content to stay up to date. We must, however, monitor for any drift – if Miles's style changes or he explores new topics like AI more (as hinted by him being "Obsessed with AI" [4]), the model should adapt to include those without forgetting the core style.
- **UI Improvements:** In the interface, we might add features like highlighting which parts of the output came from which part of the input (to build trust that the content wasn't hallucinated). Also,

could implement a comparison view to show the user "Standard GPT vs MilesGPT" outputs side by side – not typically needed for end user, but could be a transparency feature or even a marketing angle to show how much closer the Miles-tuned model is.

Conclusion

The Miles AI Tweet Processor will be a cutting-edge tool that encapsulates Miles Deutscher's online voice and allows his team to produce content with **unprecedented efficiency and consistency**. By training on his extensive history of tweets and videos, and by following the proven method of fine-tuning an AI model on personal data [6], we ensure that every generated tweet **sounds authentically "Miles"** – from the choice of words to the use of emojis and the overall vibe of the message. This PRD has detailed the requirements to implement this system, including data handling, model training, UI/UX, integration, and safeguards to maintain quality. We have drawn insights from Miles's own content and the inspirational case study of an AI tweeting exactly like its creator [6] [7] to inform these specifications.

By adhering to these requirements with *surgical precision*, the development team (with the help of AI coding assistants like Claude) can confidently build a solution that meets the desired outcomes. Success will mean Miles can **amplify his presence on X** without diluting his personal touch – effectively letting him be in multiple places at once (or multiple tweets per day) while his audience receives the same value and authenticity they expect. In the rapidly evolving social media landscape, this project will demonstrate the power of combining human creativity and AI to expand one's voice responsibly and efficiently.

Sources:

- Harper Carroll, *"POV: You taught an AI to tweet exactly like you – & you can do it too. Here's how…"* (June 26, 2025) – Demonstration of fine-tuning a model on personal tweets [6] [7].
- Miles Deutscher's X (Twitter) Profile – Bio and follower information [2] [4].
- *Fantom Community Spotlight – Miles Deutscher* (Interview, Feb 2022) – Background on Miles's philosophy of educating through his platform [3] [5].
- Press Release via Yahoo Finance – Note on Miles's well-known Twitter threads for breaking down complex concepts [1].
- Sample Miles Deutscher Tweets (2023–2025) – e.g., engaging the audience with questions and using his characteristic formatting【34†】【27†】. These examples inform the stylistic requirements of the AI output.

---

[1] Cryptocurrency Investor and Analyst Miles Deutscher Joins Crypto ...
https://finance.yahoo.com/news/cryptocurrency-investor-analyst-miles-deutscher-230000526.html

[2] Miles Deutscher (@miles_deutscher) • Instagram photos and videos
https://www.instagram.com/miles_deutscher/?hl=en

[3] [5] [9] Fantom Community Spotlight – Miles Deutscher
https://blog.fantom.foundation/fantom-community-spotlight-miles-deutscher/

[4] Miles Deutscher (@milesdeutscher) / X
https://x.com/milesdeutscher?lang=en

6 aki - X
https://x.com/Aki_laid_back/status/1947085043253588269

7 POV: You taught an AI to tweet exactly like you, & you can do it too ...
https://www.youtube.com/shorts/8jHoqqQJK7s

8 Fine tuning is so simple, yet so effective, for improving your ChatGPT...
https://www.tiktok.com/@willfrancis24/video/7524737624599203094?lang=en