Evrika Briefs

INNA IVANOVA IVANOVA

CHAT WITH YOUR VIDEO. GET A SMART BRIEF.

↳ INNOVATEINNA@GMAIL.COM

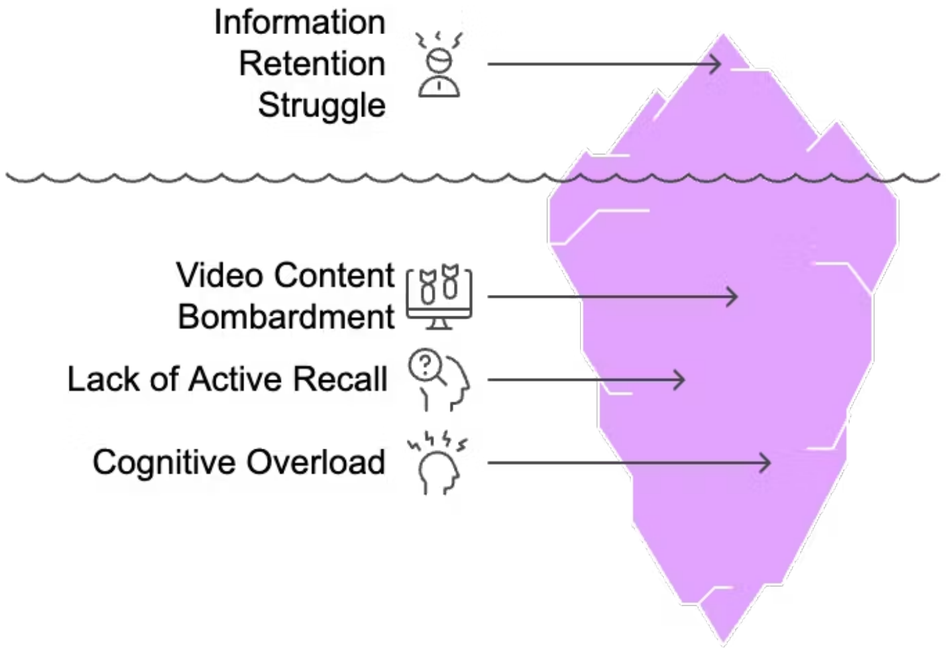
AGENDA

Introduction

EVRIKA BRIEFS

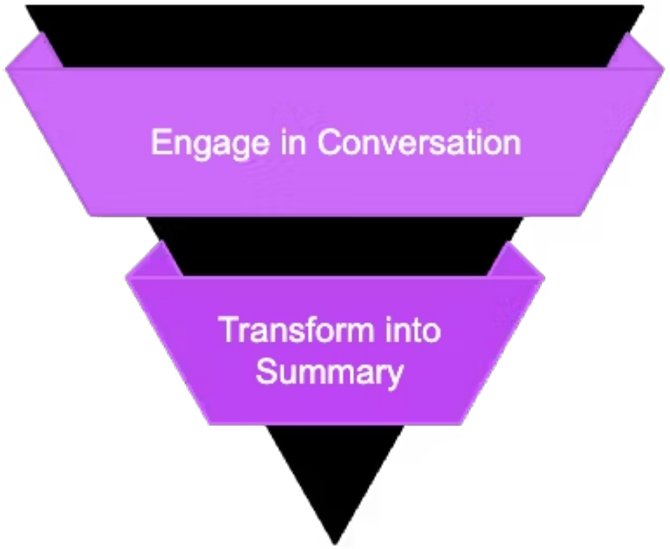
The Problem - video information overload

Video Overload Hinders Information Retention.



The Solution - Evrika Briefs

From Video through Conversation to Intelligent Summary



Technical Summary

TECHNICAL SUMMARY - USE CASE AND PROCESS

1. The user provides a YouTube **link**
2. The video is processed in the background (**ingestion**)
3. The transcript is fetched (**transcription**)
4. The **transcripts** and **embeddings** are **saved in the database**
5. The user **types a question or sends audio question**
6. The **answer** comes from the RAG Pipeline, unless it is metadata or recommendation
7. The user can **generate and edit a brief**
8. The user can **download the brief in PDF format**

TECHNICAL SUMMARY - MODULES

- **config.py** (LLM - gpt-4o-mini, temperature=0.0, as well as embeddings model and the splitter)
- **transcripts.py** (fetch YouTube transcript)
- **supabase_store.py** (store and retrieve docs)
- **rag_pipeline.py** (RAG & Tools)
- **metadata_tool.py** (metadata and recommendations)
- **audio_utils.py** (LMM - gpt-4o-mini-tts, voice=nova)
- **voice_api.py** (all endpoints: ingestion+chat, generate brief, safe brief as pdf, voice endpoint)

Module Name	Description
config.py	LLM configuration
transcripts.py	Fetch YouTube transcript
supabase_store.py	Store and retrieve documents
rag_pipeline.py	RAG and Tools
metadata_tool.py	Metadata and recommendations
audio_utils.py	LMM for text-to-speech
voice_api.py	API endpoints

TECHNICAL SUMMARY - FRONTEND <-> BACKEND CONNECTION

1. FastAPI — My Backend Logic

- Python logic
- Endpoints like /chat, /ingest, and /brief.
- Each endpoint calls the RAG pipeline, LLM, and Supabase

2. Uvicorn — Runs the API Locally

- Starts the server on http://localhost:8000

3. ngrok — Makes the Local API Public

- Creates a secure public URL (e.g., https://xyz.ngrok.app).
- Forwards all requests to my local FastAPI server

4. Lovable — The Frontend Calls the API

- Fetch requests from Lovable and use the ngrok URL
- Example:
  POST https://xyz.ngrok.app/chat → backend → response in UI

Process:

Python logic → FastAPI → Uvicorn → ngrok → Lovable

Results:

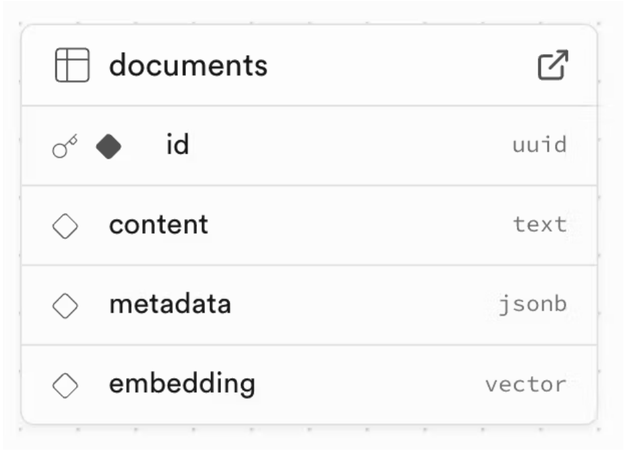
✔ Local backend
✔ Public temporary endpoint
✔ Fully connected frontend → backend workflow

Dataset

DATASET

The YouTube link is provided by the user, then:

- Fetching YouTube via transcripts.py
- Chunking during transcription process
- Storage in Supabase Database
  - Embeddings saved in embedding column
  - Metadata saved in metadata column



Transcription process

1. fetch_metadata_with_ytdlp
2. try_fetch_transcript_via_api (YouTubeTranscriptAPI)
3. fetch_audio_with_ytdlp
4. split_audio_if_needed
5. transcribe_with_whisper
6. join full_text

Storage and Retrieval processes

1. store_docs_in_supabase
   - stores document chuncs and embeddings in documents table
   - uses pgvector for the embeddings column
2. retrieve_docs_from_supabase
   - retrieves k nearest chunks for a query

Q&A Agent

CHAT WITH YOUR VIDEO. GET A SMART BRIEF.

↳ INNOVATEINNA@GMAIL.COM

Q&A AGENT - ARCHITECTURE

LLM Model

- model=”gpt-4o-mini”

Embeddings

- model = “text-embeddings-3-small“

Splitter

- splitter = RecursiveCharacterTextSplitter
- (*chunk_size*=1000, *chunk_overlap*=200)

Voice Model

- model: str = ”whisper-1” (transcription)
- model: str = ”gpt-4o-mini-tts”
- voice = “nova“

Supabase

- with pgvector enabled for embeddings

Simple memory

- CHAT_HISTORY

LLM Model:

```
model="gpt-4o-mini",
   temperature=0.0,
```

Embeddings and Chunk Splitter

```
embeddings = OpenAIEmbeddings(model="text-embedding-3-small")
splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
```

Speech:

```
model: str = "gpt-4o-mini-tts",
   voice: str = "nova",
```

Database:

```
# save embeddings, store and retrieve docs from the db
Supabase with pg vecor
```

Memory:

```
# -------- Chat history (simple memory) --------
CHAT_HISTORY: List[HumanMessage] = []
```

Q&A AGENT - RAG & TOOLS

🧠 RAG Pipeline:

- **Ingesting** YouTube videos into Supabase
- **Semantic search** over stored chunks (*match_count*=6)
- **Chatting** with a video (RAG)
- Generating an **"Evrika Brief"** in a fixed template
- Simple **recommendations**
- Saving a brief as a **PDF**

Retrieval step

– First call Supabase via a Remote Procedure Call RPC:

– match_documents with:

-> the user's question embedding

-> k = top N chunk

– If 0 chunks are returned:

Fall back to _get_all_chunks_for_video(youtube_id)
-> this is fallback retriever.

Generation step

A prompt that includes:

– System instructions (answer based only on the context, etc.)

– The retrieved chunks as context.

– The user's question.

Then call:

– llm.invoke(...) / ChatOpenAI (from config.py).

– The model generates the final answer.

Q&A AGENT - RAG & TOOLS

Tools:

- @tool: fetch_video
- @tool: semantic_search
- _run_qa function
- @tool: video_chat (RAG)

Ingest a YouTube video by URL or ID into the system.

```
@tool
def fetch_video(url: str) -> str:
```

Retrieve the most relevant transcript chunks for a question.

```
@tool
def semantic_search(question: str, video_hint: str = "") -> str:
```

Handles metadata and recommendation questions

```
def _run_qa(question: str, video_hint: str = "") -> str:
```

Answer questions about the CONTENT of the video using transcript chunks. (RAG)

```
@tool
def video_chat(question: str, video_hint: str = "") -> str:
```

Q&A AGENT - RAG & TOOLS

Tools:

- @tool: generate_brief
- @tool: recommendations
- save_brief_as_pdf function

Creates an Evrika Brief Summary PDF or a video.

```
@tool
def generate_brief(video_hint: str) -> str:
```

Suggest follow-up learning directions after a video.

```
@tool
def recommendations(video_hint: str, learning_goal: str = "") -> str:
```

Saves brief as PDF

```
def save_brief_as_pdf(brief_text: str, filename: str = "evrika_brief.pdf")
```

Q&A AGENT - RAG & TOOLS

Tools:

- metadata_tool.py - added later to improve speed issue and enable better answers to anticipated questions from the metadata without running through the whole RAG system

```
"""
LangChain tool for reading *compact* video metadata from Supabase.

This lets the agent answer questions like:
- Who is the speaker / which channel is this?
- What is the title?
- How long is the video?
- When was it published?

IMPORTANT: we deliberately return ONLY a small subset of metadata
to avoid context_length_exceeded errors.
"""
```

Q&A AGENT - PROMPT TEMPLATE

Prompts:

- Answer prompt used for RAG System

```
You are Evrika Briefs, an assistant that answers questions about YouTube videos
using their transcript chunks.

Use ONLY the information from the provided chunks to answer the question.
If the answer is not clearly in the chunks, say that you are not sure.

Relevant chunks:
{context}

User question:
{question}

Answer in a clear, concise way, 3–7 sentences maximum.
```

Q&A AGENT - PROMPT TEMPLATE

Prompts:

- Answer prompt for recommendation questions

```
You are a learning coach inside Evrika Briefs.

A user has just watched a YouTube video and ingested it into the system.
They may have the following learning goal (optional):

Learning goal: {learning_goal or "(none given)"}

Here are a few chunks from the video transcript for context:
\"\"\"{context}\"\"\"

Suggest 3–7 concrete follow-up learning steps, including:
- Search queries they could type into YouTube or Google
- Concrete topics or subskills to explore next
- Optional: types of videos (tutorial, case study, lecture, etc.)

Return the answer as a Markdown bullet list.
```

Q&A AGENT - PROMPT TEMPLATE

Prompts:

- Prompt Template for the PDF
  - Instructions

```
You are Evrika Briefs, a tool that turns YouTube videos into smart 1-page briefs.

You generate markdown that will be exported to PDF.

Very important formatting rules:

1. Use EXACTLY the headings and section order from the template.
2. Do NOT add new sections or remove any sections.
3. Do NOT add any explanatory text like “leave empty for the user to fill in”.
4. For the section "## Personal Notes":
   - Do NOT change it at all.
   - Leave the three bullet lines exactly as they are: "- …" on each line.
5. Keep the line starting with "Generated:" directly under the title. If you don't
6. For "Source, Links & References":
   - Fill only the bracketed or clearly labeled parts (title, URL, creator), but d
7. If information is unknown, write "Unknown" ONLY in clearly labeled places (e.g.
8. Keep the markdown syntax valid and do not escape or reformat the template struc

Now, using the template below, replace only the placeholder text in square bracket

Write a structured brief in Markdown following exactly this structure:
```

Q&A AGENT - PROMPT TEMPLATE

Prompts:

- Prompt Template for the PDF
  - Template

```
# [Evrika Brief – Video Title (Speaker)]

Generated: Unknown

## The Main Idea

What is this really about? Why does it matter? Summarize in up to 5–8 short sentei

---

## Relevant Models & Frameworks

- [model 1] – short description, when/why to use.
- [model 2] – short description, when/why to use.
- [model 3] – short description, when/why to use.
- [model 4] – if relevant.
- [model 5] – if relevant.

---

## Top Insights

- [insight 1]
- [insight 2]
- [insight 3]
- [insight 4] (if needed)
- [insight 5] (if needed)

---

## Memorable Quotes

- “Quote 1…” – [who said it, if relevant]
- “Quote 2…” – [who said it, if relevant]
- “Quote 3…” – [who said it, if relevant]

---

## How to Apply This

- 3–5 short, practical suggestions

---

## Personal Notes

- …
- …
- …

---

## Source, Links & References

- Original video (title + URL): [Video Title] – [Video URL]
- Creator / Channel:
- Resources or tools explicitly mentioned in the video:

Base your brief ONLY on the transcript text below.
If you are unsure about specific names or numbers, be honest and approximate.
```

Q&A AGENT - VOICE

Audio utilities for Evrika Briefs:

Speech-to-Text (STT)

- function transcribe_question_bytes ("whisper-1")
- short user question from microphone

Steps:

1. Receive raw audio bytes from the microphone (e.g. .m4a, .mp3, ...)
2. Call OpenAI whisper-1 to **transcribe** that audio
3. Return a **text string** like "What is 10x thinking?"

Text-to-Speech (TTS)

- function synthesize_answer_tts
- short spoken answer as audio bytes

Steps:

1. Take the **answer text** from the agent
2. Call the TTS endpoint
3. Return **audio bytes** (e.g. .mp3) that the frontend can play

Speech:

```
model: str = "gpt-4o-mini-tts",
   voice: str = "nova",
```

audio_utils.py

```
"""
Audio utilities for Evrika Briefs:
- STT: transcribe_question_bytes -> short user question from microphone.
- TTS: synthesize_answer_tts -> short spoken answer as audio bytes.
"""
```

def transcribe_question_bytes

```
  """
    Transcribe a short audio snippet (user question) using OpenAI Whisper.

    Accept whatever the browser records (e.g. m4a), decode it with pydub/ffmpeg,
    convert it to WAV in memory, and then send the WAV to Whisper.

    This avoids "Invalid file format" errors for tricky m4a encodings.
  """
```

def synthesize_answer_tts

```
  """
    Turn a short text answer into speech audio bytes using OpenAI TTS.

    Returns:
        (audio_bytes, mime_type)
    """
```

Tools

TOOLS

Tool Stack

Frontend - Lovable

Backend

FastAPI

uvicorn

ngrok

Supabase with pgvector als vector database for the embeddings

python files

APIs and Libraries

OpenAI

Langchain

yt-dlp, youtube-transcript-api, pydub, ffmpeg

base64, re, and others see requirements.txt

PDF Export

reportlab

Environment Variables

python-dotenv

Technical Support

ChatGPT

Evaluation

EVALUATION - MANUAL

Problems

- **Ingestion** process was too slow
- Didn't answer questions about the **metadata** or **recommendations**
- Didn't know who the **speaker** is
- PDF didn't follow the **template**

Test Video	https://www.youtube.com/watch?v=rWv-KoZnpKw		
Questions	What is the title of the video?	metadata	✅
	What is the channel of the video?	metadata	✅
	Who is the speaker in the video?	metadata	❌
	Wht is the URL of the video?	metadata	✅
	How long is the video?	metadata	✅
	What is the video about?	RAG	✅
	What is the main idea?	RAG	✅
	What are the 12 main points of the video?	RAG	✅
	When did the speaker work with Steve Jobs?	RAG	✅
	What did the speaker learn from Steve Jobs?	RAG	✅

Solutions

- **Check** if video was **already ingested**, then faster
- **Check** if it is a **metadata** or **recommendation** question, then handle separately also faster
- If it **doesn't know the speaker**, should say it's **not sure**
- PDF follows the **template** through **explicit prompt instructions**

EVALUATION - RAGAS

RAGAS Results - Averages

Faithfulness: 0.98
Answer relevancy: 0.73
Context precision: 0.91
Context recall: 0.88

Interpretation

- Evrika Briefs **retrieves the right chunks,** uses them correctly, and usually **answers the question well.** (no hallucination)
- There's still some **room of improvement** to **sharpen the answers** for a few questions. (abstract questions)

```
===== RAGAS RESULTS — PER SAMPLE =====
                                    user_input  ... context_recall
0                        What is 10x thinking?  ...            1.00
1         What is compound interest of habits?  ...            1.00
2                       What is jumping curves?  ...            1.00
3        What is the build measure learn loop?  ...            1.00
4          What are the three core principles?  ...            0.75
5     What places should I visit in Barcelona?  ...            1.00
6                                What is a RAG?  ...            1.00
7  What is the advantage of a vector database?  ...            1.00
8   When did the speaker work with Steve Jobs?  ...            1.00
9                        What makes innovation?  ...            0.00

[10 rows x 8 columns]

===== RAGAS RESULTS — AVERAGES =====
faithfulness         0.980000
answer_relevancy     0.731262
context_precision    0.910139
context_recall       0.875000
dtype: float64
```

Conclusion

CONCLUSION - NEXT STEPS

Technical improvements

Optimization Ideas:

- exchange “whisper-1“ with “gpt-4o-mini-transcribe“ (in case it works in combination with yt-dlp) **to speed up video ingestion**
- change the LLM Model to “gpt-4.1-mini” **for better quality answers**
- experiment with chunk_size and chunk_overlap in chunk_text (splitter) **for better answers**
- experiment with match_count in _match_documents (retriever) **for more relevant answers**
- implement streaming f**or faster frontend - backend connection**
- improve **personality**

↳ INNOVATEINNA@GMAIL.COM

CONCLUSION - NEXT STEPS

Further Features

Feature Ideas:

- Improve **speed** of ingestions of new videos
- Improve **recommendations** to deliver **new YouTube videos**
- Accept **other files** especially **PDFs**
- Add **login**
- Add **profile page**
- Add **document history** or archive

CONCLUSION - FINAL STATEMENT

Conclusion

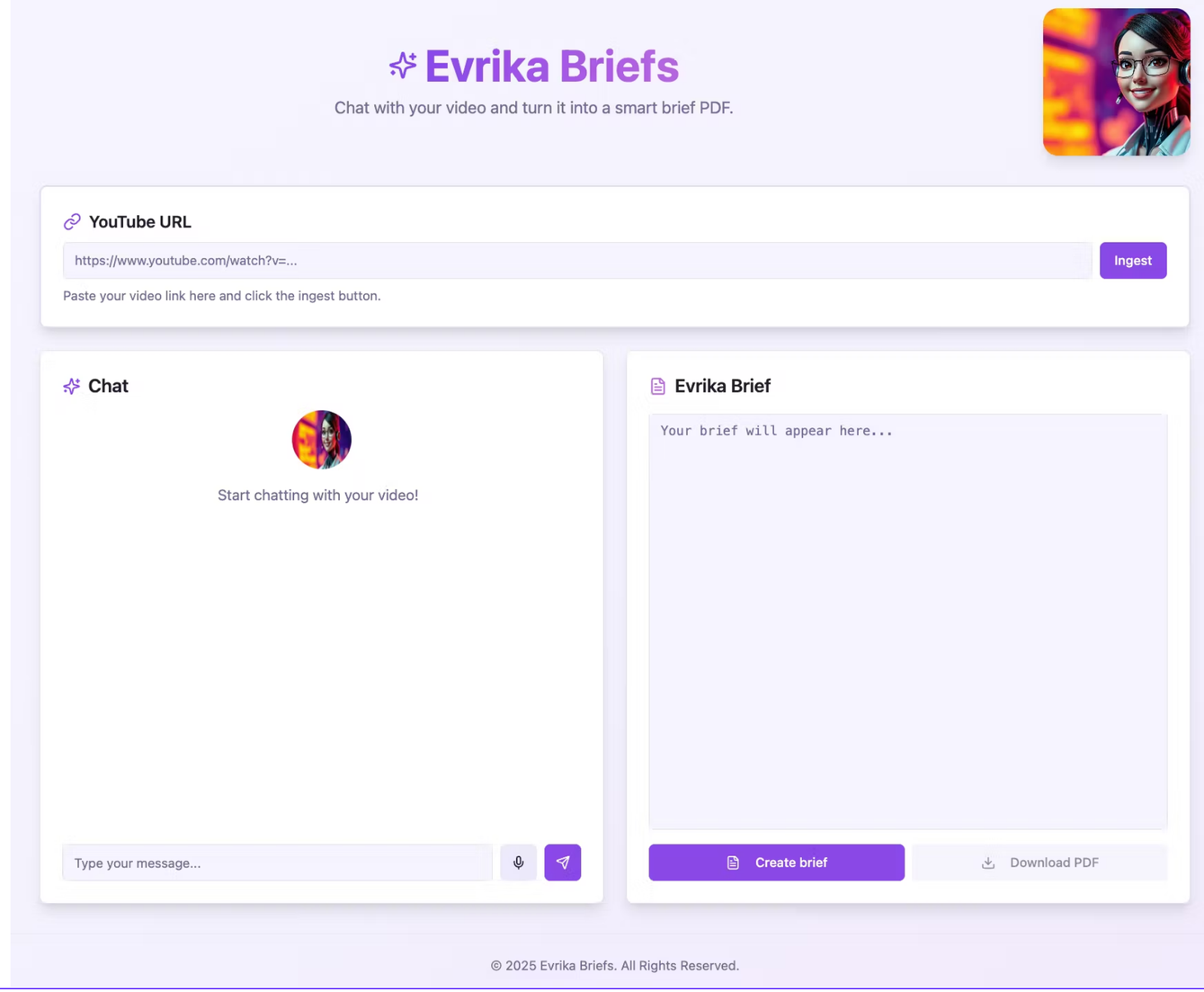
EVRIKA BRIEFS IS A PRACTICAL SOLUTION FOR A REAL PROBLEM!

EVRIKA BRIEFS.
CHAT WITH YOUR VIDEO.
GET A SMART BRIEF.

↳ INNOVATEINNA@GMAIL.COM

DEMO

DEMO


Evrika Briefs
Chat with your video and turn it into a smart brief PDF.
YouTube URL
https://www.youtube.com/watch?v=...
Ingest
Paste your video link here and click the ingest button.
Chat
Start chatting with your video!
Type your message...
Evrika Brief
Your brief will appear here...
Create brief
Download PDF
© 2025 Evrika Briefs. All Rights Reserved.

Thank you!

CHAT WITH YOUR VIDEO. GET A SMART BRIEF.

↳ INNOVATEINNA@GMAIL.COM

Try Pitch

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)