

# Project Report: Codex – A Personal Context-Aware Agent

## Overview

Codex is a personalized, context-aware assistant designed to answer questions about Umutesa using a combination of document retrieval, natural language generation, and interactive user experiences. The system integrates multiple modalities—text, voice, images, and structured data—to deliver intelligent, engaging, and personalized responses. This report outlines the technical architecture, design decisions, and creative enhancements that shaped the development of Codex.

## Step 1: Retrieval-Augmented Generation (RAG) Architecture

The foundation of Codex is built on the RAG framework, which combines document retrieval with generative modeling to produce accurate and contextually relevant answers.

### How RAG Works:

#### 1. Document Chunking

Documents are split into manageable chunks using a recursive character-based splitter:

```
splitter = RecursiveCharacterTextSplitter(
    chunk_size=800,
    chunk_overlap=100,
    separators=["\n\n", "\n", ". ", " ", ""]
)
split_docs = splitter.split_documents(docs)
```

#### 2. Embedding & Indexing

Chunks are embedded using the sentence-transformers/all-MiniLM-L6-v2 model and stored in a FAISS vector database for efficient similarity search:

```
embeddings = HuggingFaceEmbeddings(model_name="...",
    model_kwargs={'device': 'cpu'})
db = FAISS.from_documents(split_docs, embeddings)
retriever = db.as_retriever(search_kwargs={"k": 5})
```

#### 3. Generative Model Pipeline

The generative model (google/flan-t5-small) is used to synthesize answers based on retrieved context:

```
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name,
    torch_dtype=torch.float32)
hf_pipeline = pipeline("text2text-generation", model=model,
    tokenizer=tokenizer)
```

## Step 2: Parameter Tuning & Model Selection

Choosing the right model and configuration was critical to ensure performance

### Key Considerations:

- **Model Type:** Preference for lightweight models that run efficiently on CPU without requiring external APIs.
- **Performance Goals:**
  - Fast response time
  - No dependency on GPU

- **Retriever Settings:**
  - k value tuning to balance relevance and diversity of retrieved chunks
- **Chunk Size & Overlap:**
  - Optimized to preserve semantic coherence while maximizing coverage
- **Pipeline Constraints:**
  - max\_length and other generation parameters fine-tuned for clarity and depth

These parameters directly influence the accuracy, realism, and contextual depth of the agent's responses.

### Step 3: Generating Accurate Answers

#### Retrieved Content for Relevance

```
retriever = db.as_retriever(search_kwargs={"k": 5})
```

This step narrows down the vast document database to the top k most relevant chunks based on semantic similarity to the user's query.

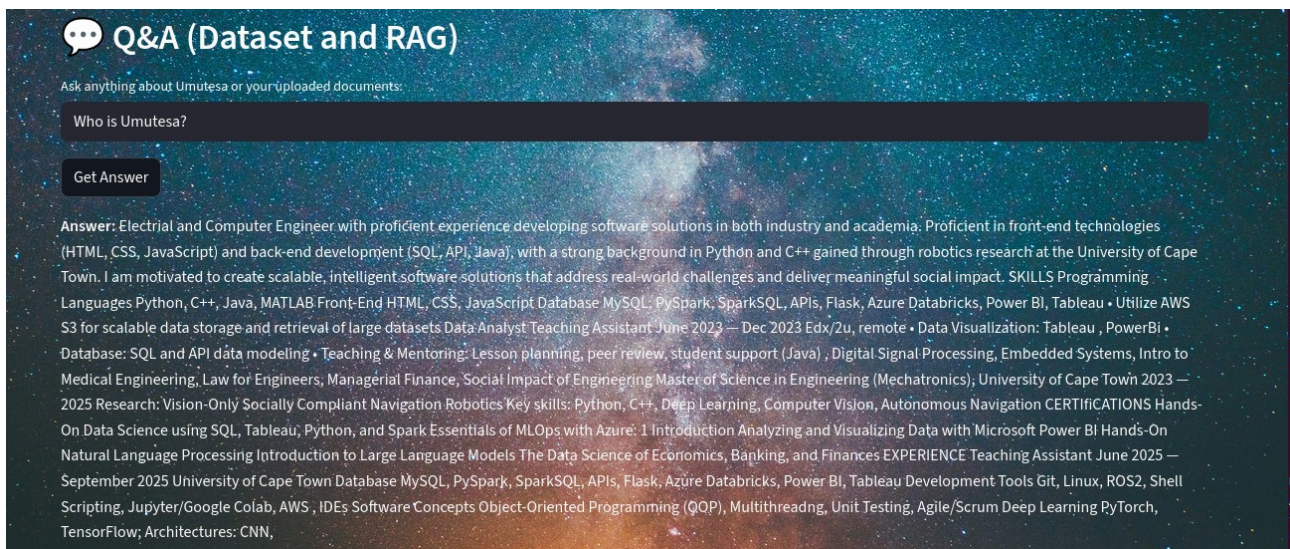
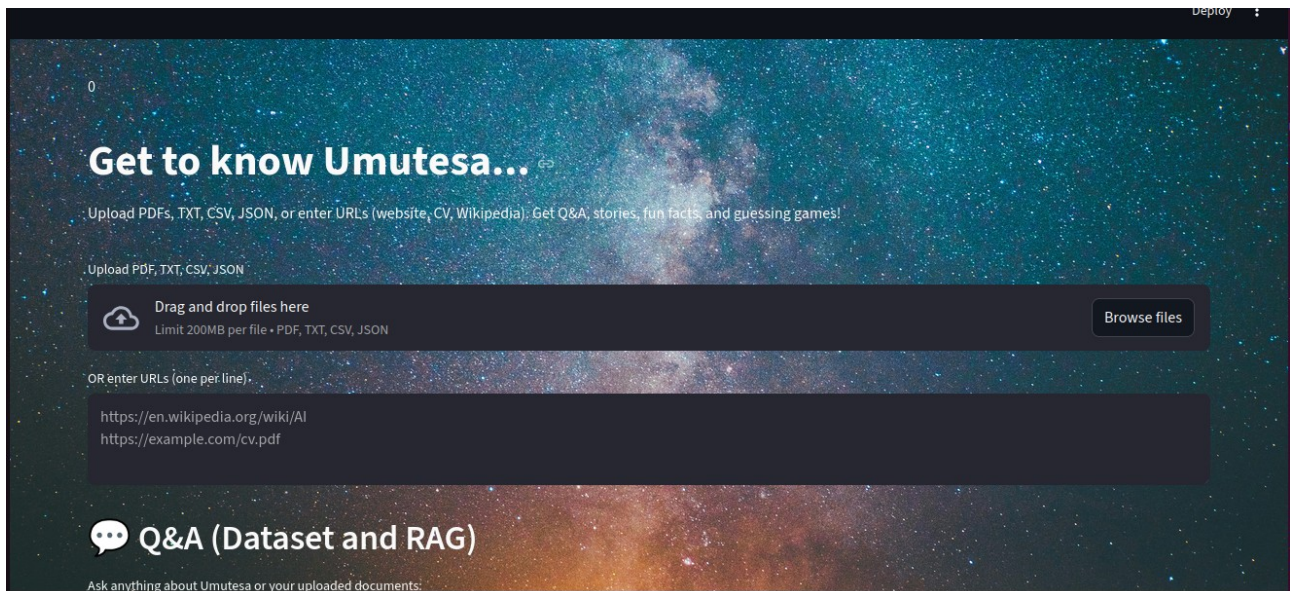
### 2. Similarity Scoring to Rank Responses

```
def get_best_facts(query, fact_database, top_n=2):
    from difflib import SequenceMatcher
    scored = []
    for fact in fact_database:
        score = SequenceMatcher(None, query.lower(),
                                fact['question'].lower()).ratio()
        scored.append((score, fact))
    scored.sort(reverse=True)
    return [fact for score, fact in scored[:top_n] if score > 0.15]
```

This function ranks pre-written facts or answers based on how closely their associated questions match the user's query. Uses SequenceMatcher from Python's difflib to compute a similarity ratio between the query and each stored question. Each fact is scored based on this ratio. The top n facts with the highest similarity scores (above a threshold of 0.15) are returned.

### Step 4: User Interface & Personalization

Codex features a simple yet expressive front-end built with **Streamlit**, chosen for its ease of use and rapid prototyping capabilities.



### Personalization Elements:

- **Custom GUI:** Includes personal information about Umutesa to enrich context
- **Image Integration:** Visual storytelling using curated images related to Umutesa's life and interests
- **Voice Features:** Intended to use a female voice resembling Umutesa's tone (though voice matching was limited by system constraints)

### Step 5: Creative Enhancements

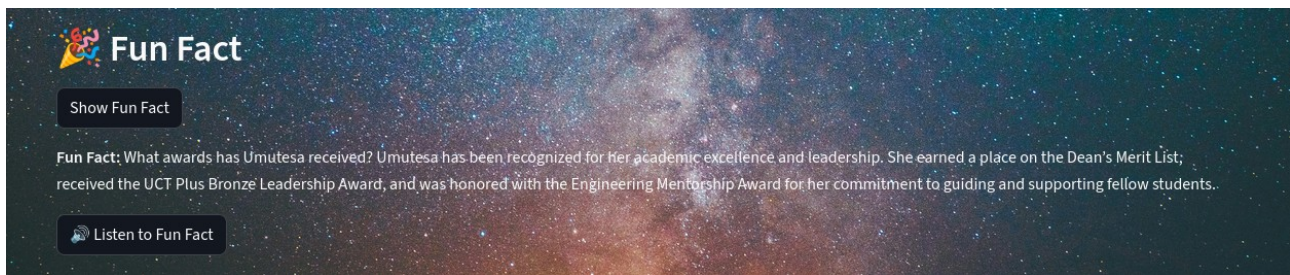
To make Codex more engaging and interactive, several creative modules were added:

#### Interactive Modules:

##### 1. Fun Fact Generator

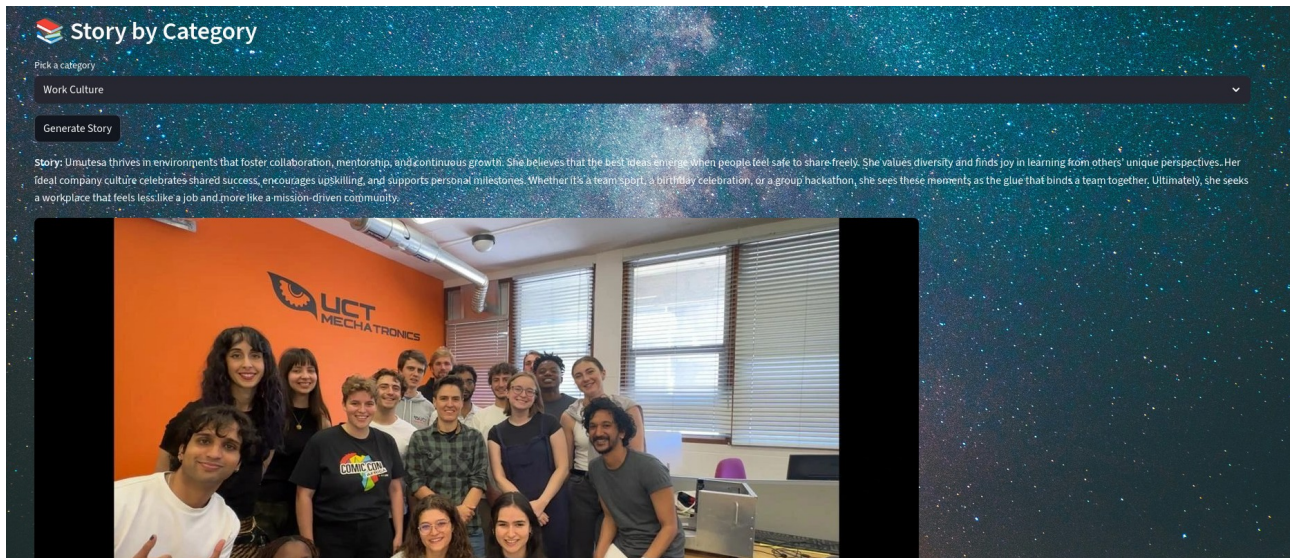
- Randomly selects a fact from the knowledge base to share with the user





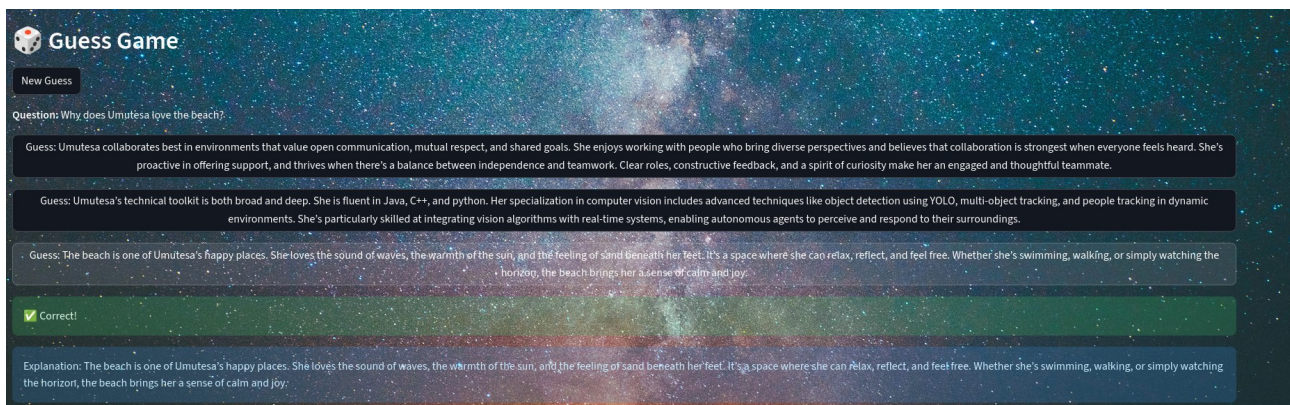
## 2. Storytelling Engine

- Categorized stories drawn from datasets to illustrate personal experiences or values



## 3. Guessing Game

- Presents multiple-choice questions based on uploaded documents or personal data



## Supported Inputs

Codex supports a wide range of input formats:

- PDF documents (e.g., CVs, academic papers)
- Text/cvs files
- JSON datasets
- URLs (e.g., personal website)

## Sample Outputs

Codex can answer personalized questions such as:

- "What kind of engineer are you?"

*Umutesa is a multidisciplinary engineer who blends her technical skilld with creative problem-solving. With a BSc in Electrical and Computer Engineering, she approaches challenges with a systems-level mindset—balancing hardware constraints with software capabilities to build solutions that are not only functional but easy to use. Her creativity shines in the way she reimagines conventional approaches, often introducing innovative twists that make her work stand out. Whether it's designing embedded systems or designing intelligent solutions, she thrives at the intersection of feasibility and imagination.*

- "What are your strongest technical skills?"

Umutesa's technical toolkit is both broad and deep. She is fluent in Java, C++, and python. Her specialization in computer vision includes advanced techniques like object detection using YOLO, multi-object tracking, and people tracking in dynamic environments. She's particularly skilled at integrating vision algorithms with real-time systems, enabling autonomous agents to perceive and respond to their surroundings.

- "What projects or experiences are you most proud of?"

Umutesa's proudest achievement is her master's thesis: 'Vision-Only Socially Compliant Navigation for Autonomous Robots.' This project tackled one of the most complex challenges in robotics—predicting human trajectories in crowded, occluded environments using only visual input. She developed algorithms that allowed mobile robots to navigate safely and respectfully around people, even when visibility was limited and both the robot and pedestrians were in motion. The work required deep understanding of human behavior modeling, real-time vision processing, and ethical design principles. It was a culmination of late nights, relentless debugging, and a passion for creating technology that harmonizes with human spaces.

- "What do you value in a team or company culture?"

Umutesa thrives in environments that foster collaboration, mentorship, and continuous growth. She believes that the best ideas emerge when people feel safe to share freely. She values diversity and finds joy in learning from others' unique perspectives. Her ideal company culture celebrates shared success, encourages upskilling, and supports personal milestones. Whether it's a team sport, a birthday celebration, or a group hackathon, she sees these moments as the glue that binds a team together. Ultimately, she seeks a workplace that feels less like a job and more like a mission-driven community.

- "What's your approach to learning or debugging something new?"

Umutesa approaches debugging as a learning opportunity. She starts by identifying the root cause using first principles such as whether the tool/concept/algorithm is being used correctly, and consulting documentation to understand expected behavior. She believes that every bug is a clue to a deeper gap in understanding. Additionally, she uses platforms like CLion for C++ and VS Code for Python, leveraging their debugging modes to inspect variables, trace execution paths, and isolate anomalies. Her methodical approach combines intuition with discipline, and she often documents her findings to build a personal knowledge base for future

reference. For her, debugging isn't just about fixing, it's closing the gap in understanding.

- "What kind of tasks energize or drain me?"

Umutesa thrives on tasks that challenge her intellectually and creatively. She's most energized when exploring something new—whether it's a novel concept, a complex problem, or an unfamiliar tool. The excitement of discovery and growth fuels her motivation. On the flip side, she finds disorganization and poor communication draining. A lack of clarity or structure can disrupt her flow and diminish her enthusiasm.

- "How do I collaborate best with others?"

Umutesa collaborates best in environments that value open communication, mutual respect, and shared goals. She enjoys working with people who bring diverse perspectives and believes that collaboration is strongest when everyone feels heard. She's proactive in offering support, and thrives when there's a balance between independence and teamwork. Clear roles, constructive feedback, and a spirit of curiosity make her an engaged and thoughtful teammate.

- "Where do I need to grow?"

Umutesa sees growth as a lifelong journey. She believes there's always room to learn—whether it's mastering a new technical skill, deepening academic expertise, or building professional confidence. Right now, she's focused on strengthening her voice in professional spaces, refining her communication, and expanding her impact through mentorship and leadership. For her, growth is about evolving with purpose and staying curious.

## References

### 1. How to use RAG

- (a) <https://dev.to/shreshthgoyal/create-your-own-ai-rag-chatbot-a-python-guide-with-langchain-dfi>
- (b) [https://medium.com/@suraj\\_bansal/build-your-own-ai-chatbot-a-beginners-guide-to-rag-and-langchain-0189a18ec401](https://medium.com/@suraj_bansal/build-your-own-ai-chatbot-a-beginners-guide-to-rag-and-langchain-0189a18ec401)
- (c) <https://huggingface.co/blog/ngxson/make-your-own-rag>
- (d) Co-pilot was also used as an assistant

### 2. Parameter Tuning

- (a) <https://medium.com/data-science/a-guide-on-12-tuning-strategies-for-production-ready-rag-applications-7ca646833439>
- (b) <https://medium.com/data-science/a-guide-on-12-tuning-strategies-for-production-ready-rag-applications-7ca646833439>

### 3. Personalized

#### Streamlit UI

- a) <https://www.geeksforgeeks.org/python/a-beginners-guide-to-streamlit/>

### **Voice Processing**

- a) <https://www.geeksforgeeks.org/python/voice-assistant-using-python/>
- b) <https://medium.com/data-science/speech-to-text-to-speech-with-ai-using-python-a-how-to-guide-ee9b0b0ef082>