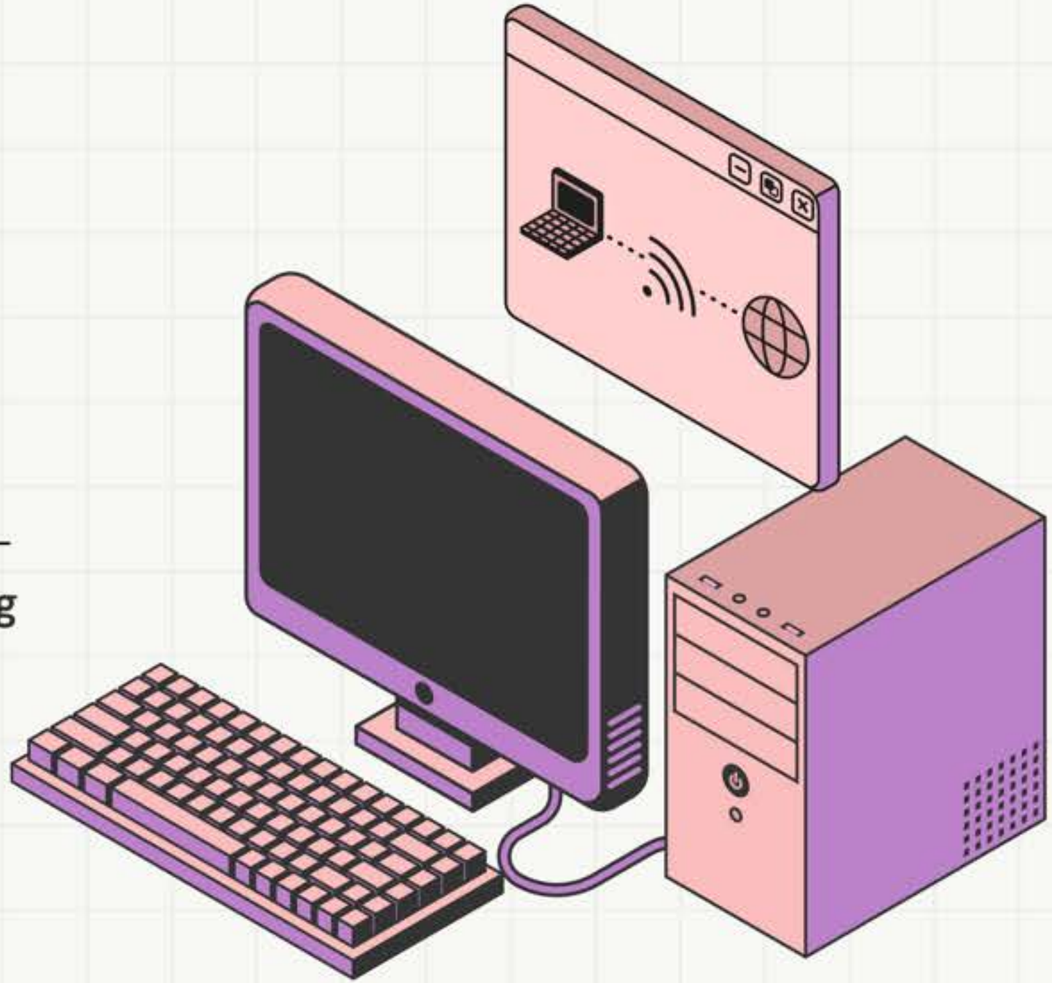
CLOUD-BASED

MULTIMODAL LANGUAGE PROCESSING SYSTEM

Text Processing | Speech Recognition | Image Captioning



Under the Guidance of Prof. Animesh Chaturvedi

Presented by :-

- **Pakhi Singhal** [22bds042]
- **Preethi Varshala** [22bds045]
- **Ravi Raj** [22bds051]

PROBLEM STATEMENT

Traditional language processing tools are often monomodal and lack the depth required for nuanced understanding across diverse data types (text, speech, image). these systems produce shallow outputs, lack contextual awareness, and are rarely scalable.

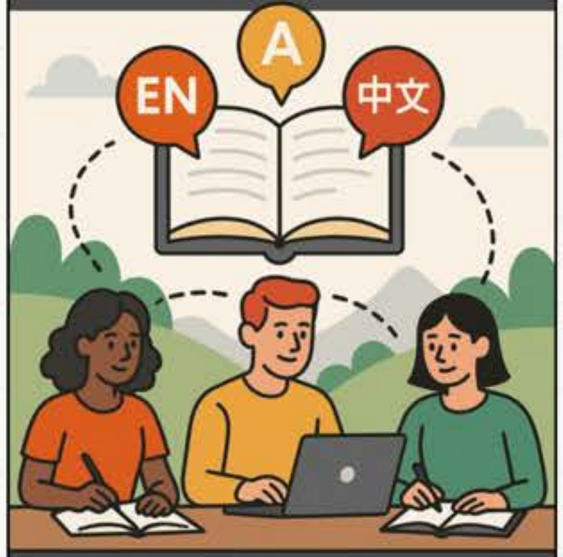
PROJECT OBJECTIVE

To develop an integrated, multilingual web application that leverages advanced AI services—such as real-time text translation, speech-to-text conversion, text-to-speech synthesis, and image captioning—to break communication barriers, enhance accessibility, and deliver seamless user interaction across diverse languages and media formats.

USE CASES AND IMPACT

EDUCATION

Language learning across regions



HEALTHCARE

Multilingual voice assistants

ACCESSIBILITY

Helps visually or hearing-impaired users interact via AI

SOCIAL MEDIA

Auto-captioning for visual content

CLOUD COMPUTING BENEFITS

- **Scalability for multimodal AI tasks**

  efficiently handles varying workloads across text, speech, and image processing pipelines.

- **Enhanced security**

  ensures robust protection of user data and media inputs using secure cloud protocols.

- **Operational efficiency**

  automates resource allocation for backend services like stt, tts, translation, and visual reasoning.

- **Elimination of infrastructure overhead**

  allows the team to focus on improving ai models instead of managing servers or compute resources.

- **Demand-based scaling**

  scales compute and storage based on real-time usage, maintaining responsiveness during heavy tasks.

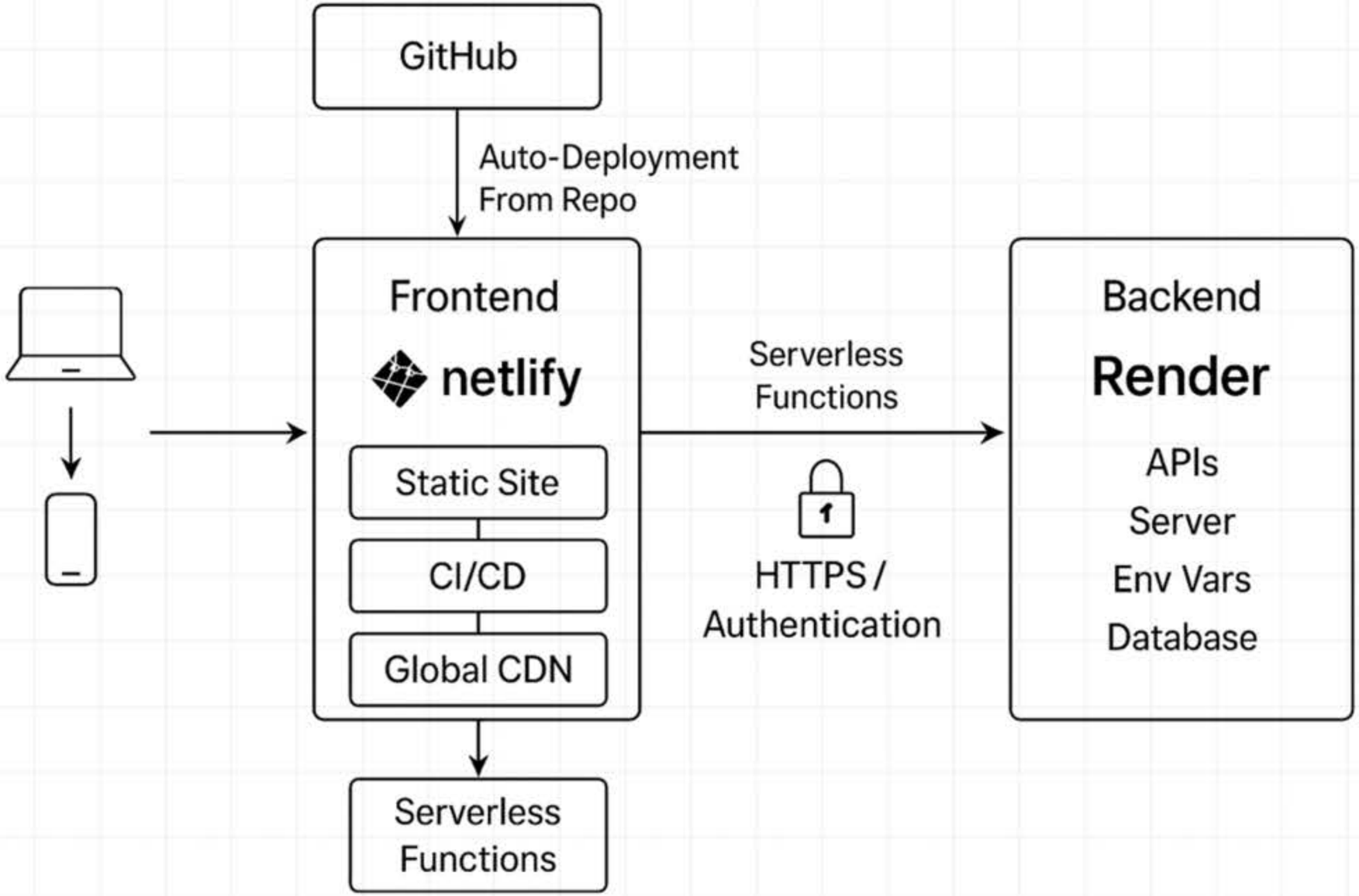
- **Consistent execution environments**

  provides reproducible and reliable deployments for ai components through containerization.

CLOUD IMPLEMENTATION

- **Frontend Deployment :-** The React.js frontend is hosted on Netlify, which supports automatic builds, global CDN delivery, and serverless functions for dynamic content handling.
- **Backend Deployment :-** The Flask backend is deployed on Render, a modern PaaS that supports web services, APIs, and background workers with built-in CI/CD pipelines.
- **High Availability :-** Both platforms offer global infrastructure, auto-scaling, and failover support, ensuring uninterrupted access to translation, STT, TTS, and image captioning services.
- **Serverless & Resource Optimization :-** Netlify's serverless functions and Render's autoscaling eliminate the need for manual infrastructure management, automatically allocating compute based on demand.
- **Cost Efficiency :-** Using Netlify and Render allows for a cost-effective, usage-based pricing model, helping reduce operational costs by only paying for active usage.

CLOUD DEPLOYMENT STRATEGY



TECH STACK

Layer	Technology
Frontend	ReactJS (JSX, Hooks, Forms)
Backend	Flask (Python REST API)
Cloud APIs	Google Cloud: STT, TTS, Translate
ML Models	HuggingFace ViT-GPT2 (Image Captioning)
Other Libs	• gtts, googletrans, • speech_recognition, • transformers, • torch, • PIL
Communication	fetch + Flask-CORS

SYSTEM ARCHITECTURE

Feedback Capture Module

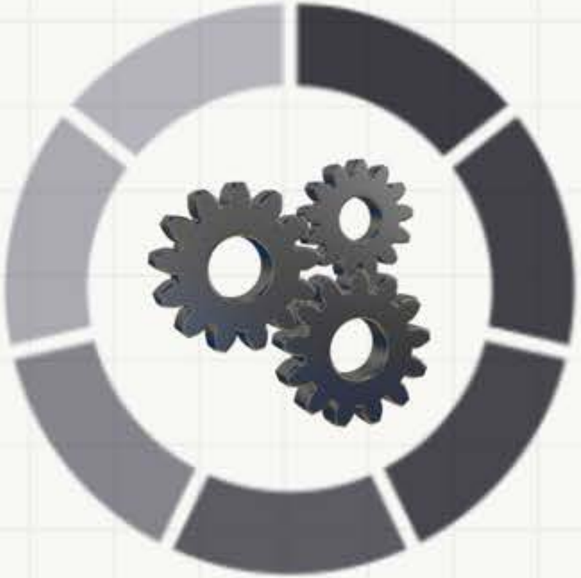
records user feedback to assess and improve system performance over time.

Frontend Interface

built with React.js, it enables users to submit text, speech, or image inputs.

Speech Processing Module

employs Google Cloud Speech-to-Text API to convert spoken input into analyzable text.



Input Handler (Flask Backend)

receives frontend requests and directs data to the appropriate processing pipeline.

Image Understanding Module

uses Hugging Face BLIP to extract contextual captions and descriptions from image input.

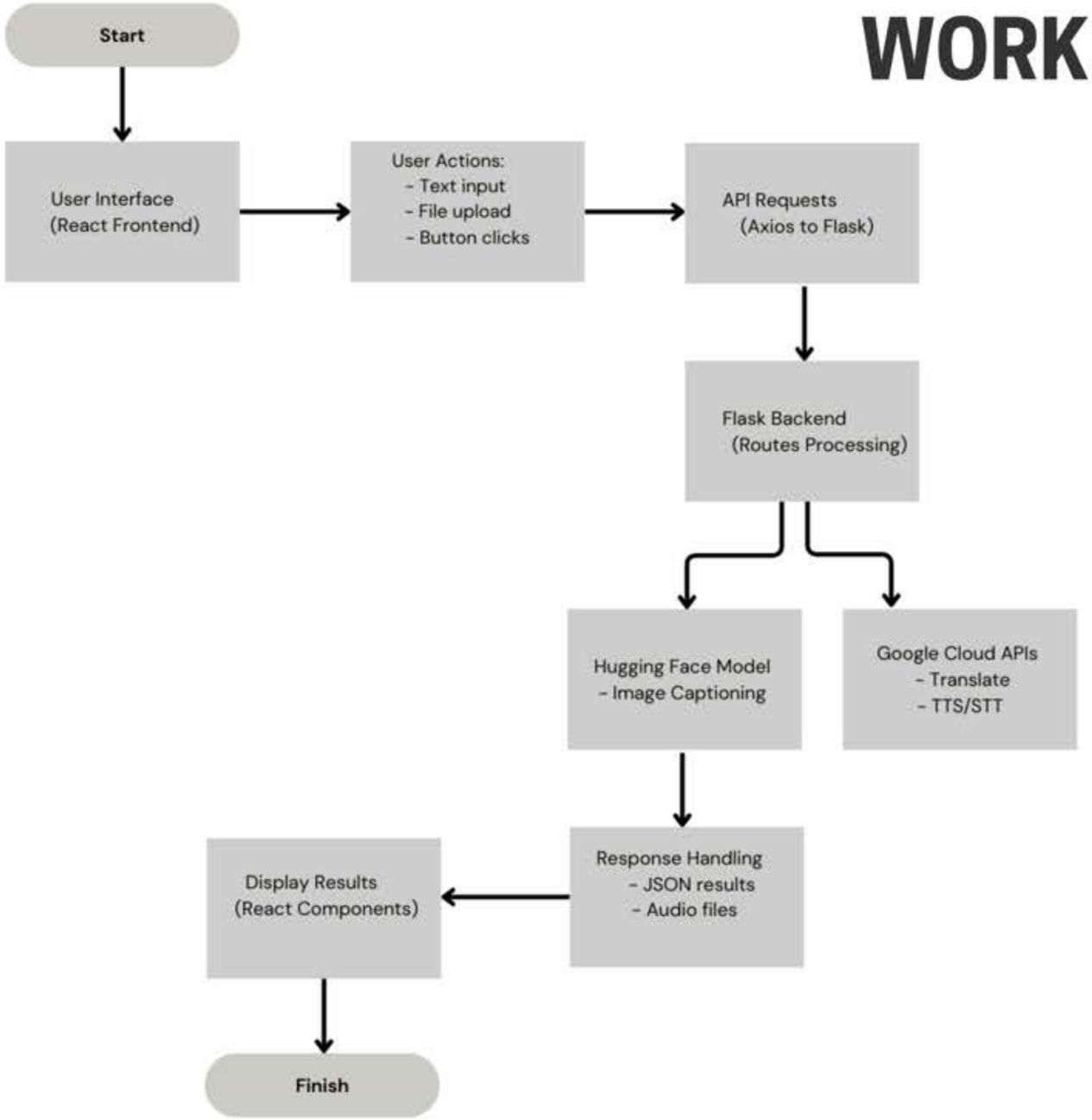
Text Analysis Module

utilizes Google Cloud Natural Language API to analyze sentiment, entities, and syntax in text data.

Response Synthesizer

generates context-aware responses through rule-based logic and templates. Optionally uses Google Text-to-Speech for voice output.

WORKFLOW OF MODEL



CORE **FEATURES**

Text Translation

- Detects input language
- Translates text in <500ms
- Supports 100+ languages

Speech Services

- STT: 92% accuracy, multilingual
- TTS: 220+ natural-sounding voices

Image Captioning

- Uses BLIP (ViT + GPT-2)
- Captions like “A group of people playing football”

TEXT SERVICE

Features :-

- detect input language
- translate to target language
- synthesize speech (TTS)

Workflow :-

TEXT INPUT
↓
DETECT LANG
↓
TRANSLATE
↓
GTTS
↓
MP3 PLAYBACK

SPEECH-TO-TEXT

Features :-

- multilingual speech transcription
- optional translation

Workflow :-

AUDIO UPLOAD
↓
ANY FORMAT
↓
STT
↓
TRANSLATE
↓
TEXT OUTPUT

IMAGE CAPTIONING

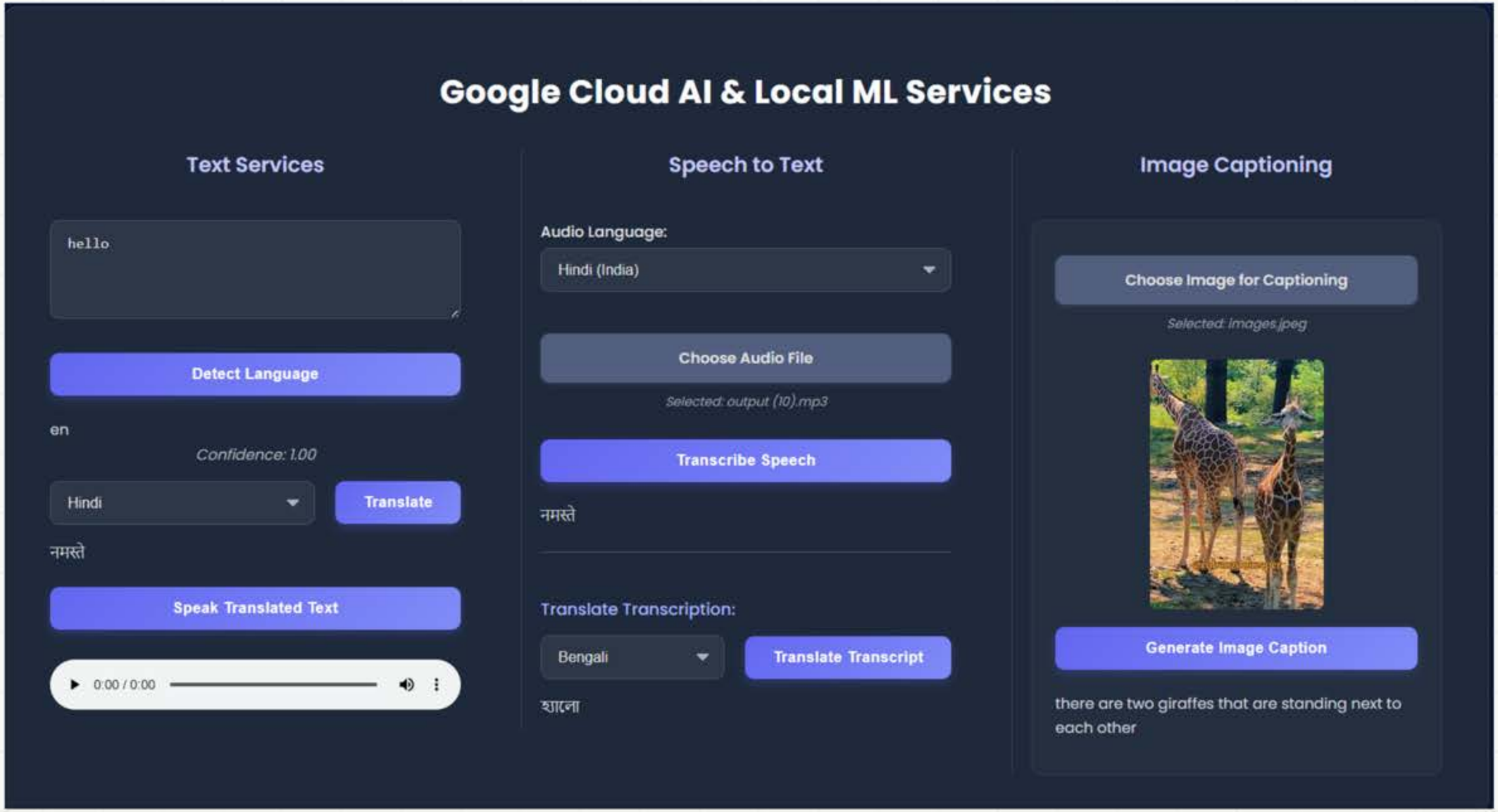
Features :-

- context-aware image descriptions

Workflow :-

IMAGE UPLOAD
↓
BLIP
↓
GPT2
↓
CAPTION OUTPUT

SAMPLE OUTPUT



TEXT SERVICE

SPEECH-TO-TEXT

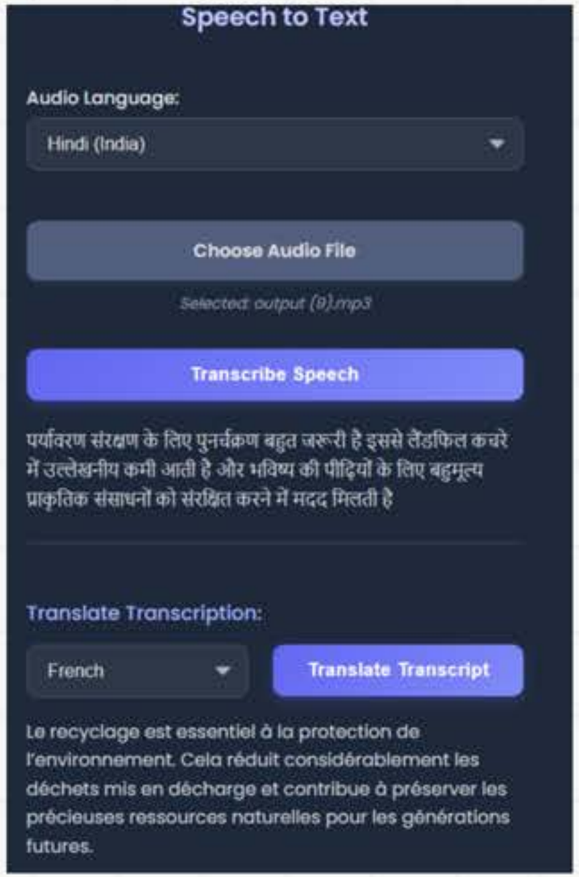
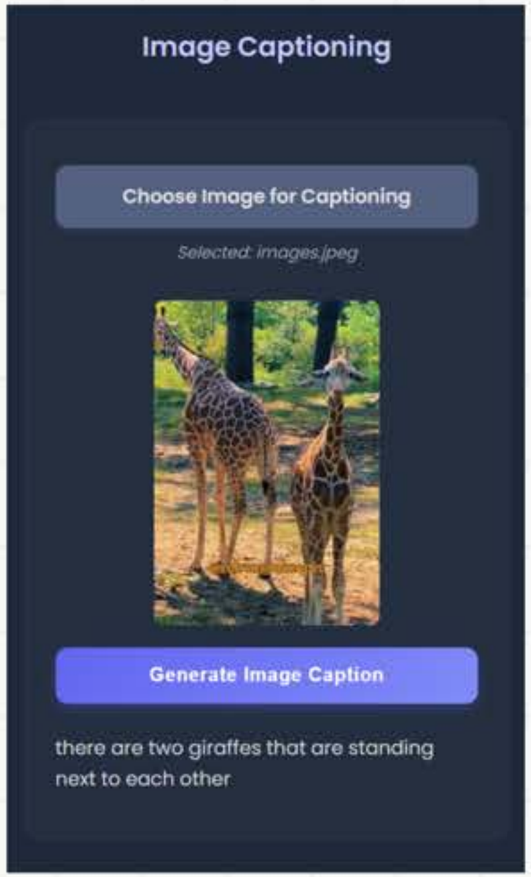


IMAGE CAPTIONING



PERFORMANCE HIGHLIGHT

🚀 Fast & Responsive

- Text translation and TTS: ~500ms (avg) for short sentences
- mage captioning: ~1–3 seconds. (local model with GPU fallback)
- Speech-to-text: Real-time transcription for <30s audio clips

🧠 Accurate AI Output

- BLIP model generates context-rich captions
- STT maintains ~85% accuracy in noisy environments
- Google Translate preserves idioms & context

🌐 Multilingual Capabilities

- Supports 100+ languages for text & speech.
- Regional language support (e.g., Hindi, Tamil, Bengali)

🔊 Real-Time Audio Playback

- Translated text-to-speech played instantly
- Auto-deletes temporary files to optimize storage

📱 Smooth UX/UI

- ReactJS single-page app ensures seamless interaction
- Instant feedback for every input type (text/audio/image)

💡 Cloud-Edge Hybrid

- Combines Google Cloud APIs + Local ML (for efficiency & control)
- Optimized for scalability and cost-effective deployment

CONCLUSION

The **Multimodal AI Assistant** successfully integrates Google Cloud services with local machine learning models to process and understand **text, speech**, and **image inputs** in real time. It provides a unified, multilingual platform capable of translating text, transcribing speech, and generating image captions—enhancing accessibility and user experience across domains.

The system's modular architecture (ReactJS frontend + Flask backend) ensures **scalability, flexibility**, and **cloud-readiness**, making it suitable for deployment in real-world applications such as **education, healthcare, accessibility tools,** and **content creation.**

This project validates the potential of combining cloud APIs with local AI models to deliver efficient, interactive, and intelligent user experiences—paving the way for the future of **human-AI interaction.**

THANK YOU