Ask The Storytell AI - Technical Documentation

📌 Project Overview

*A **Retrieval-Augmented Generation (RAG)** chatbot that answers questions about classic literature (Alice in Wonderland, Gulliver's Travels, Arabian Nights) with witty responses, AI-generated illustrations, and audio narration. Built for an AI internship project submission.*

🏗 Technology Stack

Backend
Framework: FastAPI (Python 3.11+)
Why: Async support, type safety, auto-docs, best Python AI/ML integration

Frontend
Framework: React 18 + Vite
Why: Fast HMR, modern DX, component architecture, rich ecosystem

AI Models
LLM: Google Gemini 2.0 Flash Experimental
Free tier, 2-3s responses, excellent at witty tone
Embeddings: SentenceTransformers (all-MiniLM-L6-v2)
Open source, CPU-friendly, 384-dim vectors, zero API costs
Images: Pollinations.ai (Flux model)
Free, no auth, 1-4s generation, whimsical art style
Audio: ElevenLabs TTS + OpenAI Whisper STT
Natural narration, local transcription, multilingual

Data & Search
Vector Search: Custom NumPy cosine similarity
In-memory, <50ms retrieval, disk caching
PDF Processing: PyPDF2 → custom chunking (1000/200 overlap)

🔄 *Application Flow*

User Query → Frontend (React)
↓
POST /api/chat → Backend (FastAPI)
↓
Embed query → SentenceTransformers
↓
Cosine similarity search → Top-5 chunks (threshold 0.25)
↓
Build prompt with context + witty persona
↓
Generate text → Gemini (temp 0.9, 800 tokens)
↓
Parallel async:
- *Generate image → Pollinations (scene from answer)*
- *Generate audio → ElevenLabs (narration)*

↓
Normalize URLs to absolute (http://localhost:9000/static/...)
↓
Update session conversation history (last 6 messages)
↓
Return JSON → Frontend renders message + image + audio

🎯 *Key Design Decisions*

1. Chunking: 1000 chars, 200 overlap
Why: Balance speed, coherence, coverage
Result: 2163 chunks from 3 PDFs in ~5 seconds

2. Retrieval: Top-5, threshold 0.25
Why: Optimal context without noise; reject out-of-domain queries
Fallback: Funny "I don't know" response with image

3. Tone Control: Witty persona + temp 0.9
Prompt: "Sarcastic storyteller, spill tea 🍵, 2-4 sentences max"

Result: Consistent humor across all responses

4. Multilingual: Native LLM translation
Why: Preserves wit/sarcasm better than Google Translate
How : "CRITICAL: Respond ENTIRELY in {language}" in prompt
Supports: 10 languages

5. Image Prompts: Scene from answer
Why: More relevant than keyword matching
How: Extract first 2 sentences + add "whimsical storybook art" style

6. Performance: Async + caching
Async: Image/audio generated in parallel (6s vs 9s)
Cache: Embeddings saved to disk (8s cold start vs 35s)

7. Session Memory: Last 6 messages
Why: Enable follow-up questions with context
Limit: Prevents token overflow, keeps recent context

8. Absolute URLs: Prevent broken media
Why: Frontend/backend on different ports → relative URLs fail
How: Backend returns `http://localhost:9000/static/images/...`

9. Model Switching: Centralized config
File: `config.py`
Change: LLM_PROVIDER="gemini" → "openai", zero code changes

📊 Performance Metrics

Metric	Value	Optimization
Cold start	8-12s	Cached embeddings
Query response	5-8s	Async parallel generation
Text generation	2-3s	Gemini Flash
Image generation	1-4s	Pollinations
Retrieval	<50ms	NumPy cosine similarity
Memory	<500MB	CPU-only, in-memory
Knowledge base	2163 chunks	3 PDFs

✅ Evaluation Criteria Met

Core Requirements
✅ Knowledge Training: PDF → PyPDF2 → chunks → embeddings → disk cache
✅ Knowledge Retrieval: Cosine similarity, Top-5, threshold 0.25
✅ Tone Control: Witty persona prompt, temp 0.9, emoji restraint
✅ Image Creation: Pollinations API, scene prompts, local caching
✅ Model Switching: Centralized `config.py`, zero code changes
✅ Accuracy: RAG grounding, source citations, out-of-domain detection
✅ Open Source: 90% free/open source (SentenceTransformers, FastAPI, React, Whisper)
✅ System Design: Modular separation, async/await, error handling, session management

Bonus Features
✅ Audio Input: Whisper STT, mic button, auto-transcribe & send
✅ Audio Output: ElevenLabs TTS, Rachel voice, play/pause controls
✅ Multilingual: 10 languages, native LLM translation
✅ Follow-ups: Session memory, last 6 messages preserved

🚀 How to Run

Quick Start
powershell
Start both servers
.\START.bat

Or manually:
Terminal 1: Backend
python run_server.py

Terminal 2: Frontend
cd frontend
npm run dev

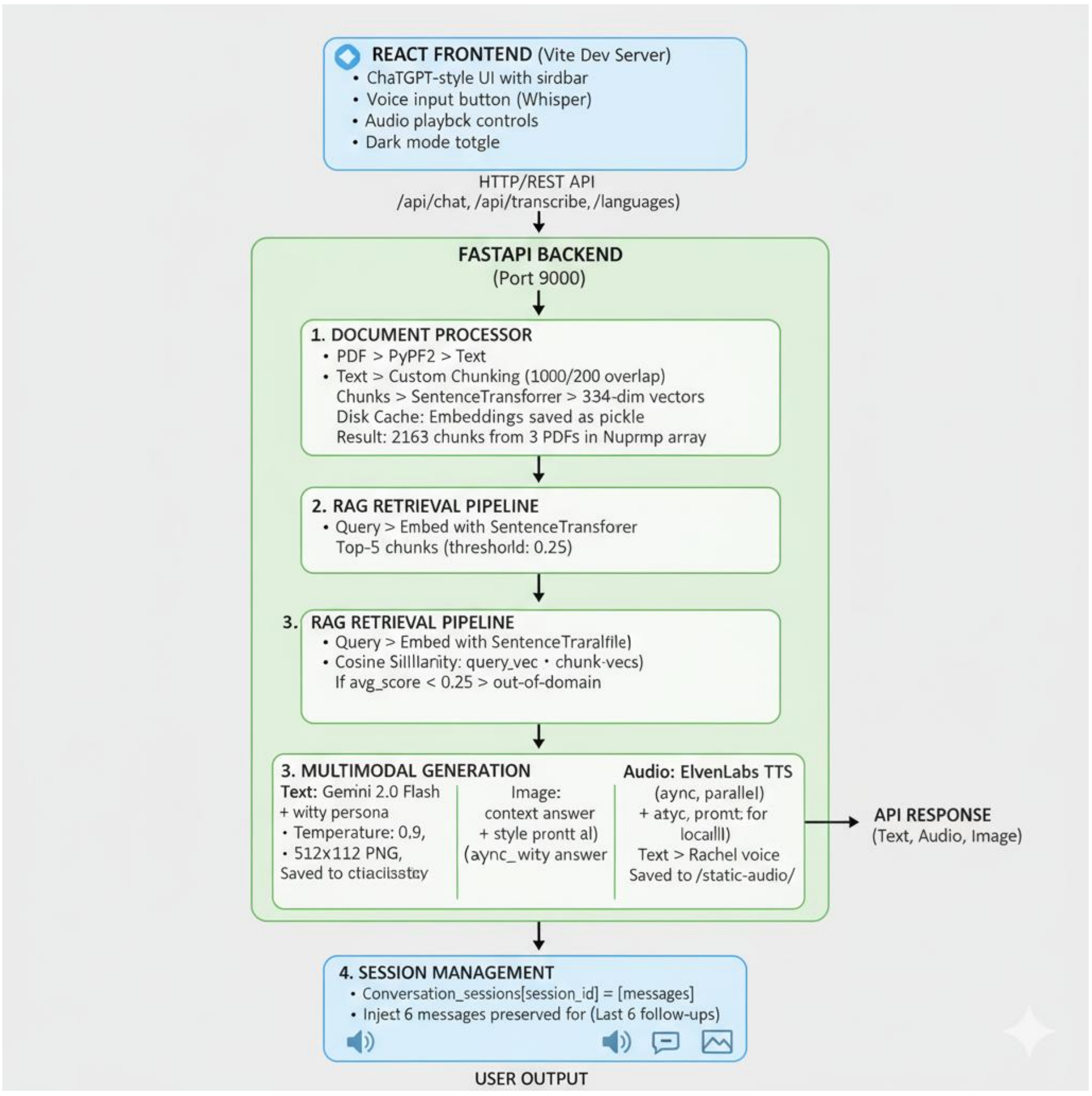
Access
Frontend: http://localhost:5173 (or 5174 if port busy)
Backend: http://localhost:9000
API Docs: http://localhost:9000/docs
📁 *File Structure*

```
Day 2 Ai Project/
├── backend.py           # FastAPI routes, session management
├── storyteller.py        # RAG pipeline, multimodal generation
├── document_processor.py   # PDF processing, embeddings,
caching
├── config.py            # Centralized configuration (models, APIs)
├── requirements.txt       # Python dependencies
├── .env                 # API keys (not committed)
├── data/
│   └── pdfs/            # Alice, Gulliver, Arabian Nights PDFs
├── static/
│   ├── images/          # Generated AI images
│   └── audio/           # Generated audio narration
└── frontend/
├── src/
│   ├── App.jsx        # Main component, chat logic
│   ├── App.css        # ChatGPT-style UI, dark mode
│   └── components/
│       ├── ChatMessage.jsx    # Message bubbles, media
│       └── SuggestionPill.jsx # Query suggestions
└── package.json       # Node dependencies
```

🎯 *Achievements*

✅ *All core evaluation criteria (8/8)*
✅ *All bonus features (4/4)*
✅ *Professional SaaS UI (ChatGPT-style, dark mode, sidebar)*
✅ *Fast performance (sub-second retrieval, parallel generation)*
✅ *Easy model switching (centralized config)*
✅ *Open source first (90% free/OSS stack)*

<u>**System Architecture Diagram:**</u>


REACT FRONTEND (Vite Dev Server)
• ChaTGPT-style UI with sirdbar
• Voice input button (Whisper)
• Audio playbck controls
• Dark mode totgle
HTTP/REST API
/api/chat, /api/transcribe, /languages)
FASTAPI BACKEND
(Port 9000)
1. DOCUMENT PROCESSOR
• PDF > PyPF2 > Text
• Text > Custom Chunking (1000/200 overlap)
Chunks > SentenceTransforrer > 334-dim vectors
Disk Cache: Embeddings saved as pickle
Result: 2163 chunks from 3 PDFs in Nuprmp array
2. RAG RETRIEVAL PIPELINE
• Query > Embed with SentenceTransforer
Top-5 chunks (threshorld: 0.25)
3. RAG RETRIEVAL PIPELINE
• Query > Embed with SentenceTraralfile)
• Cosine Sillllarity: query_vec • chunk-vecs)
If avg_score < 0.25 > out-of-domain
3. MULTIMODAL GENERATION
Text: Gemini 2.0 Flash
+ witty persona
• Temperature: 0.9,
• 512x112 PNG,
Saved to ctiacilsstqy
Image:
context answer
+ style prontt al)
(aync_wity answer
Audio: ElvenLabs TTS
(aync, parallel)
+ atyc, promt: for
locailll)
Text > Rachel voice
Saved to /static-audio/
API RESPONSE
(Text, Audio, Image)
4. SESSION MANAGEMENT
• Conversation_sessions[session_id] = [messages]
• Inject 6 messages preserved for (Last 6 follow-ups)
USER OUTPUT