TECHNICAL REPORT: DESIGN AND IMPLEMENTATION OF AN EMBEDDING GENERATION

MADE_BY: Didhit Patel

Version 1.0 Date:16th February 2025

This report details the implementation of a Remix application interface that combines text and vision embedding capabilities to provide a multimodal user experience. The system is built using modern web framework REMIX and leverages large language models (LLMs) for embedding generation and interaction.

1. Core Components

Remix.js	Modern React-based framework with built-in routing, server-side rendering, and efficient data loading.	Used for developing the frontend user interface, providing a seamless and responsive user experience with efficient client-server communication
FastAPI (server) – API Endpoints - Text embeddings - Image Embeddings - Audio Embeddings - Advanced Text + image embedding - Chat User Interface	Native python, supports background task, Prebuilt docs, Easy to develop and use	This was developed so that frontend can communicate to an endpoint for different function.
Azure OpenAI Services - text-embedding-3-large - Whisper	Enterprise-grade AI services including text embedding and speech-to-text capabilities. Used to get API's for Generating Text Embeddings.	Implemented for text embeddings using text-embedding-3-large model and Whisper model for audio transcription and translation. (In free tier No Image embedding Model available.)
Fastembed - Qdrant/clip-ViT-B-32-vision - BAAI/bge-small-en-v1.5	Efficient embedding library supporting both text and image embedding generation. Provides Free API embeddings and easy to integrate and use.	Utilized for generating embeddings locally, providing a lightweight alternative to cloud-based solutions. Used for on premise text embeddings.
Groq - llama-3.2-11b-vision-preview	AI platform providing access to advanced language models. Used to get free Api of Models that are not available on azure free trials.	Integrated with llama-3.2-11b-vision-preview model for sophisticated vision-language processing tasks.

text-embedding-3-large	Azure's advanced text embedding model. Most advanced model for Embeddings.	Deployed on Azure services for generating high-quality text embeddings with enterprise-grade reliability.
Whisper	Azure's speech-to-text model Deployed on azure for audio translation.	Implemented for audio transcription and translation tasks, providing multilingual support
llama-3.2-11b-vision-preview	Good vision model available on hugging face used for making vision and text embedding for image search.	This was also available on azure foundry but can't be used with my plan so had to use it locally.
Qdrant/clip-ViT-B-32-vision	CLIP models are designed to work with both images and text, allowing you to generate embeddings for either modality in the same vector space.	Model was hosted on fastembed and used for Image embedding. The embeddings produced by this model are typically 512-dimensional vectors
BAAI/bge-small-en-v1.5	A lightweight embedding model for efficient text embedding generation.	Model was hosted on fastembed and used for text embedding.Produces 384-dimensional embeddings.
Docker -APP container 1.Remix App container 2.FastAPI app container	Container platform for application packaging and deployment	Used for containerizing both frontend and backend services, ensuring consistent deployment across environments.
Nginx	High-performance web server and load balancer	Implemented for load balancing and routing requests between multiple FastAPI instances.

2. Pipeline Workflow

1.Input Processing:

- Text, image, or audio inputs are received and preprocessed.

2.Embedding Generation:

- Inputs are converted into embeddings using appropriate models.

3.Context Retrieval:

- Embeddings are used to retrieve relevant context from Azure AI Search.

4.Response Generation:

- Retrieved context is augmented with user input and sent to Azure OpenAI or Groq for generating a response.

5.Docker Containerisation and Load Balancing Using NGNIX:

- The system is containerized using Docker, with separate containers for the frontend, backend, and load balancer to ensure modularity and scalability.
- Multiple backend containers handle FastAPI requests, while the frontend containers serve the user interface.
- Nginx acts as a reverse proxy and load balancer, distributing traffic efficiently among backend instances.
- The setup is managed using Docker Compose or Kubernetes, enabling auto-scaling and seamless service orchestration.

Embedding Pipelines:

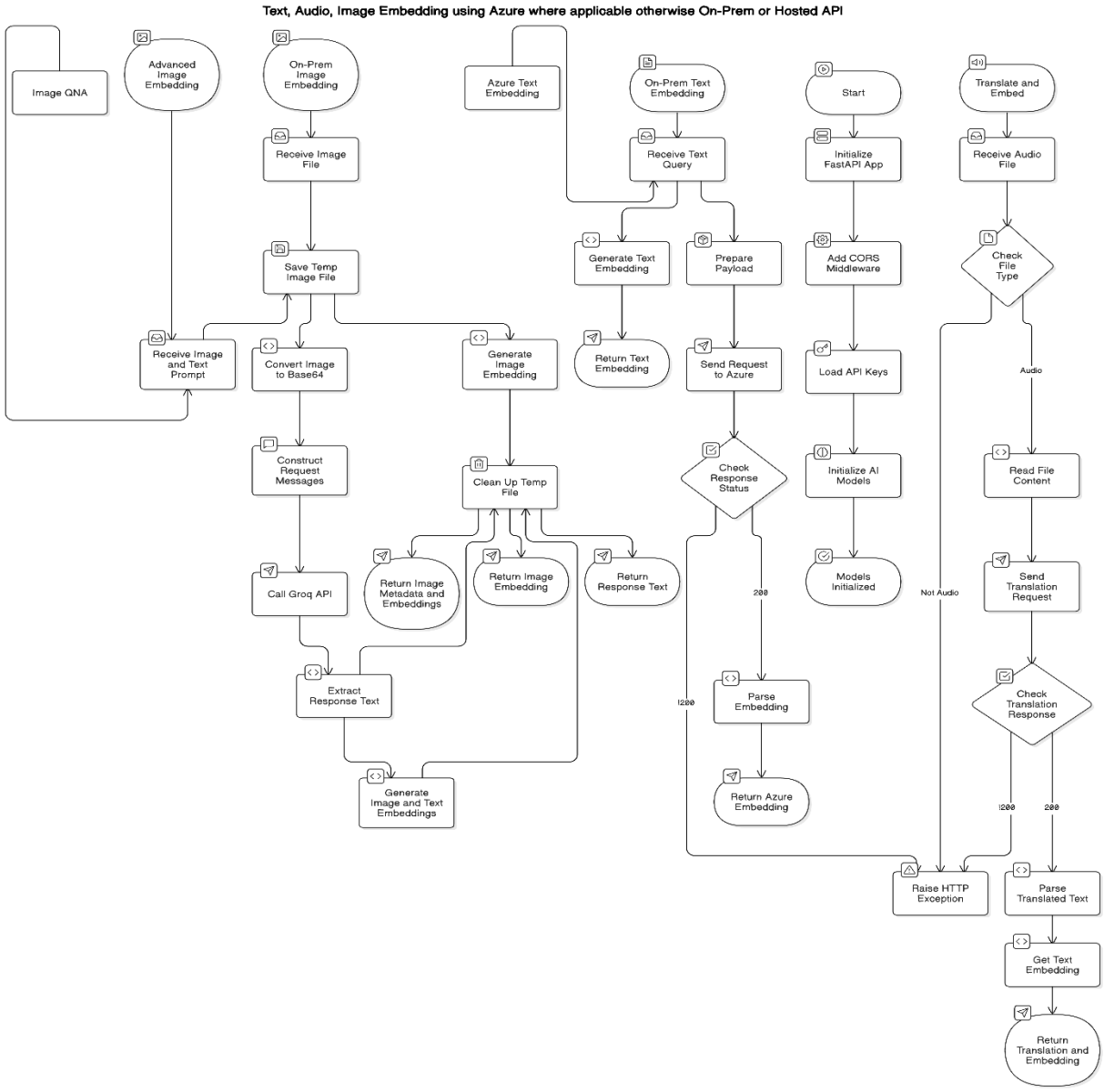

Figure: Multi-modal Embedding Pipeline

1. **Text Embedding Pipeline**:
    - inputs text via the frontend.
    - Text is sent to the FastAPI backend.
    - Embedding is generated using either:
        - Azure OpenAI (text-embedding-3-large) for cloud-based embeddings.
        - Fastembed (BAAI/bge-small-en-v1.5) for on-premise embeddings.

    -Embedding is returned to the frontend for display or further processing.
2. **Image Embedding Pipeline**:
    - User uploads an image via the frontend.
    - Image is processed by the FastAPI backend.
    - Embedding is generated using:
        - Fastembed (Qdrant/clip-ViT-B-32-vision) for on-premise image embeddings.
        - Groq (llama-3.2-11b-vision-preview) for advanced vision-language tasks.
    - Embedding is returned to the frontend.
3. **Audio Translation and Embedding Pipeline**:
    - User uploads an audio file.
    - Audio is transcribed and translated using Azure Whisper.
    - Translated text is embedded using Azure OpenAI or Fastembed.
    - Embedding is returned to the frontend.
4. Video Embedding [Fututre Aim]:
    - Upload and process a video:
    - Use the create method of the embed.task object to start a video embedding task.
    - We are using LLm model : "Marengo-retrieval-2.7".
    - Provide the video either as a publicly accessible URL or from your local file system.
    - Retrive frame based embeddings.-

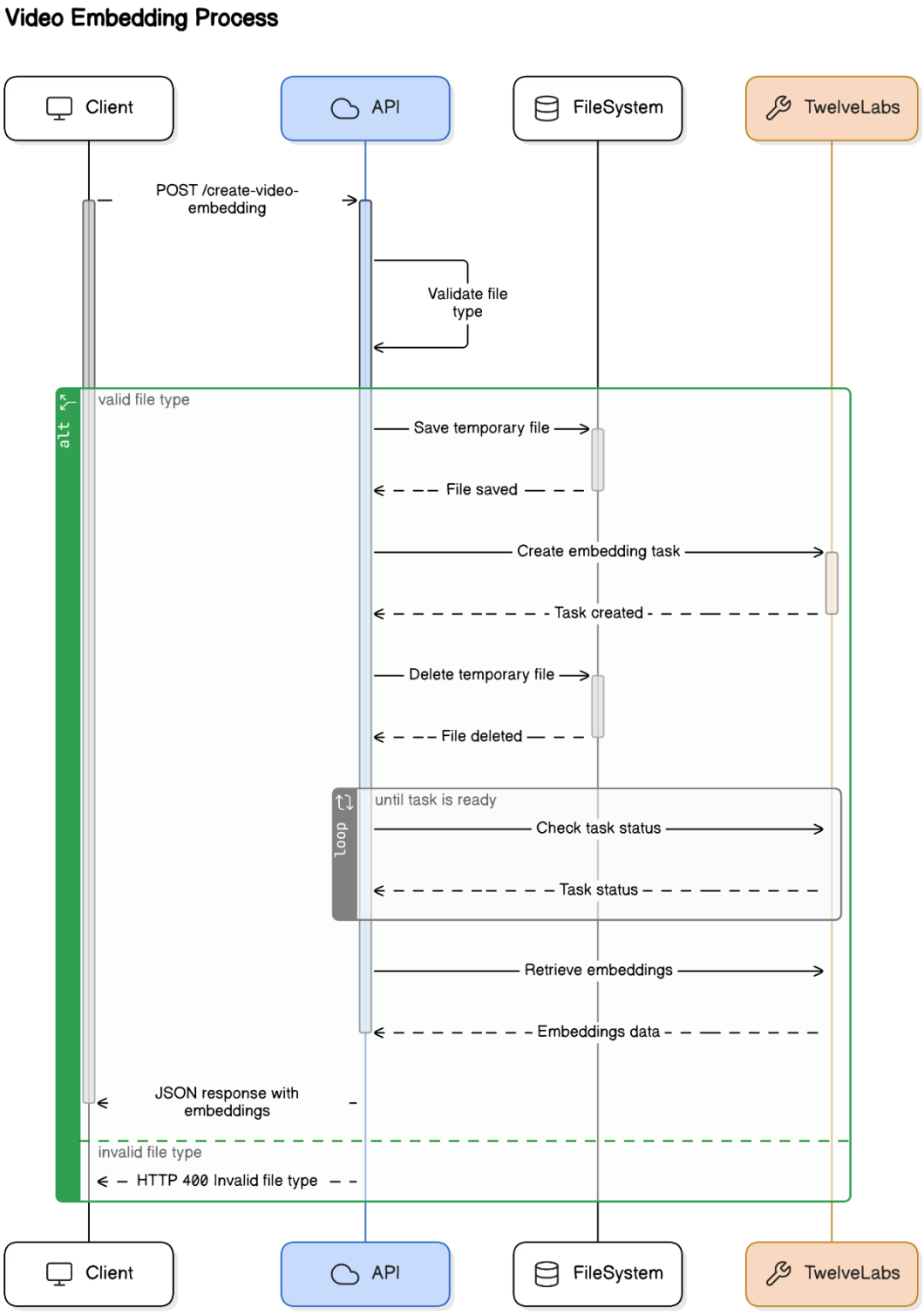

Figure: Video embedding Process workflow.

Query Processing Pipeline:

- **Text/Image input processing** – text was done by textract, images through tempfile.

- **Embedding generation:**
    - For Text: Azure openAi's text-embedding-3-large, BAAI/bge-small-en-v1.5
    - For Image: Qdrant/clip-ViT-B-32-vision, llama-3.2-11b-vision-preview
    - For Audio: Whisper Azure Ai for transcript and Above-mentioned Text models for embeddings.
- **Chat Query-Response:** The system processes user queries by generating embeddings, retrieving relevant context, and generating responses using Azure OpenAI or Groq.

3. Key Features

Text Embedding Generation

Hybrid Model Support:

- Cloud-Based: Generate embeddings using Azure OpenAI's text-embedding-3-large for high-accuracy semantic understanding.
- On-Premise: Use Fastembed (BAAI/bge-small-en-v1.5) for lightweight, privacy-focused text embeddings.

Image Embedding Generation

Multimodal Embedding Models:

- CLIP-ViT: Generate embeddings with Qdrant/clip-ViT-B-32-vision for semantic image understanding.
- Advanced Vision-Language Models: Use Groq (llama-3.2-11b-vision-preview) for complex tasks like image-to-text reasoning.

User Interface Features

1. Presentation Layer

- Remix (React-based): Leverages Remix's nested routing, server-side rendering, and built-in data loading for fast interactions.
- Styling: Uses Tailwind CSS for responsive design and Radix UI for accessible components.
- Real-Time Updates: Integrates WebSocket via Remix's useEffect and EventSource for bidirectional communication with the backend.

2. Multimodal Input Support

- **Text, Image, and Audio Processing**:
    - Text: Extracted and preprocessed using Textract or custom logic.
    - Image: Temporarily stored via tempfile for efficient handling.
    - Audio: Translated and transcribed using Azure Whisper for downstream embedding generation.
- **Unified Interface**: Supports simultaneous text, image, and audio inputs in a single chat interface.

4. Additional Features:

Audio Embedding Generation:

- User uploads an audio file.
- Audio is transcribed and translated using Azure Whisper.
- Translated text is embedded using Azure OpenAI or Fastembed.
- Embedding is returned to the frontend.

QNA with Image:

Multimodal Embedding Models:

- CLIP-ViT: Generate embeddings with Qdrant/clip-ViT-B-32-vision for semantic image understanding.
- Advanced Vision-Language Models: Use Groq (llama-3.2-11b-vision-preview) for complex tasks like image-to-text reasoning.

Advanced Multi-Containerisation and Orchestration:

Docker Bases Services:

- **Docker**-frontend-image: Remix app running in a container.
- **Backend**: Multiple FastAPI containers for parallel embedding generation.
- **Nginx**: Reverse proxy and load balancer for multiple instances.

Orchestration:

- Managed via Docker Compose (local) and NGNIX.

Enhanced UI for Embedding Management:

Copy & Download Embeddings:

- One-Click Copy: Users can copy embeddings to the clipboard via a button.
- Download as JSON/CSV: Embeddings are exported in structured formats for external analysis.

5. Implementation Details:

Backend-Key API Endpoints:

Here are the Key API Endpoints implemented in your FastAPI application, categorized by functionality:

Endpoint	Method	Description

/get-on-prem-text_embedding	GET	Generates **on-premise text embeddings** using fastembed (BAAI/bge-small-en-v1.5)
/image_embedding	POST	Generates **image embeddings** using CLIP-ViT (Qdrant/clip-ViT-B-32-vision)
/get-azure-embedding/	POST	Generates **cloud-based text embeddings** using Azure OpenAI (text-embedding-3-large)
/advanced_image_embedding	POST	Combines **LLaMA vision model** with CLIP embeddings for image metadata + multi-modal analysis
/translate_and_embed/	POST	Processes audio files to **translate speech-to-text** and generate text embeddings
/Image_QNA	POST	Performs **vision-language Q&A** using LLaMA-3.2-11b-vision-preview with image/text inputs

Frontend Implementation

- Remix based interface with session state management.
- Responsive design with sidebar navigation.
- Real-time chat interface with message history.
- Chat Interface for User Interaction.
- Routing, styling and route implementation.

Docker containerisation & Orchestration Guide

- **Docker Setup**
    - Frontend Container: Node.js image for Remix app with production build.
    - Backend Container: Python image for FastAPI with dependency installation.
    - Nginx Proxy: Configure reverse proxy for routing and load balancing.
- **Orchestration**

- Docker Compose: Define services (frontend, backend, Nginx) and network connections.

6. Deployment Considerations

Infrastructure Requirements

Azure OpenAI service access

Azure ML Studio

Deployed LLMs API Keys

Scaling Considerations

Horizontal scaling of FastAPI servers

Load balancing requirements

7. Future Improvements

1. Video Embedding Generation

- **Implementation**:
  - Integrate video processing models for frame- level embeddings.
  - Add chunking for long videos and temporal modeling for context retention.
- **Storage**: Optimize for large files with streaming support.

2. Kubernetes Deployment

- **Orchestration**:
  - Migrate from Docker Compose to Kubernetes for auto-scaling, rolling updates, and self-healing.

3. Universal Document Support

- **Parsers**: Add support for PDF (PyMuPDF), Word (python-docx), PPT (python-pptx), and OCR (Tesseract).
- **Chunking**: Implement layout-aware chunking for complex documents.

4. Error Recovery & Resilience

- **Strategies**:
  - Retry policies with exponential backoff for API calls.
  - Circuit breakers for downstream services.
  - State snapshots for session recovery.

5. Vector Database (Neo4j GraphRAG)

Integration:

    - Store embeddings as graph nodes with semantic relationships.
    - Visualize connections via Neo4j Bloom for explainability.

- **Querying**: Enable path-based retrieval for complex reasoning.

8. Conclusion

The **Embedding Generation System** successfully demonstrates a robust, multimodal approach to knowledge retrieval and generation, combining text, image, and audio processing into a unified platform. By integrating **Remix** for a responsive frontend, **FastAPI** for efficient backend services, and hybrid cloud/on-prem models (Azure OpenAI, Fastembed, Groq), the system achieves flexibility across deployment scenarios while maintaining high performance.

Key achievements include:

- **Multimodal Capability**: Seamless handling of text, images, and audio with context-aware retrieval using Azure AI Search.
- **Scalable Architecture**: Docker containerization and Nginx load balancing ensure modularity and horizontal scalability.
- **Cost-Effective Hybrid Workflows**: Leveraged free-tier models (Fastembed, Groq) alongside enterprise-grade Azure services for optimized resource use.