# TECHNOLOGY STACK

## Technology Stack (Architecture & Stack)

### Project Details

| | |
|---|---|
| Date | 19 February 2026 |
| Team ID | LTVIP2026TMIDS65707 |
| Project Name | TransLingua: AI-Powered Multi-Language Translator |
| Maximum Marks | 4 Marks |

### Technical Architecture

**[Insert your Technical Architecture Diagram Here]**

**Guidelines & Demarcations:**

- Processes (Application Logic): Users input text, select source/target languages via the Streamlit UI. The backend uses the Gemini A I model to process the request. The translated text is refined by the AI and sent back to the frontend.
- Infrastructural demarcation: Local development environment transitioning to Google Cloud Run (Serverless Container).
- External interfaces: Google Generative AI API (Gemini).
- Data Storage components: Google Cloud Firestore (for translation history) and Google Cloud Storage (for file handling).
- Interface to machine learning models: Interfacing with the pre-trained Gemini Pro model (gemini-1.5-flash) via the google.generativeai library.

### Table-1: Components & Technologies

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1 | User Interface | How user interacts with application | Streamlit |
| 2 | Application Logic-1 | Logic for a process in the application | Python |
| 3 | Application Logic-2 | Logic for interacting with the AI model | Google Generative AI (genai) |
| 4 | Database | Local testing and configuration | SQLite / .env files |
| 5 | Cloud Database | Database Service on Cloud | Google Cloud Firestore (NoSQL) |
| 6 | File Storage | File storage requirements | Google Cloud Storage (GCS) |

| | | | |
|---|---|---|---|
| 7 | External API-1 | AI-driven translation model processing | Google Generative AI API |
| 8 | Machine Learning Model | Accurate and contextually relevant translation | Gemini Pro (gemini-1.5-flash) |
| 9 | Infrastructure | Application Deployment on Cloud | Docker & Google Cloud Run |

### Table-2: Application Characteristics

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1 | Open-Source Frameworks | List the open-source frameworks used | Streamlit, Docker |
| 2 | Security Implementations | List all the security / access controls implemented | Environment variables (.env), Google Cloud IAM for API restriction |
| 3 | Scalable Architecture | Justify the scalability of architecture | Serverless containerization using Google Cloud Run |
| 4 | Availability | Justify the availability of application | Google Cloud Run multi-zone redundancy |
| 5 | Performance | Design consideration for the performance | Streamlit Caching (@st.cache_data) to minimize redundant API calls |