Cerebros NotGPT UX Wireframe & Workflow

David Thrower

Magic Patterns Links to [Partial] Live Wireframes

- https://www.magicpatterns.com/c/i1pf5qb61sidz82fkkksxs
- https://www.magicpatterns.com/c/jxegap21qrbhtwg761ycbd

Intro: The Company

- Cerebros NotGPT is a platform that provides customized AI assistants for enterprise users.
- The functionality of our offering is similar to ubiquitous AI assistant and AI inference providers such as:
  - Open AI
  - Mistral
  - Perplexity
  - Anthropic
- However, our key differentiator is user personalization: We developed frontier AI technology to train custom foundation models for individual users.
- Because our model architecture trains in linear / O(n) complexity timing, not O(n^2), we can train a custom foundation model, instead of merely steering a generic model to passively act as though it is personalized (e.g. RAG, LoRA).

Intro: The Problem:

- Workflow inefficiency:
  - I write back and forth all day with my assistant to get a document written.
  - I endlessly add nuances details, clarifications ..., and eventually I get back a correct and complete work.
  - By then I may as well have just written it myself de novo, without AI.
- Security, legal privilege, and cost:
  - More efficient model architecture = practical to run on prem.
  - Break your addiction to expensive cloud inference providers.
- The environment and economics:
  - Reduced carbon footprint.
  - Reduced strain of the energy grid.
  - Reduced water supply depletion.

Intro: The Solution:

- In summary: A personalization on-boarding to upload your data, then builds an AI assistant created for you, one that is fully personalized from day 1.
- **You upload examples of**
  - Your work (Documents and papers you write)
  - Questions people may ask you at work / Questions people may ask you … and their answers
  - Example conversations (Email threads, Slack, Discord, SMS)
  - References: Any resources you use to do your job:
    - Reference books
    - rules,
    - procedures,
    - research articles
    - Templates
    - …
- **We personalize an AI assistant for you in an automated 5 step pipeline.**
- **You use your personalized assistant:**
  - **You provide short, high level instructions and get back detailed, nuanced, work - environment - aware responses without detailed instructions or protracted iterative clarifications.**

Incomplete UIs and workflows yet to be defined

- Ecommerce website / User sign up (select a subscription option, pay, have resources provisioned)
- Workflow for on-going model updates after assistant deployment
- Billing settings / resource quotas / change subscription

MVP Constraints:

- 1024 Total context window, maybe 3000: TBD as we are scaling the model up and determining exact model training costs and hopefully have funding to work with.
- Training limited to around 70,000 - 150,000 samples
- SImple UI workflow lacking many advanced features
- Single iteration of model development for now, ongoing fine tuning loop for the model to be introduced in the next revision.
- Hopefully we can include basic web search capabilities, but may be a partially manual workaround (Like add a field to web search a specific phrase instead of LLM tool calling to determine the search terms and execute the search) …
- A proof of concept for personalization, a 500M - 750M param model to show a commercially useful, but small use case.
- Parameterized such that we can simply increase the sequence length, increase the number of samples, … and run a bigger task once the money is there to justify further scale up.

Custom LLM Training Data Set Composition

Training Stage	Corpus Type	(approx) Number of Text Samples	Purpose / Goal
Stage I-a (Neural Architecture Search)	**Synthetic Literature & K-12 textbook**	**2k–10k**	Foundational English, rapid prototyping (CEFR A1)
Stage I-b	**Synthetic Literature & K-12 textbook**	**20k–60k**	Expanded language fluency (CEFR A2)
Stage II	**Synthetic Social, web, and news**	**20k–50k**	Real-world, dialog fluency (CEFR B2)
Stage III	**Synthetic business, legal, medical, and professional documents, research, references / wiki**	**10k–15k**	Professional, critical thinking (CEFR C1 English Fluency)
Stage IV	**Instruction following (generic)**	**10k–30k**	Task prompting, basic agent behavior
Stage V	**Instruction following (User Defined)**	**5k–15k**	Nuanced personalization; Awareness of User's organizational role.

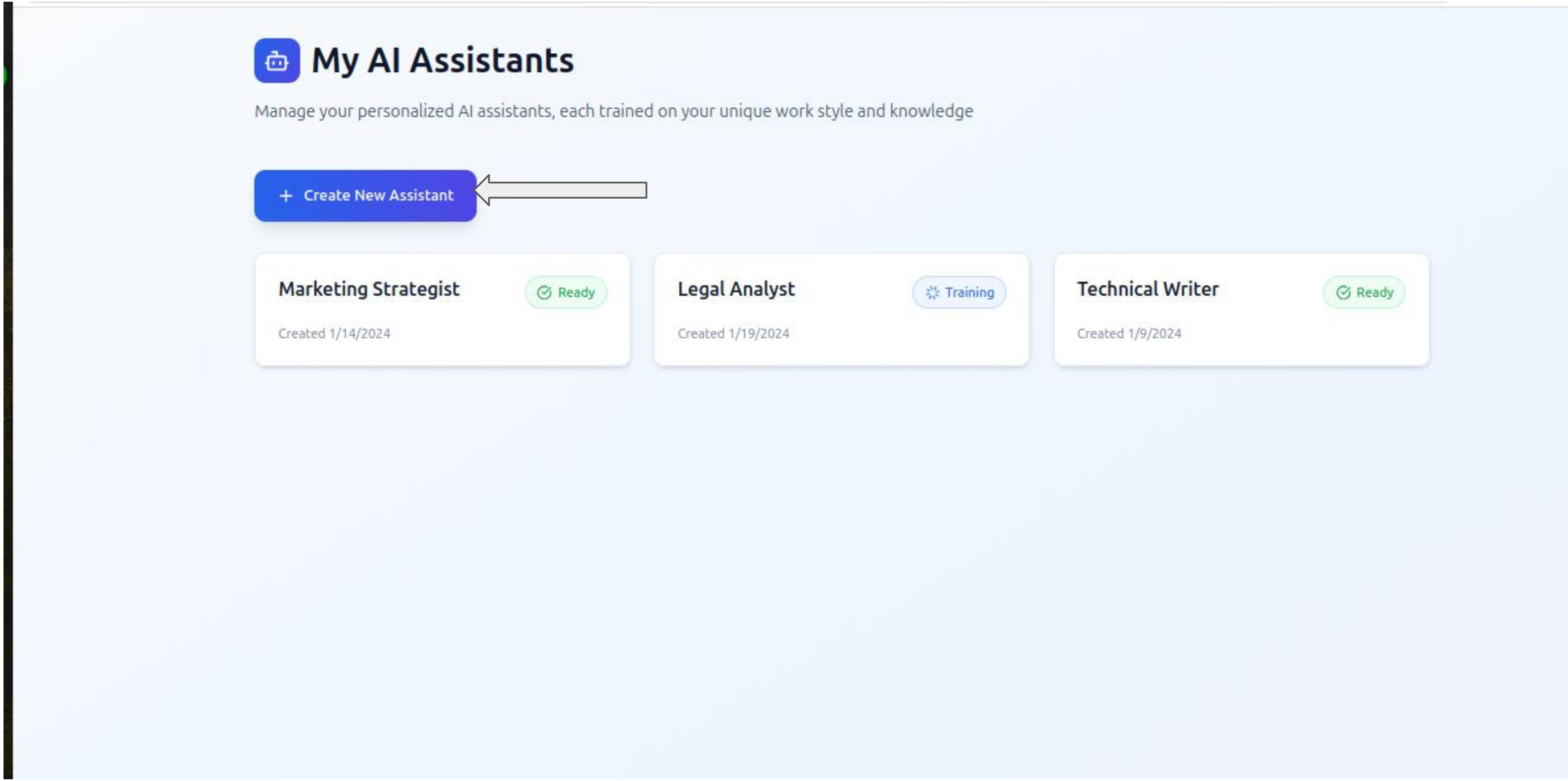
Splash page

[My Assistants] <--<<

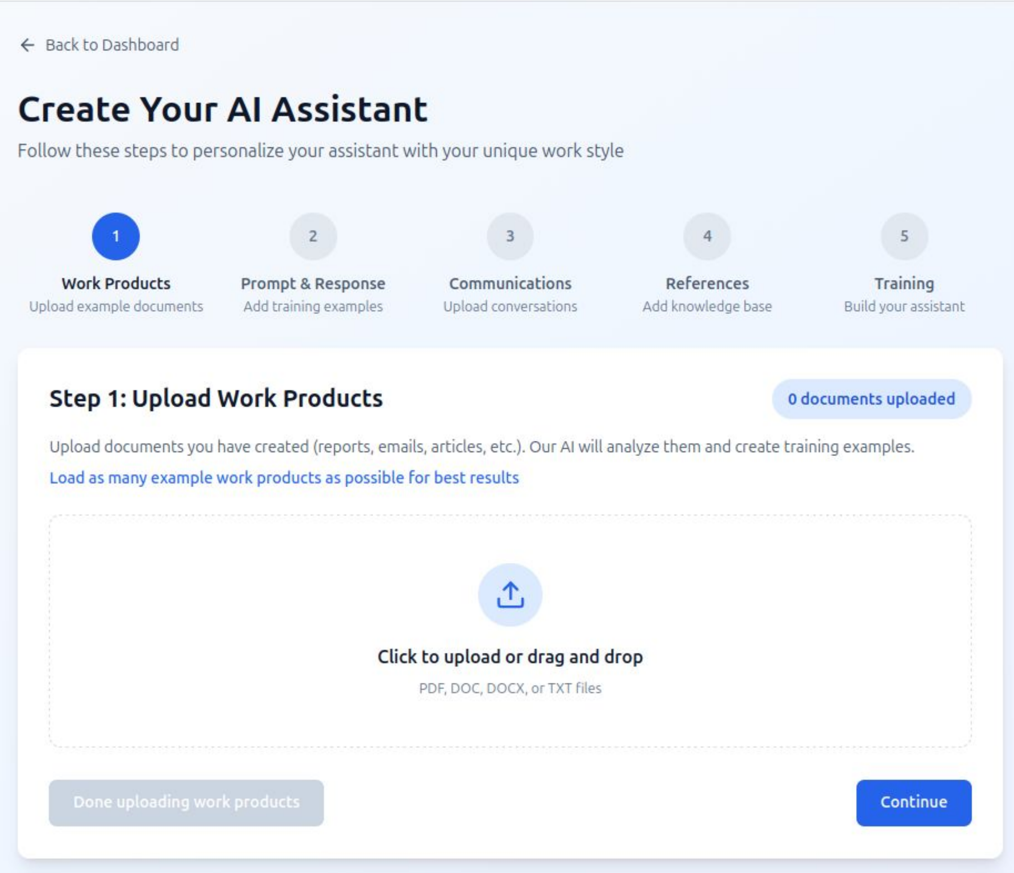
[New Assistant]

[Billing Settings]

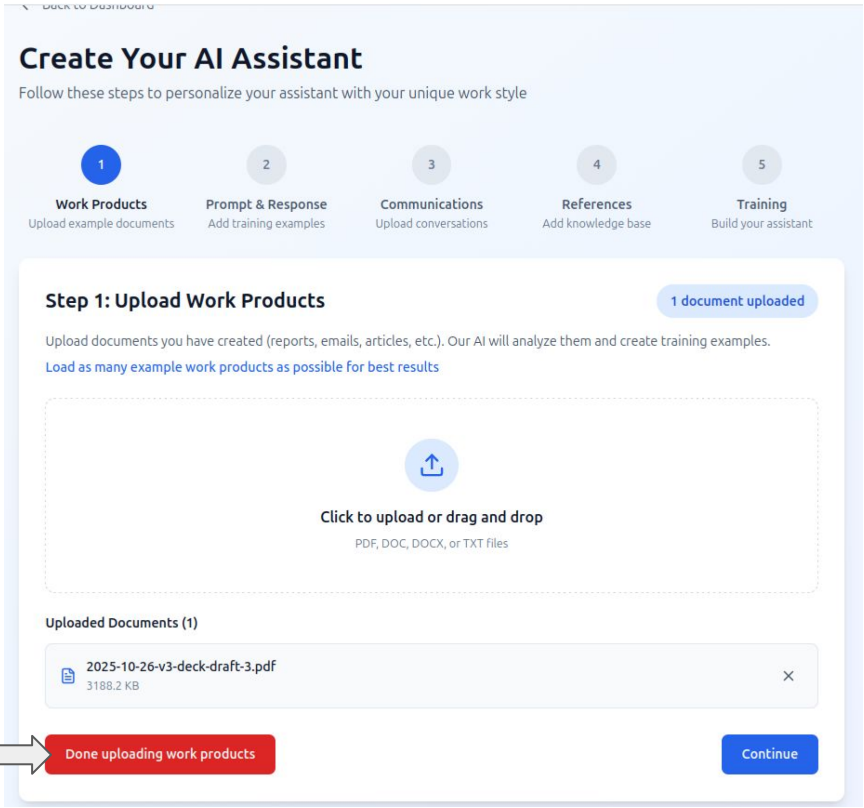
My Assistants / Select an Assistant



Create New Assistant: Step 1 (a):



Create New Assistant: Step 1 (b):



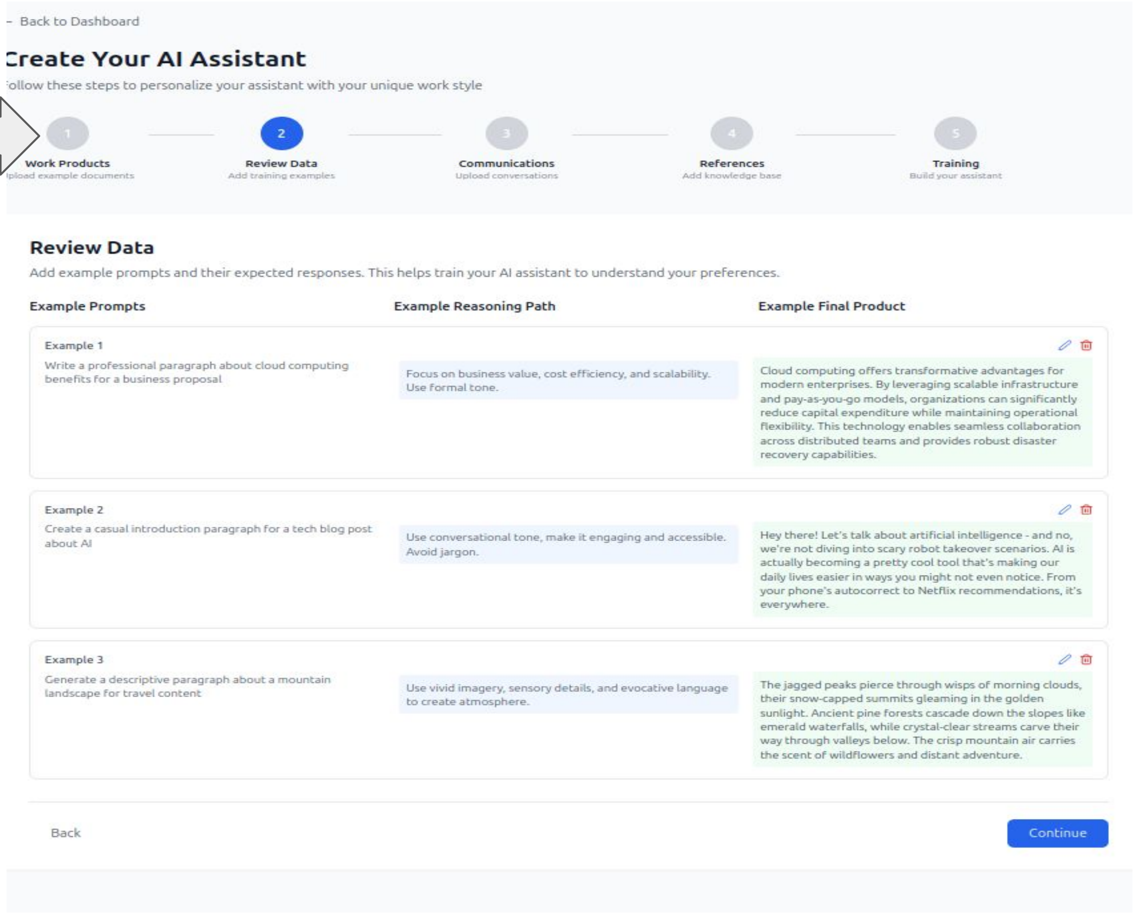
Should be still greyed out until after next step

First Pass back - end logic: (Automated Data Labeling & Upsampling)

1. Take the uploaded example “work product” the user uploaded.
2. A suitable system prompt and a call to a generic LLM inference API generates:
   a. An example of a prompt which the user may write if they wanted their assistant to generate that “work product”.
   b. A reasoning pathway for creating that work product.
   c. may do some summarization of the original work product to fit context length limits.
3. Hard coded logic packages the work product and prompt as a string “<prompt>[prompt it generated]</prompt><think>[reasoning pathway]</think><response>[the work product]</response>”
4. Another suitable system prompt creates a new synthetic example “work product” similar to the “work product” example the user uploaded.
   a. (2) and (3) is repeated using the result of (4) as the input.
5. Repeat until 5 - 20 synthetic samples are created.

Create New Assistant: Step 1 (c): User Reviews Synthetic Data

This one should be the one toggled … a defect in the UI mock - up.



Repeat ….

User directed to Step 1 (a) to upload more example work products or to click that they are done uploading work products.

Step 1: Data Relation & Outputs

Samples packaged in a flat format (CSV, SQL, or dataset committed to MlFlow)

User uploaded example documents

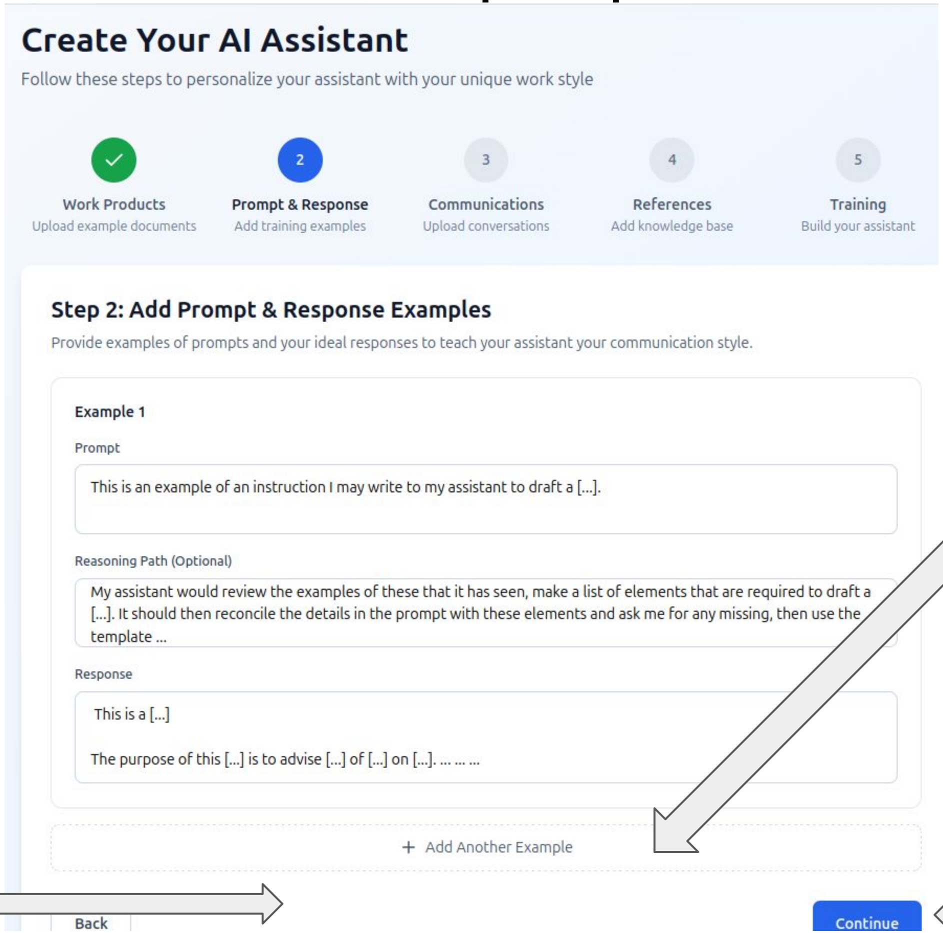
- Source Sample Number (Primary key)
- Original sample text extracted from the documents, as uploaded

User Example Work Product instruct samples augmented

- Sample number (primary key) (after synthetic up-sampling and permuting prompts and responses)
- Source sample ID (foreign key) (before up-sampling)
- Sample text: “<prompt>...</prompt><think>..</think><response>...</response>”

** Additional metadata, version tags, etc TBD

Step 2 (a): User adds example questions and answers



Remove, should return you to this page after "review sample page"

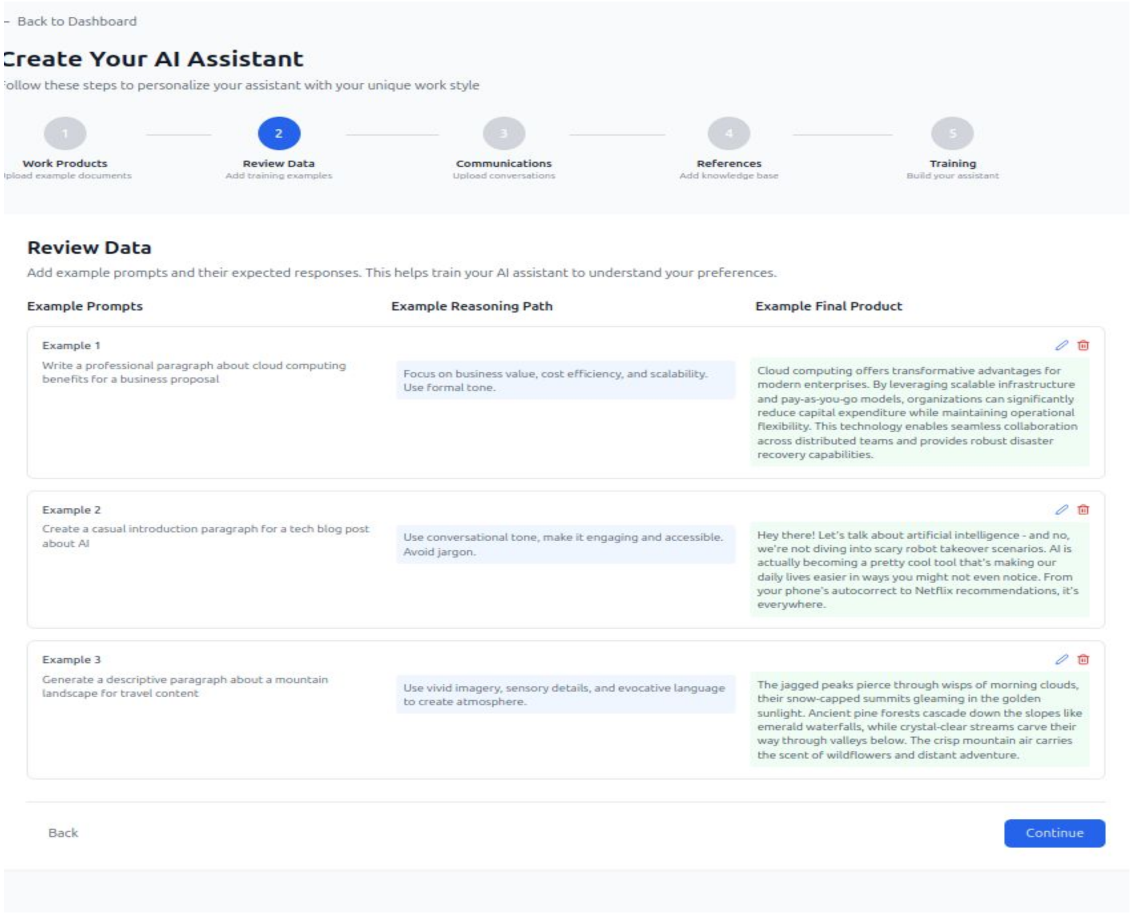
Add button [Done adding example prompts]

Replace caption with [Review samples]

First Pass back - end logic: Upsampling of Instruct samples (Questions and Answers)

1. Take the uploaded example
   a. “Prompt”,
   b. “Reasoning pathway”,
   c. “Response” the user uploaded.
2. 3 suitable system prompts and 3 separate calls to a generic LLM inference API generates a synthetic sample of each.
3. Repeat … until we have 5 - 20 synthetic examples…

Step 2 (b): Example questions and answers



Step 2: Data Relations & Outputs

Samples packaged in a flat format (CSV, SQL, or dataset committed to MIFlow)

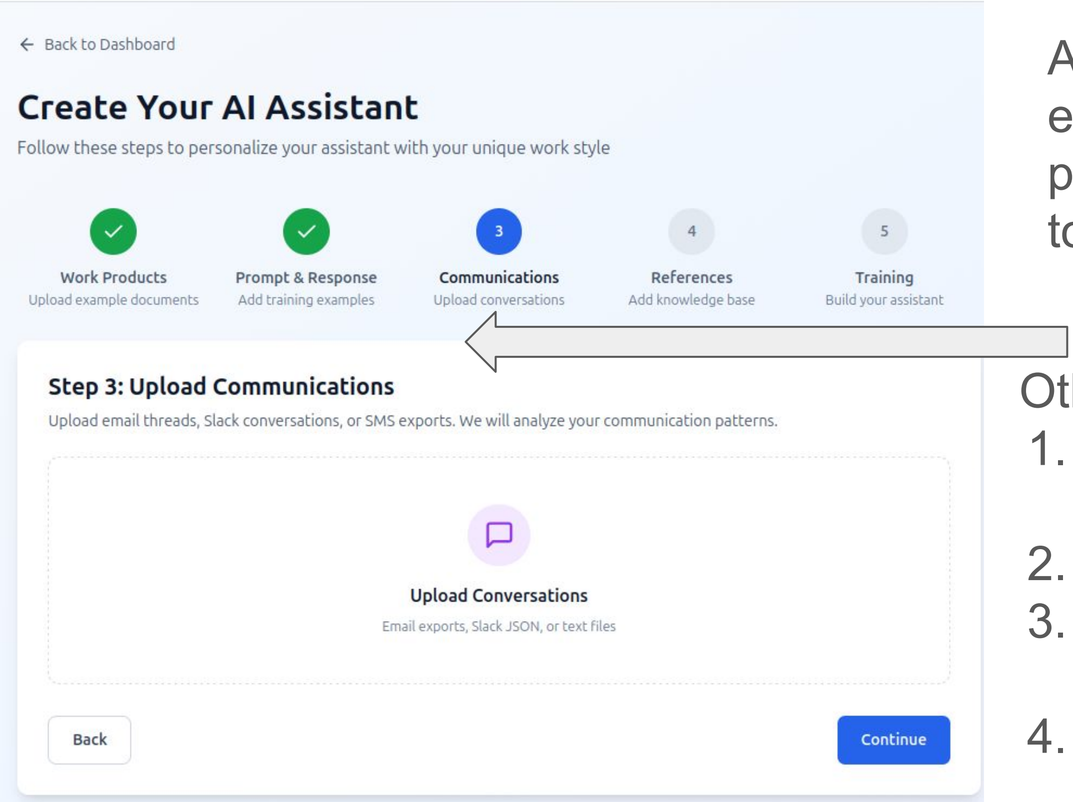
User uploaded prompt and response examples

- Source Sample Number (Primary key)
- Original prompt text
- Original reasoning path text
- Original response text

Prompt and response dataset augmented

- Sample number (primary key) (after synthetic up-sampling and permuting prompts and responses)
- Source sample ID (foreign key) (before up-sampling)
- Sample text

Step 3(a) Upload example communication Threads (Email, SMS, Slack, Discord, etc.) [PDF | word | .txt]



Add a text box for the users' email [address | username | phone number] (as it pertains to this thread)

Other elements:

1. Count of communication examples
2. Select list [email | sms | …]
3. Button [done adding communication examples]
4. ["Continue"] <-> Review Samples

Step 3: 1st pass back-end logic: A generic LLM API call packages this as training data; Up - samples

SYSTEM PROMPT = f”””

Reformat this {CONVERSATION_TYPE} as a single string

1. Reformat this as a multi - turn chat in the format “<prompt></prompt><response></response>...”
2. WHERE the messages from any user **other than {USERNAME}** is nested within the <prompt></prompt> tags and
3. The messages from {USERNAME} are within the <response></response> tags.

Example:

{EXAMPLE_OTHER+PERSON_USER_NAME}:

Foo bar

{USERNAME}:

BAZ!

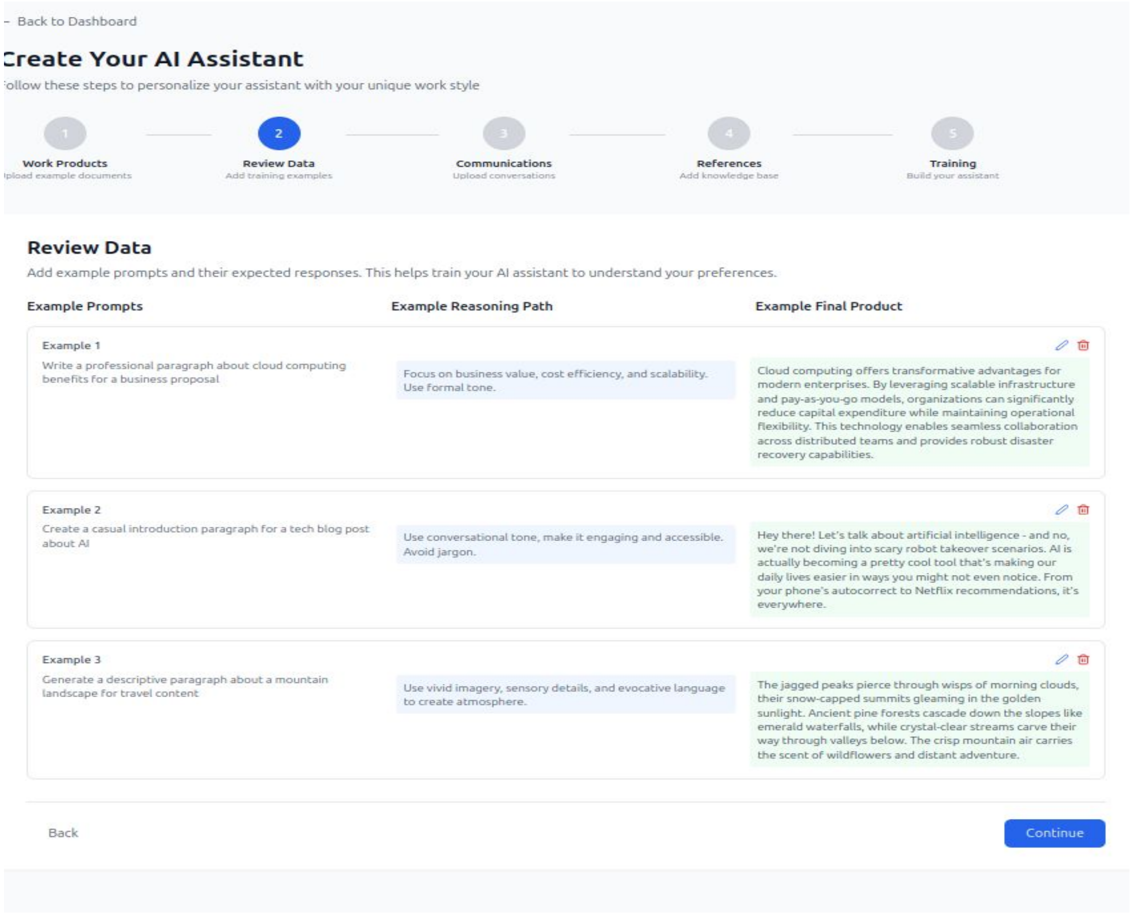
BECOMES: “<prompt>Foo bar</prompt><response>BAZ!</response>”

“””

Step 3: 1st pass back-end logic: A generic LLM API call packages this as training data; Up - samples

The same synthetic up-sampling task that happens for instruct samples is done here …

Step 3(b): User Reviews Question and answer data



Step 3: Data Relations & Outputs

Samples packaged in a flat format (CSV, SQL, or dataset committed to MIFlow)

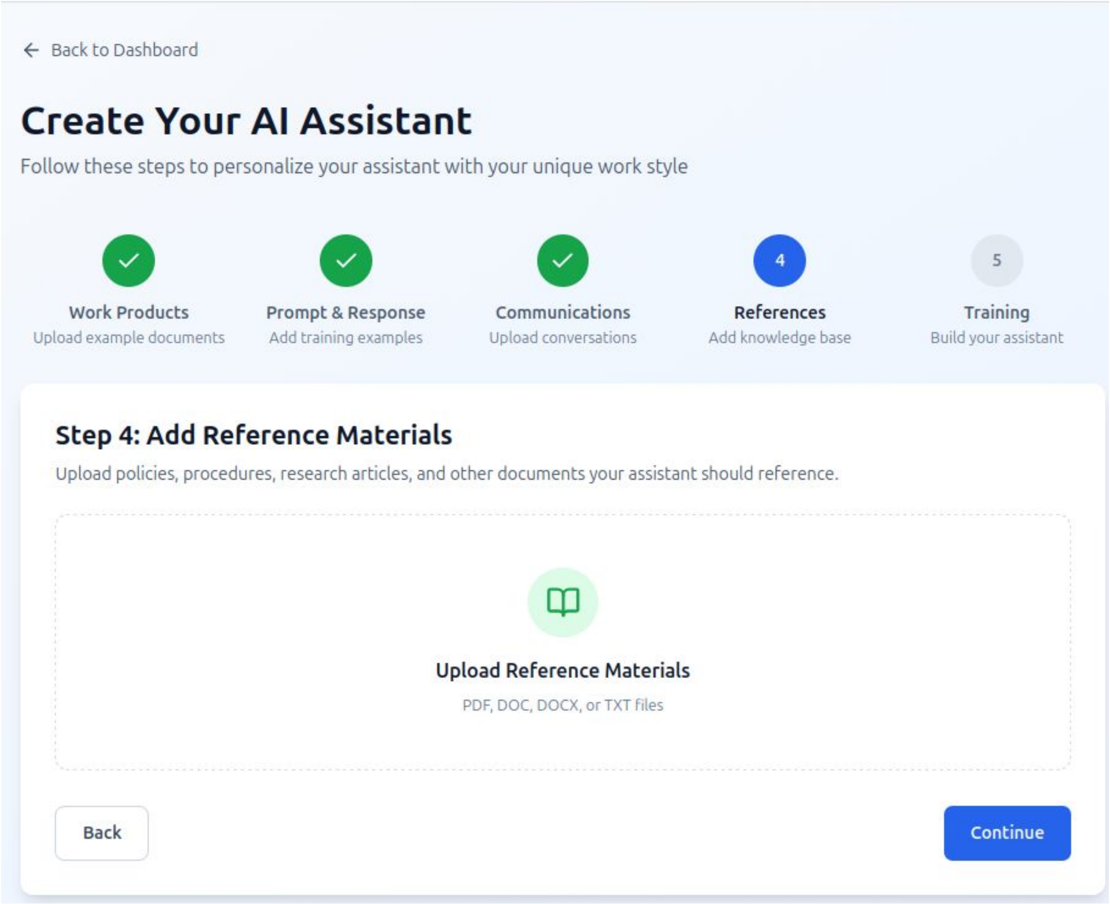
Source Data: User uploaded example communications

- Source Sample Number (Primary key)
- Original Communication Text:

Processed Data: Communications dataset augmented

- Sample number (Primary key)
- Source sample ID (Foreign key)
- Sample text

Step 4: Upload References



Back - end - logic for reference samples:

- User’s references are broken into paragraphs to fit context window.
- These are treated as simple text completion (NON-instruct) samples.
    - No <prompt></prompt>, <think></think>, <response></response> tags are applied.
    - Synthetic samples will be created to get to a few thousand samples if necessary.
    - In many cases this will not be necessary if Service manuals, SOP manuals, policy and procedure books, professional literature is uploaded.

Step 4: Data Relation Outputs

Samples packaged in a flat format (CSV, SQL, or dataset committed to MIFlow)

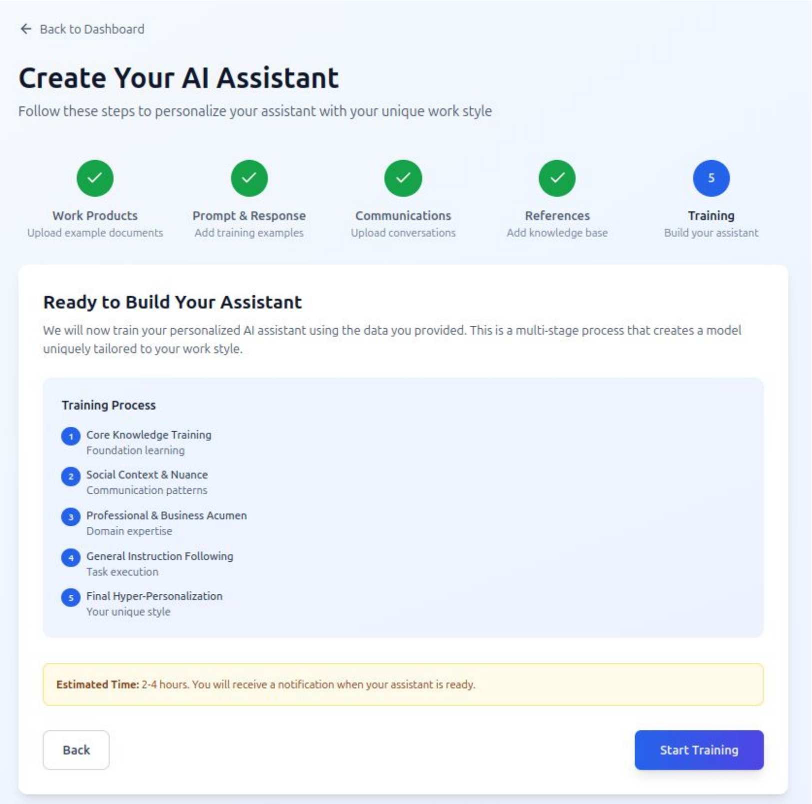
Source Data: User uploaded reference materials

- Source Sample Number (Primary key)
- Original Document Text

Processed Data: Reference dataset processed (Knowledge Base)

- Sample number (Primary key)
- Source sample ID (Foreign key): The Source Sample Number of the original document this chunk was split from.
- Chunk Text

Step 5: Conform Page



The personalization pipeline begins in the background

For stages I - IV: A subset of the Phase I,II,III, IV training data corpus is selected from each:

- **60% split of data similar to the user's uploaded data (Relevant Split)**
- **40% general randomly selected is selected and shuffled. (General Split)**

1. **Stage I Training:**
   a. Neural architecture search AND
   b. Trains a slated model on a mix of **synthetic grade-school-level textbook and literary data**.
2. **Stage II Training:**
   a. Fine-tunes the Stage I model on synthetic **social data** (news, social media conversations, sms, email).
3. **Stage III Training Lambda:**
   a. Fine-tunes the Stage II model on **general professional documents**, research articles, and business data, legal documents and additional news from a general corpus **+ the user's references which they uploaded**.
4. **Stage IV Generic Instruct Fine Tuning Lambda:**
   a. Fine-tunes the Stage III model on generic examples of **concrete instruction-following tasks** (prompt -> response examples from a common training set) to improve reasoning and adherence to user commands.
5. **Stage V User - Specific Instruct Fine Tuning Lambda (Personalization):** Performs the final fine-tuning on the user's unique, synthesized instruct data. This creates the final, hyper-personalized model checkpoint that is the model behind their assistant.

Phase I - IV Training Set Selection

Separate global corpuses for phases I–IV are stored in vector databases. From each corpus, we extract two subsets:

- *We need n samples total for a given phase*
- *k* samples for the **relevant split**, ensuring strong representation of data contextually aligned with the user's use case
- *n−k* samples for the **general split**, preserving broad knowledge for holistic reasoning.

The **search sample**—composed embedded samples from:

- Step I "work products",
- Step II "responses",
- Step III "outputs", and a
- random selection of user "references" from Step IV

The embedded **search sample** is and compared to the corpus via dot product similarity row-wise.

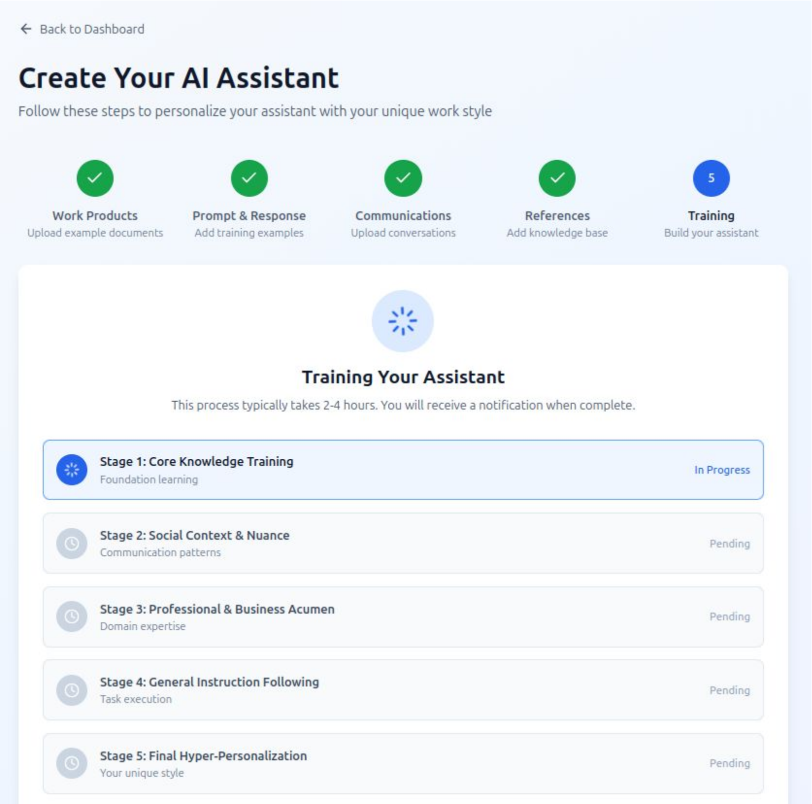
For each **search sample** row, the top-*L* most similar corpus rows (by max dot product similarity) are sampled for inclusion in the relevant split, where *L = n / number_of_rows_in_**search_sample***.

From the remaining corpus rows not sampled for the relevant split, *(n−k)* are randomly sampled for the **general split**.

What is happening in the background in a training stage

- The attached is a [very vanilla] example of the Phase I-a Neural Architecture Search, and the Phase I-b that takes the model from Phase I-a and continues training it on another set of data (Recommended audience: Those familiar with Keras or Py Torch and LLM Architecture):
    - https://github.com/david-thrower/cerebros-core-algorithm-alpha/blob/269-model-recompiled-after-phase-1-small-scale-hpo-267/generative-proof-of-concept-CPU-preprocessing-in-memory.py
    - The model is a novel model architecture, and not an attention transformer:
        - Single head architecture
        - Non - Linear architecture and other architectural details:
            - (See Readme.md) https://github.com/david-thrower/cerebros-core-algorithm-alpha
            - (Deeper technical details of the model architecture) https://github.com/david-thrower/cerebros-core-algorithm-alpha/blob/main/documentation/cerebros-technical-details.md
            - Text is embedded by iRoPE embedding

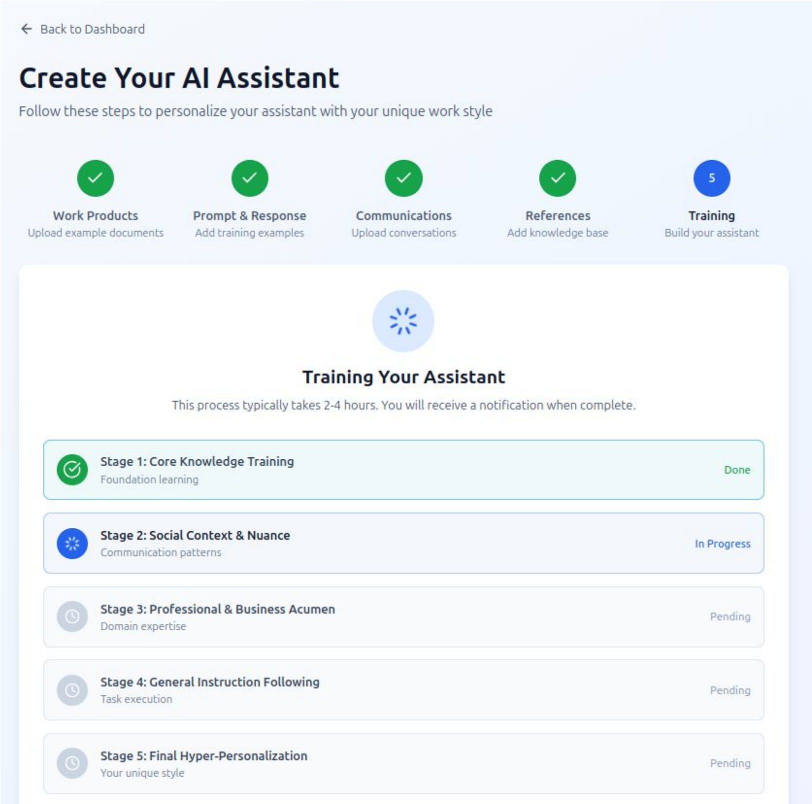
Update: Stage 1 Training in progress



Stage I training lambda Runs:

- **Inputs:**
  1. Stage I Training data - relevant
  2. Stage I Training data - general
- **Tasks:**
  - Merge the 2 data sets.
  - Shuffle the order.
  - Train Stage I-a model checkpoint on [TBD] % of the combined data
  - Train Stage I-b model checkpoint on the entire data.
- **Outputs:**
  - Stage I-b model checkpoint saved on NFS volume (.keras file)
  - Metrics posted to MIFlow
  - [Copy of model posetd to MIFLow also]

Update: Stage 1 Training done, Stage 2 Training in progress



Stage II training lambda runs:

Inputs:

- Stage I-b model checkpoint
- Stage II Training data - relevant
- Stage II Training data - general

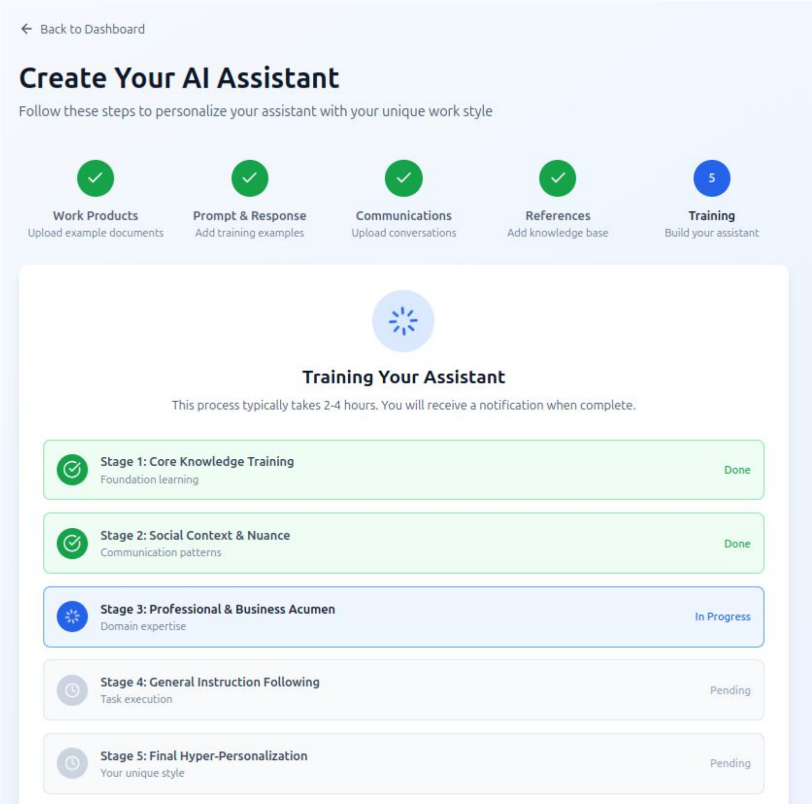
Tasks:

- Load Stage I-b model checkpoint.
- Merge the relevant and general Stage II data sets.
- Shuffle the order.
- Continue training (fine-tune) the model on the merged data.

Outputs:

- Stage II model checkpoint saved on NFS volume (.keras file)
- Metrics posted to MlFlow

Update: Stage 2 Training done, Stage 3 Training in progress



Stage III training lambda runs:

Inputs:

Stage II model checkpoint

Stage III Training data - relevant

Stage III Training data - general

User Reference dataset processed (Knowledge Base)

Tasks:

Load Stage II model checkpoint.

Merge Stage III Training data - relevant, Stage III Training data - general, User Reference dataset processed (Knowledge Base).

Shuffle the order.

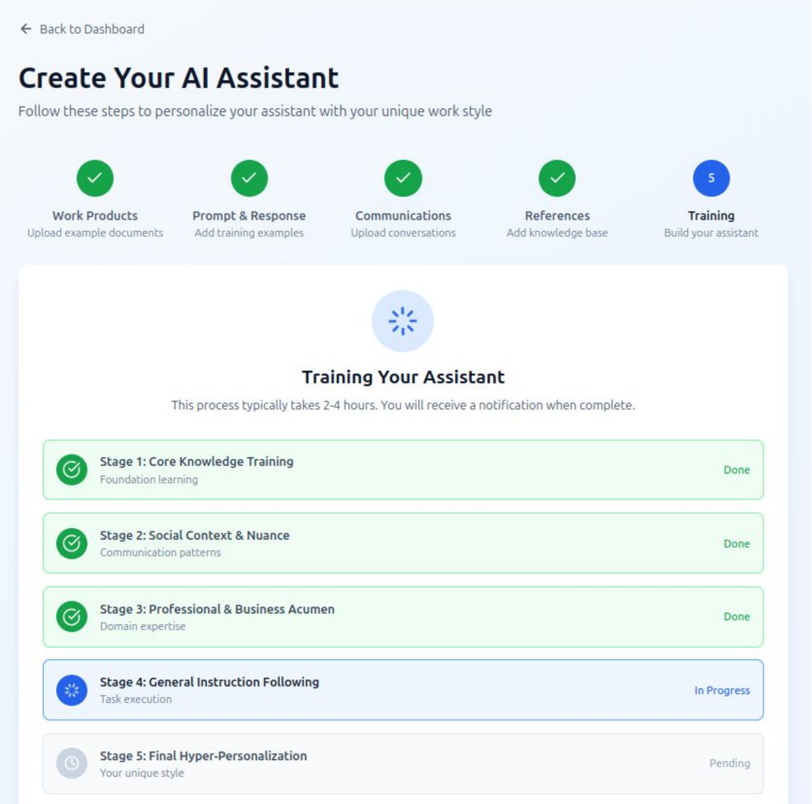
Continue training (fine-tune) the model on the merged data.

Outputs:

Stage III model checkpoint saved on NFS volume (.keras file)

Metrics posted to MlFlow

Update: Stage 3 Training done, Stage 4 Training in progress



Stage IV training lambda:

Inputs:

- Stage III model checkpoint
- Stage IV Training data - relevant
- Stage IV Training data - general

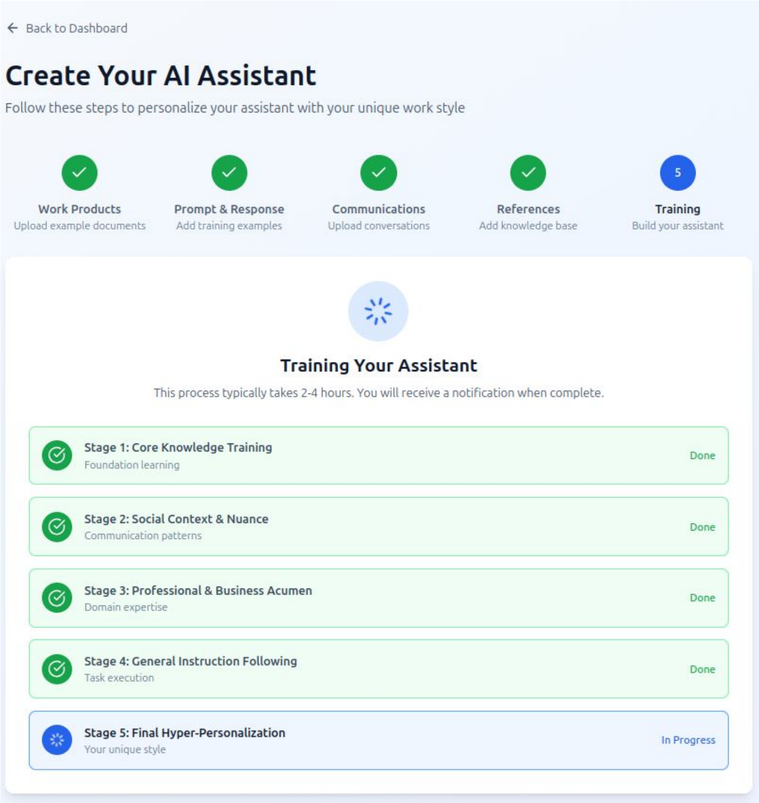
Tasks:

- Load Stage III model checkpoint.
- Merge the relevant and general Stage IV data sets.
- Shuffle the order.
- Continue training (fine-tune) the model on the merged data.

Outputs:

- Stage IV model checkpoint saved on NFS volume (.keras file)
- Metrics posted to MlFlow

Update: Stage 4 Training done, Stage 5 Training in progress



Stage V Personalization Fine-Tuning lambda:

Inputs:

Stage IV model checkpoint

User Example Work Product instruct samples augmented

Prompt and response dataset augmented

Communications dataset augmented

Tasks:

Load Stage IV model checkpoint.

Merge all of the user's augmented personal instruct data sets.

Shuffle the order.

Perform final fine-tuning on the merged user-specific data.

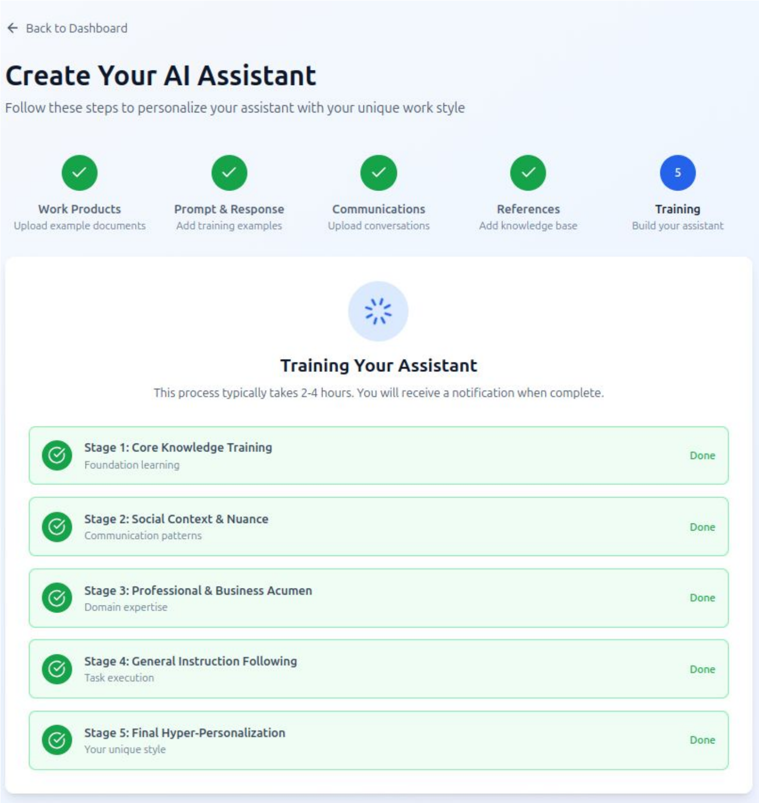
Outputs:

User's Personalized Assistant checkpoint saved on NFS volume (.keras file)

Final metrics posted to MIFlow

Signal completion to the training status tracker (UX Step 9).

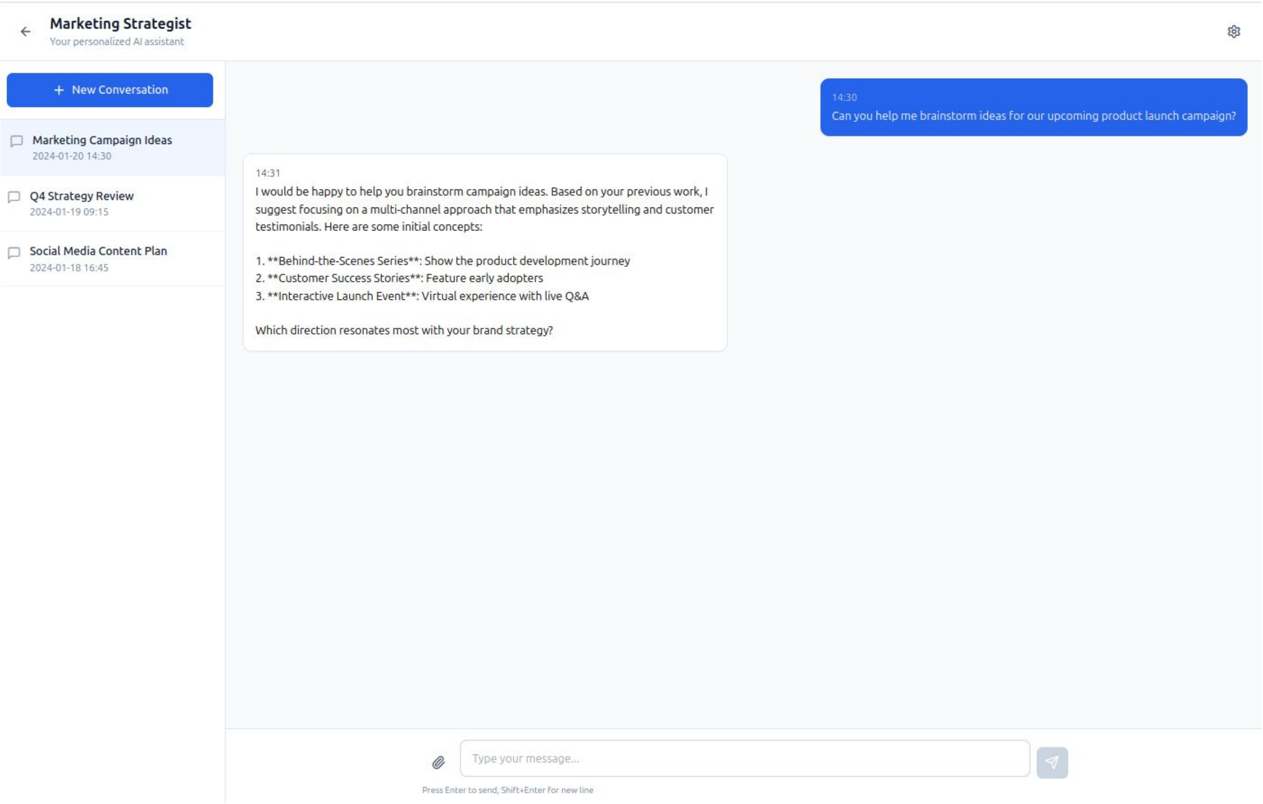
Update: Success!



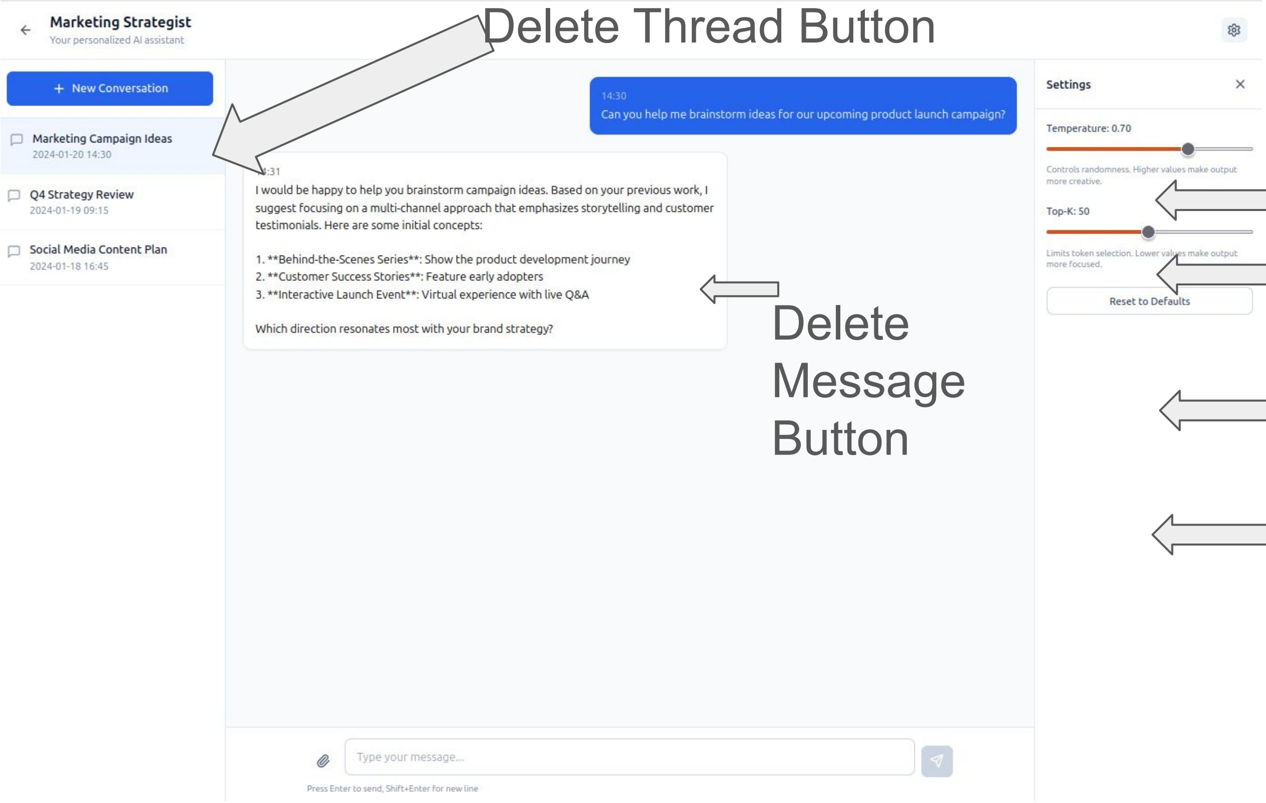
The Assistant Deployment Lambda Runs

- REST API model endpoint deployed for the final model
- UI Container deployed and directed to the model endpoint
- Tables created in Postgres for the User’s conversations

UI: User Uses Their New Assistant



Use your Assistant: Advanced Settings



Delete Thread Button

Delete Message Button

top_p: 0.6:0.995

repetition_penalty: 1:1.5

presence_penalty: 1:1.5

frequency_penlty: 1:1.5