AI Joke Generator

Documentatie proiect

Aplicatie desktop pentru generarea de glume
cu inteligenta artificiala locala

Materie: Bazele interfeţei om – maşină

Echipa:
Moroiu Eric-Gabriel
Mitu Rareș
Ianoș Luca

Grupa: 443A

Cuprins

1. Descrierea proiectului

AI Joke Generator este o aplicatie desktop care genereaza glume folosind un model de limbaj (LLM) care ruleaza local prin Ollama. Aplicatia ofera o interfata grafica moderna cu dark theme, construita in Tkinter, si vine cu functionalitati suplimentare precum citirea glumelor cu voce (text-to-speech) si introducerea contextului prin dictare vocala (speech-to-text).

Utilizatorul introduce un subiect sau cateva cuvinte cheie, alege limba (engleza sau romana), tonul dorit (curat, dark sau sarcastic) si numarul de glume. Aplicatia trimite un prompt structurat catre modelul AI local, parseaza raspunsul si afiseaza glumele in interfata.

Proiectul nu depinde de servicii cloud pentru generarea de text - totul ruleaza pe masina utilizatorului prin Ollama. Singurele servicii externe folosite sunt Google TTS (pentru sinteza vocala) si Google Speech API (pentru recunoasterea vocala), ambele fiind optionale.

2. Cerinte si instalare

Cerinte de sistem

- Python 3.9 sau mai nou
- Ollama instalat si functional (https://ollama.ai)
- Un model descarcat in Ollama (implicit: llama3.2)
- Conexiune la internet pentru TTS si STT (optional)

Pasi de instalare

Se creaza un virtual environment, se instaleaza dependentele si se descarca modelul:

```
python -m venv venv

source venv/bin/activate

pip install requests gtts python-dotenv SpeechRecognition pyaudio

ollama pull llama3.2
```

```
ollama serve

python main.py
```

Pe macOS, pentru input vocal trebuie instalat si portaudio: brew install portaudio. Pe Linux: sudo apt install portaudio19-dev. Pe Windows, PyAudio se instaleaza direct fara dependente extra.

Dependente Python

Pachet	**Versiune**	**Scop**
requests	>=2.28.0	Cereri HTTP catre Ollama API
gtts	>=2.4.0	Google Text-to-Speech
python-dotenv	>=1.0.0	Incarcare variabile din .env
SpeechRecognition	>=3.10.0	Recunoastere vocala
pyaudio	>=0.2.14	Acces la microfon

3. Structura proiectului

Fisier	Rol
main.py	Punct de intrare - verificari si lansare aplicatie
gui.py	Interfata grafica Tkinter (dark theme)
config.py	Setari centralizate (URL, model, limbi, tonuri)
prompts.py	Template-uri de prompt pentru modelul LLM
joke_generator.py	Integrare cu API-ul Ollama
text_processing.py	Analiza text (limba, keywords, statistici)
tts_engine.py	Text-to-Speech cu gTTS
stt_engine.py	Speech-to-Text cu SpeechRecognition
utils.py	Functii utilitare (validare, formatare, salvare)
pyproject.toml	Configurare pachet si dependente

4. Fluxul aplicatiei

La pornire, main.py verifica daca dependentele sunt instalate si daca Ollama ruleaza. Daca totul e in regula, lanseaza interfata grafica. Daca Ollama nu e pornit, aplicatia porneste oricum dar utilizatorul trebuie sa apese "Connect" manual.

Fluxul principal de generare a glumelor:

1. Utilizatorul introduce un context (text sau voce)

2. Se valideaza inputul (lungime 2-500 caractere, contine litere/cifre)

3. Se analizeaza textul: detectare limba, extragere cuvinte cheie, statistici

4. Se construieste promptul pentru LLM (system prompt + ton + exemplu + context)

5. Se trimite cererea POST catre Ollama API cu parametrii de generare

6. Se parseaza raspunsul in glume individuale

7. Se afiseaza glumele in interfata

8. Optional: citit cu voce (TTS) sau salvare in fisier text

5. Descrierea modulelor

5.1. main.py

Punctul de intrare al aplicatiei. Contine trei functii:

check_deps() - verifica daca pachetele necesare (requests, gtts, dotenv) sunt instalate. Foloseste __import__() pentru a testa importurile dinamic. Daca ceva lipseste, afiseaza comanda de instalare si returneaza False.

check_ollama() - face un GET rapid la Ollama pentru a verifica daca serverul raspunde. Are timeout de 2 secunde ca sa nu blocheze lansarea.

main() - orchestreaza verificarile si lanseaza GUI-ul. Importa gui.run_app() doar dupa ce confirma ca dependentele exista (import "lazy"). Prinde KeyboardInterrupt pentru iesire curata cu Ctrl+C.

5.2. config.py

Fisier scurt care centralizeaza toate setarile aplicatiei. Incarca variabile din fisierul .env (daca exista) prin python-dotenv, permitand suprascrierea setarilor fara a modifica codul.

Variabila	**Valoare**	**Descriere**
OLLAMA_BASE_URL	http://localhost:11434	Adresa serverului Ollama
OLLAMA_MODEL	llama3.2	Modelul folosit pentru generare

REQUEST_TIMEOUT	60	Timeout in secunde pentru cereri HTTP
GENERATION_CONFIG	temperature=0.9, top_p=0.95, num_predict=1024	Parametri de generare
SUPPORTED_LANGUAGES	English, Romanian	Limbile disponibile
JOKE_TONES	Clean, Dark, Sarcastic	Tonurile de umor
MIN/MAX/ DEFAULT_JOKES	1 / 10 / 3	Limite numar glume

5.3. gui.py

Cel mai mare fisier din proiect - contine toata interfata grafica si logica de interactiune. Foloseste Tkinter cu un dark theme personalizat.

Clasa ModernStyle defineste paleta de culori si fonturile. Culorile sunt inspirate din tema Tokyo Night: fundal inchis (#1a1b26), accent albastru (#7aa2f7), verde pentru succes, rosu pentru erori.

Clasa JokeGeneratorApp este componenta principala. La initializare configureaza fereastra (700x800, centrata pe ecran), aplica stilurile, initializeaza serviciile (TTS, STT) si construieste toate elementele UI.

Interfata e impartita in sectiuni de tip "card":

- Connection - status conexiune Ollama si buton Connect
- Input - camp text pentru context + buton Voice Input
- Options - slider numar glume, radiobuttons limba, combobox ton, buton Generate
- Output - zona de afisare glume + butoane Read Aloud, Stop, Save, Clear

- Analysis - statistici text (numar cuvinte, limba detectata, keywords)

Operatiile lente (conectare, generare, TTS, STT) ruleaza pe thread-uri daemon separate pentru a nu bloca interfata. Rezultatele se trimit inapoi pe thread-ul principal prin root.after(0, callback) - mecanismul standard Tkinter pentru comunicare intre thread-uri.

Layoutul foloseste un Canvas scrollabil cu un Frame interior - solutia standard in Tkinter pentru continut mai lung decat fereastra. Suporta scroll cu rotita mouse-ului pe toate platformele.

5.4. joke_generator.py

Gestioneaza comunicarea cu API-ul Ollama. Clasa JokeGenerator se conecteaza la server, verifica ca modelul cerut exista, si trimite cereri de generare.

La instantiere, face un GET la /api/tags pentru a obtine lista modelelor disponibile. Daca serverul nu raspunde sau modelul nu exista, arunca JokeGeneratorError cu instructiuni clare (ex: "Run: ollama pull llama3.2").

Metoda generate_jokes() construieste promptul prin prompts.build(), trimite un POST la /api/generate cu parametrii de generare (temperature, top_p, num_predict) si parseaza raspunsul. Returneaza un dictionar cu success, jokes, raw_response si error.

Parsarea (_parse_jokes) imparte textul brut in glume individuale dupa: linii goale, prefixe numerotate (1., 2.) sau liniute. Filtreaza fragmentele mai scurte de 10 caractere care sunt probabil artefacte de formatare.

5.5. prompts.py

Contine template-urile de prompt trimise catre model. Separat de config.py pentru claritate.

Defineste trei constante principale: SYSTEM (personalitatea LLM-ului - comedian profesionist), TONES (descrieri ale fiecarui stil de umor per limba) si EXAMPLES (exemple concrete de glume pentru fiecare combinatie ton+limba, ca referinta pentru model).

Functia build() asambleaza promptul final: system prompt + descriere ton + exemplu + contextul utilizatorului + instructiuni stricte de formatare. Foloseste cuvinte cheie ca "EXACTLY" si "IMPORTANT" pentru a creste compliance-ul modelului cu formatul cerut.

5.6. text_processing.py

Modul de analiza a textului introdus de utilizator. Rezultatele sunt afisate in sectiunea Analysis din interfata.

Functii principale:

- tokenize(text) - transforma textul in lowercase, elimina punctuatia, imparte in cuvinte
- detect_language(text) - detecteaza limba (en/ro) bazat pe cuvinte comune si caractere speciale romanesti
- extract_keywords(text) - elimina stop words si returneaza cele mai frecvente cuvinte
- get_text_stats(text) - calculeaza: numar cuvinte, caractere, propozitii, cuvinte unice, lungime medie
- analyze_input(text) - functia principala care combina toate celelalte

Detectarea limbii functioneaza prin scor: numara cate cuvinte din text se potrivesc cu pattern-urile fiecarei limbi si adauga puncte pentru caractere speciale (a, i, t, s). Limba cu scorul cel mai mare castiga.

5.7. tts_engine.py

Converteste textul in vorbire folosind Google gTTS. Genereaza un fisier MP3 temporar si il reda cu player-ul audio al sistemului de operare.

Fluxul: text -> gTTS genereaza audio -> salvat in fisier temporar .mp3 -> detectare player (afplay pe macOS, mpg123 pe Linux, PowerShell pe Windows) -> redare in thread de fundal -> callback la terminare.

Suporta oprire manuala prin metoda stop() care termina subprocess-ul de redare. La inchiderea aplicatiei, cleanup() opreste redarea si sterge fisierele temporare.

5.8. stt_engine.py

Converteste input vocal (microfon) in text folosind SpeechRecognition si Google Speech API.

La initializare verifica ca SpeechRecognition si PyAudio sunt instalate, configureaza recognizer-ul (energy_threshold=300, dynamic threshold, pause_threshold=0.8s) si testeaza accesul la microfon.

Metoda listen() ruleaza pe un thread daemon: calibreaza pentru zgomot ambient (0.5s), inregistreaza cu timeout configurabil, trimite audio-ul la Google Speech API si returneaza textul prin callback-ul on_result. Erorile (timeout, audio neclar, eroare retea) sunt trimise prin on_error.

5.9. utils.py

Functii helper folosite in mai multe locuri din aplicatie:

- validate_context() - verifica inputul: non-gol, 2-500 caractere, contine litere/cifre
- format_jokes_for_display() - formateaza glumele pentru afisare (numerotare uniforma, linii goale)
- format_jokes_for_tts() - formateaza pentru citit cu voce (elimina markdown, adauga "Joke number X")
- save_jokes_to_file() - salveaza in fisier .txt cu header (timestamp, limba, context)
- get_language_code() - conversie "English" -> “en"

6. Arhitectura si design

Principii de design aplicate in proiect:

UI single-threaded: Interfata Tkinter ruleaza pe thread-ul principal. Toate operatiile lente ruleaza pe thread-uri daemon separate pentru a nu bloca interfata.

Comunicare intre thread-uri: Thread-urile de fundal nu modifica direct widgeturile. Folosesc root.after(0, callback) pentru a programa actualizarile pe thread-ul principal.

Exceptii custom: Fiecare modul are propria exceptie (JokeGeneratorError, TTSEngineError, STTEngineError) pentru tratare diferentiata a erorilor.

Configurare centralizata: Toate setarile sunt in config.py, modificabile dintr-un singur loc. Se pot suprascrie prin fisier .env fara a modifica codul.

Cross-platform: Detectare automata a platformei (macOS/Windows/Linux) pentru redare audio. Interfata Tkinter functioneaza nativ pe toate sistemele.

Fara baze de date: Totul in memorie. Salvarea glumelor e optionala, in fisiere .txt simple.

Servicii optionale: TTS si STT sunt optionale. Daca bibliotecile lipsesc, aplicatia functioneaza fara ele - butoanele sunt dezactivate.

7. Utilizare

Pasi pentru utilizarea aplicatiei:

1. Porneste Ollama cu comanda: ollama serve

2. Ruleaza aplicatia: python main.py

3. Apasa "Connect to Ollama" daca nu s-a conectat automat

4. Introdu un subiect in campul de context (ex: "programming", "pisici", "scoala")

5. Selecteaza numarul de glume (slider 1-10), limba si tonul dorit

6. Apasa "Generate Jokes" si asteapta raspunsul

7. Optional: "Read Aloud" pentru a auzi glumele citite cu voce

8. Optional: "Save to File" pentru a salva intr-un fisier text

Aplicatia suporta si input vocal - apasand butonul "Voice Input" se activeaza microfonul si textul dictat se adauga automat in campul de context.