Expression of Interest (EoI)

Consultancy for Implementation of a chatbot framework and the implementation of new chatbot use case(s) for the public sector.

Reference number: 83493986

1 General information

1.1 Background

1.1.1 About GIZ Rwanda and the Digital Transformation Center

The Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH is a federally owned international cooperation enterprise for sustainable development with worldwide operations. The GIZ Office in Kigali covers GIZ's portfolio in Rwanda and Burundi. GIZ Rwanda/Burundi implements projects on behalf of the German Federal Ministry for Economic Cooperation and Development, the European Union and other commissioning authorities in the following priority areas: Sustainable Economic Development, Good Governance, Climate, Energy and Sustainable Urban Development, Digitalization and Digital Economy, Mineral Governance, Peace and Security in the Great Lakes Region.

To advance digital transformation in Rwanda and beyond, GIZ and the Rwandan Ministry of ICT and Innovation (MINICT) have formed the Digital Transformation and Digital Economy Cluster. The activities, themes, topics, and projects of the cluster are dedicated to delivering impact-driven digital solutions, developing the capacities of the local innovation ecosystem, and replicating and scaling up digital solutions at regional and continental level. The initiative supports both the public and private sector by transferring knowledge and skills and building organizational, structural, and technological capacities. It also provides a modern space to the Rwandan partners and innovators to boost creativity and collaboration: The Digital Transformation Center Rwanda (DigiCenter).

1.1.2 About the Digital Transformation Center's AI Hub

One of the focus areas of the Cluster Digital Transformation & Digital Economy is Artificial Intelligence. Through the AI Hub, based at the Digital Transformation Center, GIZ particularly contributes to sustainable and open approaches to AI in Rwanda and at the regional level. The vision of the AI Hub is to co-create a vibrant and inclusive ecosystem in harnessing the benefits of open and ethical AI. The mission is empowering our partners through providing open AI training data, capacity building, and development of ethical policy frameworks towards building community-driven AI solutions.

A specific objective is to make AI solutions available for the Rwandan population in Kinyarwanda. Therefore, the AI Hub has contributed to the creation of AI training data sets and has piloted AI-based solutions in the areas of natural language processing (NLP, e.g.: voice assistants) and machine translation, for example through a recent collaboration with the Rwandan Center for the Fourth Industrial Revolution (C4IR) and its AI Innovation Lab.

1.1.3 C4IR and the AI Innovation Lab

The AI Innovation Lab is part of the non-profit organisation Center for the Fourth Industrial Revolution (C4IR). C4IRs mission is to promote AI technologies in Rwanda. The AI Innovation Lab facilitates AI projects through different services, e.g., technical and business consultancy.

1.1.4 GovStack Initiative

In 2020, the GovStack Initiative was launched by Estonia, Germany, the International Tele-communication Union (ITU) and the Digital Impact Alliance (DIAL) together with partners such as Smart Africa, the World Bank, GSMA and SAP. The initiative aims to create a common framework for and foster technical practice with regards to developing reusable and interoperable digital government elements – so-called “digital building blocks” – which are needed to create full-fledged digital government services.

The global GovStack Community invests in these digital building blocks to help governments reduce costs, time, and resources required to create or modify digital services and applications. As these digital building blocks are interoperable and based on open-source software, they are easier to customise, design and implement, combine as needed, and scale across multiple sectors. Examples of digital building blocks include:

- Digital registries
- Identification and authentication
- Payments
- Registration
- Security

In addition, the GovStack Initiative supports partner governments in making their digital administration more efficient, faster, and secure and to reduce system duplication. By providing building blocks that increase security and transparency, as well as accountability of how government services are delivered, fraud is prevented, and corruption reduced. GovStack also aims to increase country ownership of e-government solutions and digital sovereignty, e.g. by making governments less dependent on individual software solutions

(vendor lock-in) thanks to the open-source nature of the building blocks. As a central pillar of all its activities, GovStack focuses on adherence to international data protection and IT security standards while working hand in hand with civil society and academia to ensure a rights-based approach to e-governance.

1.1.5 RISA and the Centre for Digital Public Infrastructure (DPI)

As Rwanda's leading public agency mandated to implement national ICT policy, RISA oversees infrastructure planning, innovation programs, and the integration of digital solutions to enhance service accessibility and socio-economic progress.

The recently launched Center for DPI is Rwanda's strategic hub for inclusive, interoperable, and scalable digital systems. Hosted by RISA, the Center promotes open frameworks, modular architectures, and gender-responsive solutions while serving as a testbed for emerging DPI technologies that support national and regional transformation.

1.2 Context for this project

1.2.1 Chatbots and digital public goods

A chatbot is a computer program or artificial intelligence application designed to simulate human conversation and interaction through text or voice-based interfaces. It is designed to engage in natural language conversations with users, responding to their queries, providing information, and performing tasks.

Chatbots can be utilized as digital public goods to improve access to services and information, particularly in underserved or remote areas, whilst adhering to privacy and other applicable laws, do no harm by design, and help attain the SDGs (as per the DPG standard of the Digital Public Goods Alliance). By providing automated responses to common queries and tasks, chatbots can help reduce the burden on human service providers and improve the efficiency of public services. For example, chatbots can be used to provide information on healthcare services, education, and government services, among others. This can help improve access to information and services for people who may not have access via channels such as government websites or government offices. Additionally, chatbots can provide round-the-clock assistance and support, ensuring that citizens can easily obtain information about services when needed.

1.2.2 GovStack Building Block Approach

The chatbot framework will be designed and implemented following the GovStack Building Block approach. The GovStack approach aims to provide a reference architecture for digital governance software to support sustainable development goals. Rooted in a "Whole-of-Government" approach, the GovStack Framework provides a methodology for leveraging common technology components and infrastructure to create and deploy

interoperable digital platforms which can address high-priority use cases across multiple sectors more easily.

What is a building block? Generally speaking, a building block:

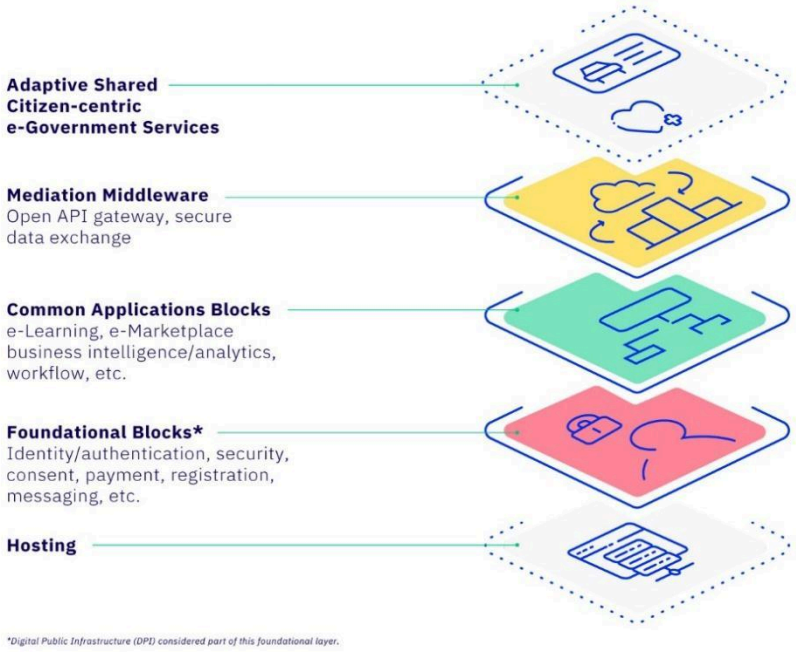
1. Refers to software code, platforms, and applications that are interoperable, provide a basic digital service at scale, and can be reused for multiple use cases and contexts.

2. Serves as a component of a larger system or stack.

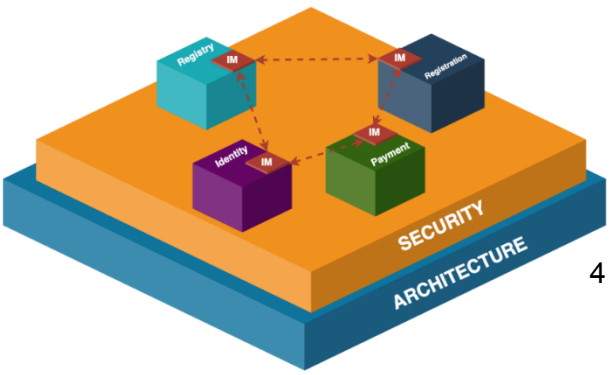
3. Can be used to facilitate the delivery of digital public services via functions, which may include registration, scheduling, ID authentication, payment, data administration, and messaging.

4. Building blocks can be combined and adapted to be included as part of a stack of technologies to form a country's Digital Public Infrastructure (DPI).

5. Building blocks may be open source or proprietary and therefore are not always DPGs.



Building blocks are composable, inter-operable software modules that can be used across a variety of use cases. They are standards-based, open source and designed for scale. Each Building Block represents, as much as possible, the



minimum required functionality (MVP) to do its job. This ensures each Building Block is usable and useful on its own, and easily extensible to support a variety of use cases.

A Building Block is composed of domain-driven microservices, modelled as closely as possible on existing roles and processes. This helps ensure each building block is as useful as possible in the real world. Building Blocks exchange data using lightweight, human-readable data that can easily be extended where needed. Data models and APIs are described in a lightweight manner that's human-readable, allowing them to be easily and quickly understood and validated.

Characteristics of Building Blocks:

1. **Autonomous**: building blocks provide a standalone, reusable service or set of services; they may be composed of many modules/microservices.

2. **Generic**: building blocks are flexible across use cases and sectors.

3. **Interoperable**: building blocks must be able to combine, connect, and interact with other building blocks.

4. **Iterative evolvability**: building blocks can be improved even while being used as part of solutions.

Per the Digital Public Goods Alliance (DPGA) definition, to be considered a Building Block, solutions must meet the following technical requirements determined by the GovStack Initiative which includes:

1. Open API, Open API Specifications, Rest API

2. Packaged in a container

3. Include an Information Mediator where communication flows between services that are not co-located

1.2.3 Previous work

1.2.3.1 Mbaza Chatbot

The Mbaza AI Chatbot aimed to provide reliable and up-to-date information on COVID-19 to the Rwandan population in the local language Kinyarwanda. The chatbot was developed

through several iterations: while the first iteration focused on a simple menu-based USSD chatbot, the product has evolved into a text-based conversational chatbot, and finally into a conversational voice-based chatbot.

All parts of the chatbot were made available to the Rwandan AI community as an open-source package. The repositories include speech-to-text (STT), text-to-speech (TTS) models for Kinyarwanda and infrastructure code to deploy the designed architecture into different environments, providing the community with an open resource for chatbot development and supporting the creation of future use cases. Through meetups and a community of practice, the AI Hub continues to raise awareness and build capacities around the resources. Currently, the repositories are still hosted on infrastructure of the Rwanda Information Society Authority (RISA), while the STT data resides at Mozilla Common Voice Portal, and models on Hugging Face, and GitHub.

The repo with reusable components; language models and RASA NLU, as well as pipelines to deploy various channels such as USSD, web widget, and interactive voice response (using Asterisk) for voice support, shall be made available as building blocks for the Chatbot Federation.

1.2.3.2 Farmer Decision Chatbot

The FAIR Forward – Artificial Intelligence for All project together with the Digital4Rwanda project is implementing a project activity aiming to leverage advancements in AI and language modeling to provide Rwandan farmers with easily accessible, real-time agricultural advisories through voice. Information may include but is not limited to general advisory information, weather advisory, pest and disease prevention tips, and information on seasonal subsidy programs sponsored by public and non-profit agencies. This initiative aligns with Rwanda's commitment to enhance agricultural productivity through digital transformation. The interactive voice response (IVR) system will deliver information in Kinyarwanda directly to farmers and will be accessible on all types of phones. The advisories will be based on reliable data sourced from credible public and private entities, ensuring that the content is both accurate and relevant. The Farmer Decision Chatbot was implemented in a collaboration of GIZ as funder, the Centre for the Fourth Industrial Revolution (C4IR) as grant partner, the Ministry of Agriculture and Animal Resources (MINAGRI) as the use case owner and KINLP as implementation partner.

The system is a voice-only interface that operates exclusively in Kinyarwanda, designed for accessibility via telephone or a softphone. It is implemented mainly in the JAVA programming language. It leverages a Retrieval-Augmented Generation (RAG) architecture, based on Large Language Models (LLMs) provided as a service. The system includes specialized trained models for Automatic Speech Recognition (ASR), Text-to-Speech (TTS), and semantic Embeddings, all tailored specifically for the Kinyarwanda language. The complete infrastructure is hosted at the Ministry of Agriculture and Animal Resources (MINAGRI), ensuring localized deployment and data sovereignty.

1.2.3.3 Data Protection Chatbot

To support the implementation of Rwanda's Data Protection Law, the AI Hub project together with Data Protection Office (DPO) developed a chatbot as digital tool to assist both the general public and organizations in navigating their rights and obligations under the data protection and privacy law No. 058/2021. The chatbot was designed to automate routine compliance interactions, streamline the complaint submission process, and disseminate key educational resources on data protection.

The initial version of the chatbot was built using the RASA framework, offering a rule-based and intent-driven conversational experience. This version supported interactions in English only, with pre-configured flows tailored to the three primary stakeholder groups: Data Subjects, Controllers, and Processors. It addressed frequently asked questions, provided downloadable guides, and offered structured support for filing data protection complaints. It was implemented in Python.

To improve the system's ability to handle a wider range of user inputs—especially queries outside its predefined intents—the chatbot architecture was later extended to incorporate Retrieval-Augmented Generation (RAG) using a LLaMA model which is available through the Groq API. The system first tries to answer the question using RASA. If RASA cannot answer the question, it will answer via the RAG system. This enhancement allowed the system to dynamically pull information from a curated knowledge base and generate more accurate, contextually relevant responses to open-ended or out-of-vocabulary queries. Performance testing showed marked improvements in response accuracy and user satisfaction, particularly for less frequent or complex questions.

Key features of the chatbot included:

1. **Natural Language Understanding** to recognize user roles and topics related to data access, rectification, and complaint procedures.
2. **Role-Based Personalization**, ensuring that guidance was customized depending on whether the user was a Data Subject, Controller, or Processor.
3. **Complaint Management**, enabling users to submit structured complaints through a step-by-step interface, with automated email routing and tracking.
4. **Educational Resource Distribution**, such as downloadable legal documents, templates, and links to the Data Protection Law.
5. **A Fallback Mechanism** (human-in-the-loop) to redirect users to human support or suggest related topics when a query could not be fully resolved.

The chatbot also integrated analytics capabilities, allowing monitoring of user activity across user types (data controller, data processor, and data subject) to inform future updates and stakeholder engagement strategies for awareness raising strategies.

1.2.3.4 Kenya Chatbot for Discoverability of Government eServices

In Kenya, significant progress has been made towards digitizing government services, with over 5.000 services having been digitized as part of the government's eCitizen portal as of May 2023. Despite the digitization efforts by the Government of Kenya, citizens still face difficulties in discovering and navigating through the vast array of online services. The lack of a centralized and user-friendly platform leads to inefficiencies, frustration, and potential underutilization of valuable services.

To simplify the discoverability of government eService, the ICT Authority of Kenya (ICTA) and Konza Technopolis Development Authority (KoTDA) are collaborating with DTC Kenya and the Multi-Donor Action "Initiative for Digital Government and Cybersecurity" (IDGC) to develop an AI-enhanced chatbot that will make it easier for citizens to find government services for their demands. The chatbot development is further supported by the ITU as a reference implementation of its "Open Source Ecosystem Enabler (OSEE)" project which supports the Kenyan government with setting up an Open Source Project Office (OSPO).

The chatbot will be developed as a DPG-ready platform following the GovStack approach (as described in chapter 1.2.2). It will allow other Ministries, Departments and Agencies in Kenya to consolidate information about digitized services they offer and guide users on how to access relevant services. The design phase of the chatbot was concluded in early 2025 and the chatbot implementation is currently underway with a launch foreseen for Q4 2025.

Several technical components of the Kenyan chatbot, as well as lessons-learnt from its design and implementation phase, may be relevant for informing the development of the chatbot framework as part of this assignment (see chapter 1.3).

1.3 Objectives

This section lists the objectives of the generic chatbot framework (1.3.1), the implementation of several text / voice bots (1.3.2), and the cooperation with other chatbot projects (1.3.3).

1.3.1 Create a generic chatbot framework

Our goal is to develop a versatile, technical framework, that shall hereby be referred to as Chatbot Federation or ChatFed, designed to enable the rapid and cost-effective development of chatbots. The framework aims to address the following needs:

1. **Technology Reusability:** Currently, each GIZ chatbot is built using a unique technology stack, resulting in duplicated effort and high development costs. ChatFed will standardize development by allowing component reuse across different chatbot projects, significantly reducing both time and costs.

2. **Modular Upgradability:** As AI technologies evolve rapidly, maintaining and upgrading individual chatbot components becomes challenging. ChatFed will ensure compatibility across chatbots, so an upgrade to one component (e.g., natural language processing) can be applied across all chatbots built on the framework. Upgrading chatbots to the latest development in AI will be more cost-efficient in this way.

3. **Simplified Hosting:** With the increasing number of government-hosted chatbots in Rwanda, a unified architecture will simplify deployment and maintenance. ChatFed will provide a single comprehensive hosting manual, along with short chatbot-specific guides, streamlining the onboarding process for system administrators.

4. **Shared Components:** Without a unified framework, each chatbot requires its own modules (e.g., Text-to-Speech). ChatFed enables shared use of such modules, reducing duplication and lowering hosting costs, especially beneficial for organizations deploying multiple chatbots.

5. **Standardization:** Adopting a unified chatbot framework is a crucial step toward creating a standardized GIZ chatbot product, enhancing consistency, maintainability, and scalability across projects.

1.3.2 Implementation of several text / voice bots

The primary objective of this project is to design and implement several chatbots capable of engaging users through either text or voice interfaces. Through a structured approach encompassing comprehensive requirement analysis, user-centered conversational design, and robust AI model training, the solution aims to deliver high levels of usability, accessibility, and adaptability across various domains or industries. We can either extend existing chatbots or create new chatbots. We will decide on the exact chatbots in the beginning of the project.

1.3.3 Cooperation with other Chatbot Projects

We aim for the widespread adoption of ChatFed as a foundational chatbot platform. To achieve this, the project must be strategically and technically aligned with other chatbot initiatives to: (1) promote standardization and interoperability, (2) increase the visibility and adoption of ChatFed, and (3) support other teams in leveraging ChatFed as a technological base for their own projects, while also encouraging contributions back to the core platform.

To this end, the contractor will actively collaborate with existing GIZ chatbot initiatives, for example those led by the global FAIR Forward Initiative and GIZ's Data Service Centre. Additionally, we remain open to engaging with further partners and initiatives that may arise over the course of the project.

1.3.4 Cooperation with the AI Innovation Lab

The AI Innovation Lab (C4IR) will consult the technical implementation and ensure that the technical implementation follows the specifications. Therefore, contractor will include the AI Innovation Lab in all necessary communication.

The Lab will play an advisory and potentially facilitative role in the project to make sure that the project meets the needs of the Rwandan stakeholders and feeds into existing initiatives and priorities – for example around Digital Public Infrastructure (DPI) – such as the GovStack Initiative and the Centre for DPI by the Rwanda Information Society Authority (RISA).

1.3.5 Cooperation with RISA and the Centre for DPI

The service provider is required to provide an initial infrastructure sizing assessment – forecasting compute, storage, and AI model development needs (in consideration of existing open-source resources for Kinyarwanda). In addition, a digital readiness & use-case suitability framework built on established ICT4D and DPI guidelines e.g., ICT Sector Strategic Plan (2024 – 2029), the DPI Centre's interoperability criteria – to demonstrate methods for ensuring that use cases meet technical and operational thresholds. Detailed consultation with RISA will occur post-award to validate and refine assessments.

The chatbot framework development must be executed in close collaboration with RISA and the DPI Center to ensure compatibility with existing and future citizen-facing platforms.

2 The ChatFed Framework

ChatFed (The Chatbot Federation) is a framework that will allow for the quick and cost-efficient development and operation of chatbots. Through reusability, new chatbot projects will not need to reinvent the wheel. It will be an open framework composed of numerous small, interoperable building blocks. When creating a new chatbot using the framework, developers will be able to select from existing building blocks within ChatFed. Additionally, if new building blocks are created, contributors are encouraged to add them back to ChatFed, fostering system growth.

Although ChatFed will be initially build alongside Rwandan use cases, it will have a focus beyond East Africa. We aim at gradually extending the focus from Rwanda over east Africa to other GIZ partner countries use cases. Please note that it is neither planned nor intended to place the AI Solution on the European Union market or put the AI Solution into service in the European Union. The service provider should not use any outputs produced by the AI system in the European Union in any manner (for more information see chapter 10).

ChatFed will be multilingual. Initially, ChatFed will offer a limited set of functions. However, its open, modular, growth-oriented architecture simplifies the addition of new building blocks.

We particularly encourage every GIZ project to make the growth of ChatFed part of their TOR. ChatFed will grow with each chatbot that uses the framework and follows the same open-source mindset.

The ChatFed technology as implemented in this tender is generally open source under a permissive license and free of charge. We strive for widespread adoption of ChatFed, making it available outside the GIZ ecosystem under an open, permissive license that also allows the creation of proprietary closed-source products with ChatFed.

ChatFed will support a diverse range of chatbots, from task-oriented dialog systems of the 2010s (e.g., RASA or MultiWOZ) to modern RAG Systems, from ELIZA to LLM-based personal assistants (e.g., ChatGPT) and Agentic AIs. Moreover, it will support various user interfaces, including voice-only, text-only, multimodal (e.g., chat + hypertext), feature phones, and more. This wide variety of applications will be enabled by the open architecture that provides loosely coupled, common building blocks. However, we will not implement all these functionalities in this tender.

2.1 Open Innovation – Partnership Model

The ChatFed framework (not the individual chatbot use cases) will be developed as an open-source project, guided by an open innovation approach. Our goal is to foster a global community of chatbot initiatives built on the ChatFed framework. In the early stages, the community will be supported through resources provided by this tender and DTC Rwanda. Throughout the project, we will work in close collaboration with GIZ FMB and the Data Service Center to ensure strong alignment and shared value. During the project, we plan to transition single modules of the ChatFed Framework as knowledge outputs to the Data Service Center. In the long term, our vision is to transfer stewardship of ChatFed to a recognized open-source foundation, such as the Apache Software Foundation.

2.2 Overview of the Framework

The architecture of ChatFed will be built upon three main pillars:

1. **Modular Docker Containers**: ChatFed consists of a collection of Docker containers that can be assembled and configured according to the chatbots needs. These containers communicate with each other via APIs, primarily using HTTP APIs.

2. **API Standards**: ChatFed defines a set of API standards that facilitate communication between Docker containers. This ensures that the various building blocks of ChatFed are compatible and interoperable.

3. **Comprehensive Documentation**: ChatFed is thoroughly documented, making it reusable and accessible even without direct assistance from its developers.

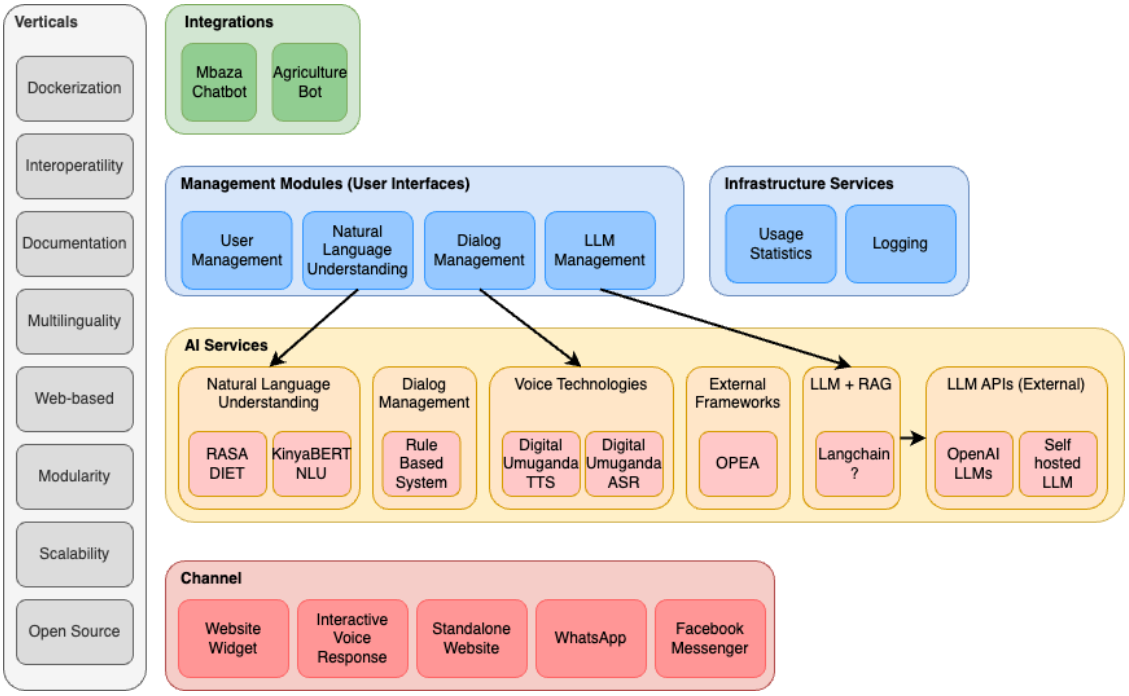


Figure 1: Overview of the ChatFed Framework

Figure 1 shows an overview of the Framework. The verticals outline its design principles.

1. **Dockerization**: ChatFed will be based on Docker containers, with each building block of ChatFed being a Docker container. This open-source industry standard facilitates easy deployment and scalability.

2. **Interoperability**: At the core of ChatFed are a set of API definitions that facilitate communication between different components, ensuring interoperability.

3. **Documentation**: ChatFed will emphasize comprehensive documentation, ensuring that anyone with technical skills can use it effectively without needing assistance from the core developers.

4. **Multilinguality**: Unlike most technologies that are built primarily for English or a small set of Western languages, ChatFed is designed as a multilingual framework with a focus on low-resource languages. ChatFed will be designed to be multilingual from the start. Even if a component is implemented only monolingual, it will be designed to allow the extension to additional languages in mind. We will start with use cases in Kinyarwanda, Swahili and English.

5. **Web-Based**: ChatFed is a web-based system, meaning the chatbot itself will be installed on a web server. Client applications on devices (web browsers, smartphone apps, etc.) will connect to these web servers. This architecture does not support local installations on devices like phones or local computers.

6. **Scalability**: ChatFed will achieve scalability through its Docker-based architecture.

7. **Open Source**: ChatFed will be open- source at its core, free for all types of use, including commercial use cases.

The remainder of this section explains the other components of Figure 1.

- **Integrations** A chatbot utilizing ChatFed is an integration composed of various ChatFed components. An integration could include, for example, ChatFed's management modules, certain infrastructure tools, and one of the communication channels. Other components of the integration can be implemented independently of ChatFed.
- **Management Modules** Chatbot frameworks should be manageable not only by computer programmers but also by individuals without deep technical skills. To facilitate this, we need management modules that provide user-friendly interfaces. These modules allow non-technical administrators to perform tasks such as adding new users to the system, incorporating new documents into a RAG system, analyzing chat log files, and more.
- ChatFed will feature a unified management system with modular building blocks that can be enabled or disabled as needed. For instance, if the chatbot does not require user management, the corresponding building block can be turned off.
- **Infrastructure Services** Every chatbot framework requires infrastructure tools, such as technical logfiles, logfiles for the conversations or usage statistics. These will be subsumed under infrastructure services.
- **AI Services** ChatFed supports a wide variety of chatbot types, necessitating the inclusion of diverse AI services. These AI services will often be connected to management interfaces to facilitate their administration. We aim to incorporate other open-source tools into ChatFed, such as RASA, LangChain, and OPEA.
- Although we initially aim at modern, LLM based architectures, we will make sure that ChatFed will also support other types of chatbots. LLMs will be optional. The LLM implementations of ChatFed will be independent from the actual LLM. LLMs can be either hosted as part of ChatFed or outside of ChatFed as APIs.
- **Channels** Users interact with the chatbot through different channels, which can include feature phones, WhatsApp, website widgets, and more.
- The services listed in Figure 1 serve as an example. Not all these services will be implemented during the project. We will only implement services that we require in the use cases.

2.3 Growth oriented architecture

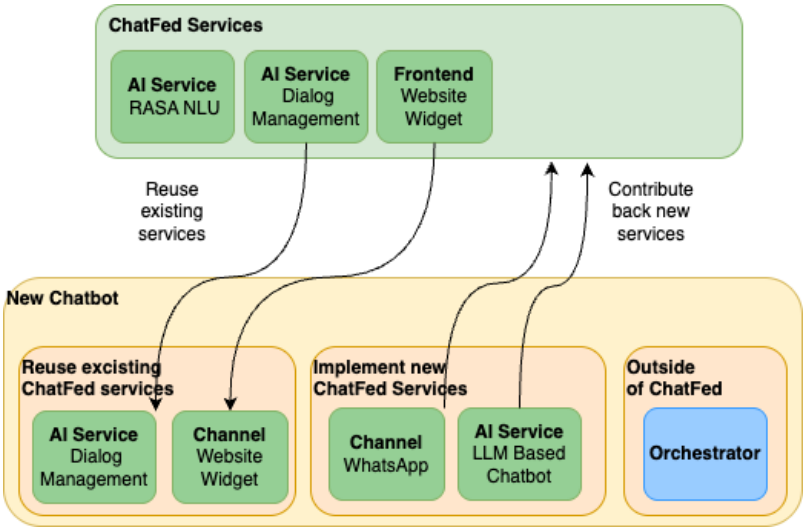


Figure 2: Creating a new chatbot with ChatFed

Figure 2 illustrates the process of creating a new chatbot using ChatFed. The green box at the top of the figure highlights a number of existing ChatFed services that are available for reuse. A new chatbot project can choose to integrate any of these services. In the example shown, the new chatbot reuses two existing ChatFed services.

At the same time, we strongly encourage developers of new chatbot projects to contribute any newly developed services back to the ChatFed ecosystem. The example in Figure 2 implements two new services and contributes them back. Contributing a service back to ChatFed typically includes:

1. Implementing the service as a generic, reusable building block.

2. Providing clear and comprehensive documentation.

3. (Optional but encouraged) Publishing all related materials, such as source code, documentation, and Docker containers, under the official ChatFed repositories (e.g., GitHub organization, documentation site).

4. (Also encouraged) Releasing contributions under a permissive open-source license.

This architecture is designed for scalable growth—it allows projects to start small and expand as needed. There is no requirement to implement all services shown in Figure 1 from the outset. Instead, each chatbot project should aim to reuse existing services wherever possible and contribute new ones as necessary.

2.4 Architecture

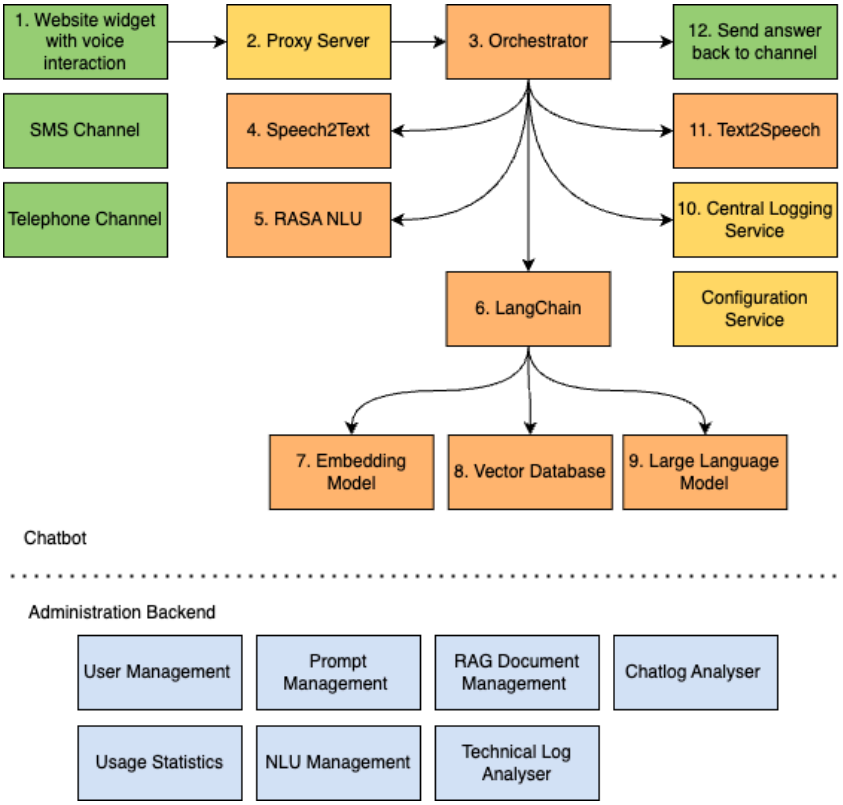


Figure 3: Architecture of a Hybrid RAG / rule-based Chatbot. The system shows the interaction channels (green), chatbot services(orange), administration components (blue) and infrastructure services (yellow). The horizontal line separates the chatbot (above) from the administration backend (below). The chart shows the processing pipeline of the answer generation and the numbers in the boxes show the execution order in the pipeline.

Figure 3 shows an exemplary hybrid RAG chatbot that is implemented using ChatFed to illustrate the software architecture of chatbots implemented with ChatFed. The user interacts with the chatbot through a website widget with voice input and output. The exemplary system offers other interaction channels based on SMS and telephone, but this example explains the website widgets workflow only. The AI functionalities of the chatbot are a hybrid, combined system of RAG and a rule-based dialog system (RBDS). The remainder of this section explains the processing pipeline from a user message to the chatbot answer.

1. **User Interaction:** The process begins when a user sends a message through one of the available channels, which can be text- or voice-based. In this example, the message is received via the telephone (voice) channel. While Figure 3 displays two additional channels, the interaction here occurs solely through voice.

2. **Proxy Server:** The proxy server acts as router and maps the outside facing APIs of internal components to a single, unified API accessible from the internet. In case the chatbot holds web user interfaces, these are mapped also by the proxy server to the

outside of the chatbot. The proxy server also functions as a firewall and ensures that internal services are not accessible from the internet.

3. **Orchestrator:** The orchestrator is the central component that manages incoming messages and returns responses to the appropriate channel. It implements the processing pipeline and coordinates the execution of various chatbot services.

4. **Speech2Text:** The orchestrator first sends the voice message to the Speech2Text service, which converts the audio input into text.

5. **RASA:** The RBDS is implemented using RASA. RASA allows rule-based, precise answer generation and dialog control for a narrow domain. Also, the RASA NLU allows for information extraction. In case the user's request cannot be answered by RASA, the system will hand over the RAG system.

6. **LangChain:** The textual message is then forwarded to LangChain, a standard library for building LLM-based applications, including RAG systems. LangChain offers a unified interface to a wide range of LLM and embedding models (e.g., Hugging Face, OpenAI, self-hosted models, and more). ChatFed provides a generic LangChain service that chatbot projects can reuse. This service can be directly integrated into the orchestrator and does not require a separate Docker container.

7. **RAG Workflow:** LangChain executes a standard RAG pipeline by invoking the **embedding model**, retrieving relevant information from a **vector database**, and generating a response using the **LLM**.

8. **Central Logging Service:** All components log to the central logging service. The orchestrator also stores the chat conversation in this service.

9. **Text to Speech:** The generated response is passed to the **Text to Speech** service which converts the text back into audio format.

10. **Answer Generation:** The final audio response is then sent to the user via the original voice channel.

11. **Administration Backend:** The bottom of Figure 3 shows the administration modules of the chatbot that allows also non-technical experts to manage parts of the chatbots functionality.

12. **Configuration Service:** The system deploys a centralized configuration service that delivers the systems configuration to all

2.5 Requirements of ChatFed

2.5.1 Requirements of a chatbot for the implementers of the framework

- ChatFed uses a multi-tier configuration.
    1. Each service implements their own default configurations.
    2. A central configuration file for the whole chatbot system overrides the local configurations of each chatbot.
    3. A selection of configuration variables can be managed from the administration backend.

2.5.2 Requirements of services implemented with ChatFed

1. Services are generally implemented in a generic way and all data, business logic and such is exemplary. The documentation website hosts this generic chatbot. Instances of these generic services are implemented in the respective chatbots. So, each service will be implemented at least twice during this project: Once as a generic example service and once as an instance of this generic service in a chatbot.
2. Each service contains unit and integration tests.
3. Each service is deployed as a docker container. Complex services with multiple components (e.g. a service + its database) is deployed via docker compose.
4. All services write their technical logs to the central logging service. Along that, they can also maintain local logfiles if necessary.
5. Each service provides an interactive API documentation build with Swagger.
6. Each service is documented in a single README file. In the project, we will define a common format with at least these sections:
    6.1. Introductory text – What is this service about
    6.2. Links to further materials, e.g., on the documentation website, the interactive API documentation or on other websites.
    6.3. Getting started
        6.3.1. Hardware requirements
        6.3.2. Installation
        6.3.3. Configuration
        6.3.4. Usage Instructions
    6.4. Technical information
        6.4.1. Architecture diagram (e.g. flow of user input, services used, orchestration)
        6.4.2. Component descriptions (e.g. Orchestrator, NLP services, databases)
        6.4.3. Data flow and processing pipeline explanation
        6.4.4. Tech stack and dependencies
7. Each service should be implemented in Python or Java (if possible)
8. Restart the service via HTTP API and reload configuration, models, etc.
    8.1. Each service has a restart HTTP endpoint that restarts the service and reloads the configuration that is configured in a secure way.

8.2. This allows to reload configuration variables via the administrator backend. Also, it allows to restart a service after a model has been trained.

2.5.3 Documentation requirements of the chatbots

In the project, we will define documentation guidelines. They should encompass at least the following:

1 Functional Overview: Explanation of the capabilities of the chatbot

1 Technical Overview

1.1. Architecture diagram (e.g. flow of user input, services used, orchestration)

1.2. Component descriptions (e.g. Orchestrator, NLP services, databases)

1.3. Data flow and processing pipeline explanation

1.4. Tech stack and dependencies

1.5. Installation guide

1.6. Hardware requirements

2. Configuration

2.1. Overview of configuration options

2.2. Environment setup (e.g. API keys, config files, env variables)

3. Common issues and fixes

4. Evaluation methodology and results

1.1.1 Requirements for AI models and datasets

1. ChatFed uses the HuggingFace tech stack as a basis for machine learning models. An exception is RASA.

2. All machine learning models and datasets generated in the project should be uploaded to HuggingFace if the model can be useful for other use cases.

1.1.2 Information Security Requirements

(Please note that requirements listed here are provisional and might adapt, depending on the requirements of a specific use case, particularly for data protection).

1. Authentication & Authorization
    1. All user and admin access must require secure authentication (OAuth 2.0, SSO, MFA where applicable).
    2. Implement role-based access control (RBAC) to restrict permissions based on user roles (e.g., user, admin, moderator).
    3. Tokens (e.g., JWTs) must be signed, securely stored, and have expiration policies.
    4. Session management must include secure timeout and reauthentication mechanisms.
2. Data Protection
    1. All sensitive data (user messages, personal info) must be encrypted in the database (data-at-rest encryption) and in transmission (data-in-transit encryption, e.g. through TLS/SSL)
    2. Personal identifiable information, e.g. IP Addresses, must be masked or anonymized where not strictly required.
    3. Ensure the system complies with, where applicable, with Rwandan Data Protection and Privacy Law and/or potentially other applicable local data privacy regulations and/or European GDPR (General Data Protection Regulation).
    4. Provide users with clear information on data collection, usage, and retention.
    5. Implement a mechanism for data deletion requests (Right to be Forgotten).
3. Passwords
    1. All databases and management user interfaces (such as database frontends) must be password protected.
    2. Default passwords must not be used.
    3. Passwords must not be stored in the source code or in the GitHub repository.

4. Input Validation & Output Encoding

    1. All user inputs must be validated and sanitized to prevent Cross-site scripting, SQL/NoSQL injection

    2. Ensure proper output encoding when displaying user-provided content.

5. Logging & Monitoring

    1. Log all authentication events, administrative actions, and suspicious activities.

    2. Ensure logs do not expose sensitive information (e.g., no logging passwords or secrets).

6. API Security

    1. All APIs must be protected using authentication tokens (e.g., OAuth 2.0, API keys with scopes)

    2. Validate all API inputs server-side.

    3. Implement CORS (Cross-Origin Resource Sharing) policies tightly.

7. Third-party Integrations

    1. Perform security assessments of all third-party libraries, APIs, and plugins.

    2. Prefer libraries with active maintenance and strong security track records.

8. Infrastructure Security: The system must deploy a firewall

1.1.3 Licensing

All technologies of ChatFed, including the framework, all its services and all datasets and AI models, must be published by the Contractor under a permissive open-source license. GIZ will decide on the exact licenses. Therefore, the Contractor shall grant GIZ at least those rights of use required to publish products and deliverables of this tender in accordance with section 1.1.3. All deliverables developed within this assignment shall be publication-ready under an open- source licence. The Contractor shall not use pre-existing components such as libraries that would not allow the dissemination of the deliverables under such open-source licences without prior express consent from GIZ. In so far as licences for pre-existing components would not allow the dissemination of deliverables themselves or as a part of the Chat Fed technologies, the Contractor may request permission together with a justification of such use; however, GIZ shall be entitled to decide in its sole discretion.

The contractor must research the licenses of all dependencies and ensure that they comply with our permissive open-source licensing standards and allow professional use and the integration into commercial, closed source products. In case the licensing of components deviates from these standards, their usage must be approved by GIZ prior to such integration.

2 Tasks to be performed by the contractor

This section lists the work packages (WP) of the project.

2.1 WP 1: Interoperability, Documentation and Cooperation

WP 1 drives the topics Interoperability, Documentation and Cooperation.

2.1.1 WP 1.1 Requirement Analysis

Key Tasks

- Assess detailed requirements of the ChatFed framework with use case partners.
- Create a list of services that will be made available through ChatFed based on requirements and in discussion with GIZ.
- Create a list of interoperability standards that will be created in the project. Generally, each service should create an API standard.
- Further develop and define requirement standards for ChatFed, its services and AI models based on Sections 2.5.1, 2.5.3, 2.5.4 to guide future developments of the framework.

Deliverables

- Requirement analysis report
- Define standard requirements for ChatFed framework, services and the models.

2.1.2 WP 1.2 Technology Review

Key Tasks

- Research, list and assess framework technologies for chatbots, such as LangChain, OPEA, RAG Web UI and more.
- Identify technologies, e.g., individual services, interoperability standards and more, that can be reused within this project. Generally, we aim at reusing existing standards as much as possible.

Deliverables

- Technology Review Report

2.1.3 WP 1.3 Interoperability

Key Tasks

- Coordinate standardization between use cases and existing services from Section 3.1.2.
- Define and document standards
- Ensure that the use cases follow the standards

Deliverables

- Documented interoperability standards on documentation website
- The components developed during the use cases follow these standards

2.1.4 WP 1.4 Documentation and Project Website

Key tasks

- Define documentation by further developing the standards defined in Sections 2.5.2 and 2.5.3.
- Create documentation website using GitHub pages and Jekyll
- Create a project website using GitHub pages and Jekyll
- Collect all documentations generated through the services and publish public-facing documentation as part of the documentation and/or project website.
- Create tutorials, such as “How to create a RAG application with Voice Input and Output using ChatFed”
- Create a GitHub Organization to and document host newly developed source codes. Each of these repositories will be well documented itself.

Deliverables

- A website containing technical documentation such as interoperability standards, system overviews and more
- Documentation standards documented on the website
- Create ten tutorials

2.1.5 WP 1.5 Cooperation with other chatbot projects

Key tasks

- Have meetings and calls with partners from outside of this project, also possibly in the East Africa region, and collaborate with them including, but not limited to, projects mentioned in chapter 1.2.3.

- Engage with these partners in discussions around architecture, interoperability and other topics regarding the design of the framework. This includes knowledge sharing, capacity building activities, and presentations of lessons learned from the chatbot framework development and pilot implementation.
- Scope opportunities for incorporating components of existing open source chatbot projects into the ChatFed framework.
- Give technical support to partners so they can create chatbots using ChatFed and contribute their services back to ChatFed.
- Enable these users to add contents to the project, the GitHub organization and the documentation websites.

2.2 WP 2: Creation of ChatFed Services

WP 2 creates services for ChatFed that can be reused in the implementation of Chatbots during WP 3, 4 and 5.

2.2.1 WP 2.1 Administration Backend

This section describes the administration backend of ChatFed, that the contractor must implement. First, the section lists Non-Functional (2.2.1.1) and Functional (2.2.1.2) requirements. Then, Section 2.2.1.3 describes the different user interfaces / views of the backend. The administration backend exposes several APIs (Section 2.2.1.4). Then, Section 2.2.1.5 explains the batch upload functionality. The remaining two sections explain a model deployment mechanism (2.2.1.6) and the Deliverables (2.2.1.7).

2.2.1.1 Non-Functional requirements

1. **Usability** The interface must be intuitive and easy to navigate for users with minimal training. The UI must follow established design standards and conventions. It provides helpful error messages and tooltips.
2. **Performance** UI response time should be under 2 seconds for 95% of user interactions. Pages or screens must load within a maximum of 2 seconds. Real-time updates should occur within 500ms of the event trigger.
3. **Multi-language support** Ability to switch languages without reloading the application. However, the initial implementation will be in English. All texts must be stored in configuration options outside of the source code. The documentation must contain information how to localize the admin backend to other languages.
4. **Maintainability** The codebase should be well-documented and modular for ease of updates.
5. **TYPO3** The administration backend must be implemented using TYPO3.

2.2.1.2 Other functional requirements

- **Login / Logout** The system must provide a secure login and logout mechanism for users. User sessions must timeout after a period of inactivity.

- **Security** The administration backend must comply with the security standards described in Section 2.5.2.

- **Toggle Visibility of Backend Views** Administrators must have the ability to enable or disable access to specific backend views. This allows for a more customized and secure administration interface. For example, one user group can access the RAG document management view only, while others can use more views.

- **Assign Views to Roles** Backend views must be assignable to specific roles. This can be configurable with a configuration file.

- **Unit and Integration Testing T**he application must include a comprehensive suite of unit and integration tests to ensure all functionality behave as expected and to guard against regressions during development.

- **User Groups** The administration backend supports the following user roles

    - **Frontend user** This user can login to the system, but he / she cannot access the administrator backend.

    - **Content Editor** This user can access the RAG document management system of the chatbot.

    - **Administrator** This user can use all functionalities of the backend.

2.2.1.3 Administration views

This section lists all administration views that will be implemented as part of the chatbot administration backend.

1. **User Management** Administrators can add, edit and remove users in the administration backend. Users will be described via some fields, such as name, email, password and user role.
2. **Update User Profile** Each user can change his information (name and password)
3. **Prompt Management** The administration backend has a functionality to create, list, update and delete prompts. Prompts will be identified by an ID. Also, the administration backend offers an ID that allows the LangChain service to load ID / prompt pairs via the administration backend. With the click of a button, the user can restart the LangChain service so that the changed prompts will be loaded inside in the LangChain service.

4. **RAG Document Management** The administration backend has a functionality to add documents to and remove documents from the RAG vector store.
    - Upload documents to the vector store. The system can process many document formats (PDF, MS Word, Plain Text, all documents supported by Apache Tika) by using the Apache Tika Library. These documents will be automatically split into segments of configurable length LangChain Text Splitters. After upload, the system will store both the original and the segmented document.
    - The administration website has a batch upload functionality to upload many documents at once (see Section 3.2.1.5).
    - The administration website offers an HTTP API to upload single documents and add them programmatically to the vector store.
    - Via the HTTP API, it is also possible to manually specify the segments of a document.
    - The RAG document Management shows a list of all the documents uploaded to the vector store.
        i. It has appropriate pagination, sorting and filtering functionalities to navigate large number of documents.
        ii. One can download the original document
        iii. See the open a document and see the segmentation
    - During the project, we will define additional metadata for the vector store which can be edited via the HTTP API and the RAG document management view. The metadata can also be defined during batch upload, so that all uploaded documents have the same metadata. Examples of this metadata:
        i. Additional filters for the search. For example, we can define a flag “active” to manually remove a document from the search results without deleting it. We can define an activity date so that a certain document is only available at certain dates.
        ii. Provide categorization to organize large document collections.
        iii. Store additional metadata such as creation date or created by (user)
    - The RAG document management will be able manage several vector databases, e.g., one for each language or to manage multiple bots within the same backend.
5. **Chatlog Analyzer** Each user / chatbot interaction will be stored in textual format in the central logging service (see Section 3.2.3). This view allows the analysis of these interactions:
    - “List conversations” shows all conversations in a table, one line per conversation. It has appropriate pagination, sorting and filtering functionalities to navigate large number of conversations.
    - Opening a conversation in List conversations shows a single conversation. This visualizes all information about that conversation that are stored in the database.
6. **Technical Log Analyzer** This view shows the technical logs from the central logging service (see Section 3.2.3) with appropriate pagination, sorting and filtering functionalities to navigate large number of log entries.
7. **Usage statistics** This view shows a dashboard that shows the most important statistics in a table (e.g., total number of conversations) and three plots with the most important

usage statistics (e.g., a line plot about “number of conversations in the last 90 days”). We will define the metrics and plots in the project. This data will be loaded from the central logging service (see Section 3.2.3).

8. **NLU Management** This view allows to generate training data for Natural Language Understanding (NLU). See the RASA documentation for definitions of the terms intents, entities and NLU training data.
    - Define intents and entities.
    - Create sentences for intents as training data for an NLU model.
    - Annotate entities in the training data.
    - A click on a button can train and deploy a new model (see Section 3.2.1.6).

2.2.1.4 Admin Backend APIs

- **User Management** The user administration backend offers an API that can
    - Create, read, update, delete users. This API endpoint requires authentication.
    - Handle authentication for other chatbot services.
- **Document upload** After authentication, one can upload documents to the document store. One can upload all formats that the RAG document management described above supports and they will be automatically segmented. Alternatively, one can upload the original document with its segments to skip the automatic segmentation and provide a custom segmentation. This API endpoint requires authentication.

2.2.1.5 Batch Upload

The system must provide a batch upload feature for the RAG document management. A user can batch upload many documents at the same time via the HTTP API or the RAG document management view.

- This will trigger an asynchronous batch job.
- The user can see the progress of the upload and error messages, e.g., if documents cannot be processed by the system.

2.2.1.6 NLU Model training and deployment

- Users can define NLU training data in the NLU management administration view. A click of a button trains and deploys a new RASA model.
- NLU management does not implement the full set of RASA features. E.g., it will leave out regular expressions or lookup tables. To be able to use these RASA features, we will implement a template NLU definition file that can contain this information. The NLU data defined via the UI will be inserted in this template so that the resulting NLU training data hosts both the data generated by the administration backend and defined in the template.

- The NLU training data will be send to the RASA NLU service which trains and deploys the model. The user sees feedback about the progress of the training.

2.2.1.7 Deliverables

- A service ChatFed service “administration backend”.
- A user’s manual in English that explains the functionalities of the backend from a user’s perspective.
- Technical documentation according to Section 2.5.

2.2.2 Generic RAG service

Key Tasks

- Implement a generic ChatFed orchestrator that integrates LangChain. This service is divided into two parts
  - Orchestrator: Common functionalities for all types of chatbots, e.g., logging, generation of conversation ids, …
  - LangChain for the LangChain / RAG
- Implement two generic RAG systems that serves as an example for the documentation. System one is the minimal example that, e.g., does not use an administration backend. System two is the full system including ChatFed services.
- Implement vector database. The contractor needs to decide on a suitable vector database to be used within the project. During the project, we will define our requirements. The contractor should then decide between Qdrant, Milvus and PgVector.
- Integrate embedding models (see Section 3.2.4).
- RAG systems require the conversation history which will be delivered by the central logging service.

Deliverables

- Generic ChatFed services for RAG
- Technical documentation according to Section 2.5.
- A tutorial “How to build a RAG system with ChatFed”

2.2.3 Central Logging Service

A distributed logging service to store technical logs and chat logs from all different services in a single database.

Key Tasks

- Implement a logging service that stores log messages from all other services.
- It uses an appropriate, industry standard database (e.g., Elastic Search with Kibana).

- The service offers an HTTP API to store log messages. It can store technical logs (such as error messages by the individual services) and chat logs.
- Chat logs are identified by a session id.
- Each log entry can have a specific tag, such as “general”, “error”, “user-message”, “bot-message” or “conversation-start”.
- The HTTP API has a functionality to retrieve the whole conversation by the chat session id. The LangChain service can retrieve the conversation history to a given conversation using the central logging service.
- Ensure that the central logging service complies, where applicable, with Rwandan Data Protection and Privacy Law and/or potentially other applicable local data privacy regulations and/or European GDPR (General Data Protection Regulation).

Deliverables

- The generic ChatFed service “Central logging service”
- Technical documentation according to Section 2.5.
- A documentation of the required format of log messages for compatibility with other parts of ChatFed, e.g., fetching the conversation history by the RAG system (Section 3.2.2) or the chat logs, technical logs and usage statistics by the administration backend (Section 3.2.1).

2.2.4 Embedding Models

RAG requires an embedding model that converts textual documents into vector embeddings for similarity search. GIZ will provide an embedding model for Kinyarwanda that was developed through the Farmer Decision Chatbot which is based on Pytorch. For English and French, we can reuse existing embedding models as a service through appropriate external APIs.

Key Tasks

- Implement a ChatFed service for the Kinyarwanda embedding model.

Deliverables

- A new generic ChatFed service.
- Technical documentation according to Section 2.5.

2.2.5 Automatic Speech Recognition and Text to Speech

Key Tasks

- Implement an Automatic Speech Recognition (ASR) ChatFed service. The contractor must research the best model, e.g., the Mamba ASR system. We need a system for Kinyarwanda, English and maybe other languages.
- Implement a Text to Speech (TTS) ChatFed service. The contractor must research the best available model, e.g., the TTS System of the Agricultural Chatbot which is based on PyTorch. We need a system for Kinyarwanda, English and maybe other languages.

Deliverables

- A ChatFed service for TTS
- A ChatFed service for ASR
- Technical documentation according to Section 2.5.

2.2.6 RASA Service

Key Tasks

- Create a RASA service for ChatFed for English, Kinyarwanda and French. RASA NLU supports English out of the box. Instructions for French are to be researched. English was already implemented by the Data Privacy Chatbot. RASA for Kinyarwanda was implemented by the Mbaza Chatbot.
- Offer HTTP APIs for
    - NLU (Intent detection and entity recognition).
    - Answer generation
    - Training and deployment of the NLU models triggered by the admin backend (see Section 3.2.1.6).

Deliverables

- A generic ChatFed service for NLU
- Technical documentation according to Section 2.5.

2.2.7 Configuration Service

Key Tasks

- Implement a centralized service for configuration to centrally store configuration options.
- Compare existing services, e.g., Spring Cloud Config and others and find the best open-source solution.

Deliverables

- A configuration ChatFed service.
- Technical documentation according to Section 2.5.

2.2.8 Proxy Server

Key Tasks

- Implement a reverse proxy server that serves as an entry point for all incoming requests to the chatbot from the outside based on NGINX.
- This proxy server defines which APIs and user interfaces are available from the outside (e.g. the orchestrators API to start chats or the administration backend) and which services cannot be called from the outside (e.g. internal APIs for the inter-service communication).

Deliverables

- A generic ChatFed service “proxy server”
- Technical documentation according to Section 2.5.

2.2.9 TYPO3 Widget Channel

Key Tasks

- Implement a generic TYPO3 website chat window as a channel for the chatbot.

Deliverables

- A TYPO3 website widget.
- Technical documentation according to Section 2.5.

2.2.10 USSD Channel

Key Tasks

- Implement a USSD Channel for our chatbots.

Deliverables

- A USSD channel service
- Technical documentation according to Section 2.5.

2.2.11 IVR Channel

Key Tasks

- Implement a IVR Channel based on Asterisk for our chatbots so that the chatbots can integrate into a telephone channel.

Deliverables

- A USSD channel service
- Technical documentation according to Section 2.5.

2.2.12 Multilingual Demonstrator

Key Tasks

- Implement a web-based user interface that interfaces with the standard ChatFed orchestrator.
- It features a text interface and a voice interface.
- It features a machine translation system to enable users to interact with the chatbot in another language. In the Agricultural Chatbot, for example, we had the problem that the chatbot speaks Kinyarwanda only and some developers of the Chatbot do not know this language. Instead, they want to interact with the bot in English. The machine translation system will use a LLM to translate textual (not voice) interaction to and from English. LLMs generally speak many different languages. The source and target languages of the machine translation system must be configurable.
- It is possible to deactivate the multilingual translation for the whole system by configuration in case it is not needed.
- The user can switch the multilingual translation on and off in the frontend. To save cost, it is switched off by default.

Deliverables

- A ChatFed service “multilingual demonstrator”.
- Technical documentation according to Section 2.5.

2.3 WP 3: Implementation of multiple chatbots

WP3 builds several chatbots from the building blocks implemented in WP2. We can either extend existing chatbots or build new chatbots.

Key Tasks

For each chatbot implementation, the following tasks will be carried out:

- **Requirement Gathering:** Engage with the use case owner to collect and document functional and non-functional requirements.
- **Data Collection Support**: Assist use case owners in gathering and preparing relevant datasets.
- **Chatbot Development:** Design, build, and configure the chatbot based on collected requirements.
- **Deployment:** Deploy the chatbot in the designated target environment.
- **Training for Data Management**: Train use case owners on how to add and manage chatbot data using the administration backend

- **Training for administration backend:** Provide guidance on how to independently perform routine administration tasks using the administration backend.
- **Internal Testing:** Conduct thorough internal testing to ensure functionality, reliability, and performance.
- **User Testing:** Facilitate testing with end users to validate usability and gather feedback.
- **Extension of ChatFed Services (if applicable):** Develop and integrate new ChatFed services based on specific use case requirements.

Deliverables

The following deliverables will be produced for each chatbot:

- **Requirements Documentation:** A comprehensive record of the use case requirements. The requirements will be collected in a series of workshops.
- **Dataset Package:** Collected datasets in collaboration with use case owners, including a manual outlining the data collection strategy to enable future updates.
- **Implemented Chatbot:** Fully functional chatbot deployed on the use case owner's system.
- **Training Workshops and Manuals (Data Management):** Workshops and written manuals to enable use case owners to update chatbot datasets independently.
- **Training Workshops and Manuals (Backend Use):** Workshops and written manuals focused on using the chatbot administration backend for common tasks.
- **Testing Reports:** Detailed reports on internal and user tests, including methodology, data used, outcomes, and analysis.
- **New ChatFed Services (if applicable):** Specifications and documentation of newly developed services tailored to the use case.

3 Timeline

The following table shows a timeline of completion. It encompasses three phases. Each of the phases ends with a milestone.

Phase	Duration	Key Activities	Milestone
1. Framework Implementation	Months 1–6	● Requirements gathering & stakeholder validation ● Core ChatFed engine development ● Service modules (NLU, dialogue management, channel adapters) ● Documentation & initial deployment scripts	M1

Phase	Duration	Key Activities	Milestone
2. Chatbot Development & Integration	Months 7–10	• Detailed use-case design workshops & user journey mapping • Build and configure Chatbot 1 (e.g. FAQ service) • Build and configure Chatbot 2 (e.g. form-based workflow) • Build and configure Chatbot 3 (e.g. voice-enabled interaction) • Channel-specific adaptations, API integrations & security review • End-to-end functional testing & performance tuning	M2
3. Testing & Ongoing Maintenance	Months 10–14	• Finalize chatbot deployments across targeted channels • Conduct user acceptance & accessibility testing • Monitor live performance, gather feedback • Bug fixes, updates & knowledge-transfer sessions	M3

Note: **Phases can run in parallel where feasible. Use-case implementations (Phase 2) may be sequenced by priority or by channel readiness to accelerate service delivery**.

Period of assignment, from contract start on **01.01.2026** until the end **30.11.2026**.

4 Technical-methodological concept

In the conceptual design of the tender (technical-methodological approach, project management, if necessary other requirements), the tenderer is required to take specific objectives and requirements into consideration and describe them, as explained below.

4.1 Strategy:

The tenderer must assess the objectives, critically examine the tasks for feasibility, and identifies relevant technical and non-technical constraints (Section 1.1 of the assessment grid). This includes description of how required services are provided and delivered to achieve the objective.

4.2 Cooperation:

. The tenderer presents how roles, responsibilities, and information flows will be arranged among relevant actors (Section 1.2(a) of the assessment grid). With a clear strategy for establishing cooperation and working together throughout delivery (Section 1.2(b) of the assessment grid).

4.3 Steering Structure:

The tenderer explains how measures will be steered with project partners, including decision paths and escalation points in case of prolonged delays (Section 1.3(a) of the assessment grid). The tenderer also describes methods for monitoring results and mitigating technical bottlenecks (Section 1.3(b) of the assessment grid).

4.4 Processes:

The tenderer presents an implementation plan that explains work steps, milestones, schedule, and the development/implementation methodology (Section 1.4(a) of the assessment grid). The tenderer shows how partner contributions will be integrated into these processes (Section 1.4(b) of the assessment grid).

4.5 Learning and Innovation:

The tenderer describes knowledge-management, testing, and documentation practices, including IT security and documentation standards, to ensure maintainability and reproducibility (Section 1.5(a) of the assessment grid). This includes staff and/or community training or similar measures that support sustainability and scale-up (Section 1.5(b) of the assessment grid).

4.6 Project management of the contractor:

The tenderer sets out methods for coordinating implementation with GIZ and key stakeholders, including reporting cadence and change control (Section 1.6(a) of the assessment grid). In addition, provides a personnel assignment plan according to the personnel concept below and expert days for each key activity per phase (Section 1.6(b) of the assessment grid).

5 Personnel concept

The tenderer is required to provide staff for the positions ("experts') referred to and described here in terms of the scope of tasks and qualifications on the basis of corresponding CVs.

The qualifications specified below meet the requirements for achieving the highest score in the technical assessment. Criteria weights are given in brackets. If no weights mentioned, they are weighted equally.

Please list relevant references according to the qualifications structure.
If project references are requested, concrete examples of work with a minimum duration of 15 days should be listed, no experiments, proof of concepts or similar.

The below specified qualifications represent the requirements to reach the maximum number of points. The bidder's personnel concept may assign any number of experts to each role; however, must cover all roles and required profiles listed below.

Please note that the below-specified qualifications represent the ideal requirements to reach the maximum number of points – not fulfilling the below-specified qualifications fully will not lead to an exclusion of the bidder.

We encourage consortia of multiple companies to apply for this tender to optimally cover the required expertise.

5.1 Team Leader: Project lead

Main tasks of team leader

- Overall responsibility for the tasks to be performed by the contractor.
- Coordinating work across technical and functional streams
- Facilitate communication between technical team, GIZ, and client stakeholders (e.g., MINAGRI, DPO, RISA)
- Personnel management, in particular managing the assignment within the available budget, as well as planning and steering assignments.
- Regular reporting in accordance with deadlines, ensure resource alignment and scope control
- Coordinate technical implementation of the project, especially regarding architecture and interoperability
- Manage risk register and mitigation actions.

Qualifications of team leader

- Education/training (2.1.1): Master's level degree in computer science, data science, machine learning, artificial intelligence, natural language processing, computational linguistics or in related fields.
- Language (2.1.2): C2-level language proficiency in English.
- General professional experience (2.1.3): 5 years of experience in project or product management roles in software/AI projects.
- Specific professional experience (2.1.4):

    - 5 years of experience in machine learning, artificial intelligence or natural language processing, experiences with chatbot project management
    - 5 years of experience applying agile project management frameworks (Scrum, Kanban)
    - Experience in public sector or civic tech projects is a strong plus.
- Leadership/management experience (2.1.5): 5 years of experience in managing cross-functional teams
- Regional experience (2.1.6): 2 years of experience in East Africa
- Development Cooperation (DC) experience (2.1.7): 2 years of experience working with German Development Cooperation or other development cooperation partners in digital transformation projects

5.2 Key Expert 1: AI Chatbot Engineer

Main tasks of AI Chatbot Engineer

- Support the implementation of the AI components
- Build, train, evaluate und deploy AI models, especially RASA and LangChain
- Develop interoperability standards
- Write documentation
- Support data collection
- Documentation

Qualifications of AI Chatbot Engineer

- Education/training (2.2.1): Master's level degree in computer science, computer engineering, or related relevant field
- Language (2.2.2): Good business language skills in English.
- General professional experience (2.2.3): 5 years of professional experience in AI chatbot engineering or related fields
- Specific professional experience (2.2.4):
    - 3 years of hands-on experience in NLP technologies for Kinyarwanda
    - Strong understanding of the relevant chatbot technologies RAG Systems, LangChain, RASA
    - Proficient in Python and relevant ML libraries (Hugging Face, PyTorch, etc.,)
- Regional experience (2.2.6): 2 years of experience in Rwanda
- Development Cooperation (DC) experience (2.2.7): -
- Other (2.2.8): -

5.3 Key Expert 2: Backend Developer

Main tasks of Backend Developer

- Technical implementation of the services (non-AI parts)

- Design and implement a modular chatbot backend framework
- Dockerization of the services
- Define interoperability standards
- Documentation and Knowledge transfer

Qualifications of Backend Developer

- Education/training (2.3.1): Master's level degree in computer science, computer engineering, or related relevant field
- Language (2.3.2): C2-level language proficiency in English
- General professional experience (2.3.3): 6 years of experience in software engineering; implementing backend services for multi-channel communication systems, including USSD
- Specific professional experience (2.3.4): 5 years of experience
    - Proficiency in Python or Java
    - Proficiency with Docker
    - Strong understanding of cloud / microservice architectures
- Regional experience (2.3.6):

5.4 Key Expert 3: Web Frontend Developer

Main tasks of Web Frontend Developer

- Implementation of user interfaces, especially using TYPO3

Qualifications of Web Frontend Developer

- Education/training (2.4.1): Bachelor's degree in computer science, computer engineering, or related relevant field
- Language (2.4.2):. C2-level language proficiency in English.
- General professional experience (2.4.3): 5 years of experience in frontend web development
- Specific professional experience (2.4.4): 3 years of experience
    - Web development
    - Development with TYPO3
    - Relevant web frontend technologies such as React, Vue.js or similar
    - Understanding of multilingual UI practices
    - Experience with GitHub Pages / Jekyll
- Regional experience (2.4.6):

5.5 Key Expert 4: Conversational Design, User Experience

Main tasks of Conversational Design, User Experience

- Design and document end-to-end multilingual conversation flows, including intents, entities, slots, and fallback strategies for all use cases; ensuring clarity, consistency, and adaptability across RASA, LangChain, and other chatbot frameworks

- Develop a structured quality assurance framework and auditing toolkit to evaluate accuracy and usability of chatbot outputs; including tone and style aligned with corresponding product owner's communication standards

- Lead user research and usability testing, gathering insights to improve interaction design and inform user personas

- Collaborate with AI chatbot engineer and developers to implement and optimize conversation logic, multilingual support, and RAG integrations, providing hands-on review

- Create training material and deliver knowledge transfer to build internal capacity; equipping service and IT teams of product owners to maintain and evolve the chatbot(s) independently

Qualifications of Conversational Design, User Experience

- Education/training (2.4.1): Bachelor's degree in computer science, computer engineering, or related relevant field
- Language (2.4.2): C2-level language proficiency in English (6 points) and fluent in Kinyarwanda (4 points).
- General professional experience (2.4.3): 5 years of experience in designing conversational experiences for AI-powered chatbots or virtual assistants across multiple frameworks (RASA, Dialogflow, LangChain, or equivalent)
- Specific professional experience (2.4.4): 3 years of experience
    - Applying user-centred design in multilingual
    - Developing style guides, tone of voice guidelines, and training materials for conversational products
    - Prototyping and conversation mapping tools (e.g., Voiceflow, Figma, etc.,)
- Regional experience (2.4.6): 2 years of experience in East Africa

5.6 Short term expert 1: Business analyst

Main tasks of Business analyst

- Facilitate stakeholder consultations (government agencies, service providers, communities of practice, development partners etc.,) in close coordination with the AI Innovation Lab and RISA's Centre for DPI
- Document and validate user stories for each chatbot framework channel (e.g., USSD, web, IVR)

- Translate business needs into detailed epics, features, and user stories.
- Develop business models and sustainability plans
- Assess change impacts and recommend adoption strategies for service and IT teams of product owners
- Collaborate with technical writer to develop briefing materials and onboarding support for non-technical stakeholders.
- Support definition of success metrics (e.g., time savings, service reach, user satisfaction)

Qualifications of Business analyst

- Education/training (2.5.1): Master's degree in business administration, information technology, or related relevant field
- Language (2.5.2): C2-level language proficiency in English (6 points) and fluent in Kinyarwanda (4 points).
- General professional experience (2.5.3): 5 years of experience in public service digitalization, ICT for development, or digital governance projects
- Specific professional experience (2.5.4):
    - 5 years of experience in requirements engineering (with exposure to technical and non-technical stakeholders)
    - 3 years of experience with voice technology or NLP/AI systems
    - Experience using tools such as Jira, Confluence, and Miro
- Regional experience (2.5.6): 2 years of experience in East Africa
- Other (2.5.7):

5.7 Short term expert 2: Technical Writer

Main tasks of Technical Writer

- Write documentations, reports and training materials

Qualifications of Technical Writer

- Education/training (2.6.1): Bachelor's degree in information technology, computer engineering, or related relevant field
- Language (2.6.2): C2-level language proficiency in English (6 points) and fluent in Kinyarwanda (4 points).
- General professional experience (2.6.3):3 years' experience in technical writing, preferably in software or AI domains
- Specific professional experience (2.6.4):
    - Excellent written English and ability to simplify complex systems for different audiences
    - Familiarity with documentation platforms (e.g., Markdown, Docusaurus, Git-based docs).

- Regional experience (2.6.6):
- Development Cooperation (DC) experience (2.6.7): -
- Other (2.6.8): -

6 Consortium-based Collaboration

Given that this project builds on learnings from previous pilot-level initiatives and the multidisciplinary nature of the chatbot framework—spanning conversational AI, voice and text interfaces, multilingual NLP, inclusive design, and public sector integration—**bidders may, but not required, to apply as a consortium of complementary service providers or startups.**

This approach can offer a structured way to align formerly siloed contributions, mitigate gaps in technical or contextual knowledge, and support long-term maintainability.

6.1 Requirements for consortium applications

- **Lead Entity Designation**: One organization must act as the lead applicant, assuming responsibility for contract management, coordination, and reporting.
- **Defined Roles and Contributions**: Each consortium member must be listed with a description of their specific responsibilities within the project.
- **Consortium Agreement or MoU**: A signed letter of intent or formal agreement detailing collaboration mechanisms, task distribution, and decision-making processes must accompany the proposal.
- **Capacity and Experience Profiles**: All members should submit a brief statement highlighting their relevant track record and how their expertise strengthens the joint offer.

7 Costing requirements

7.1 Assignment of personnel and travel expenses

The total number of expert days allocated to this assignment is **702 days for the contractor.**

The fixed, unalterable travel-expense budget for all trips within Rwanda for the service provider is **7.800.000 RWF.** Additional details are described in the table below, and daily rates for per-diem allowances, transport costs, and accommodation are specified in the price schedule.

Per-diem allowances are reimbursed as a lump sum up to the maximum amounts permissible under tax law for each country as set out in the country table in the circular from the German Federal Ministry of Finance on travel expense remuneration (downloadable from the German Federal Ministry of Finance – tax treatment of travel expenses and allowances for international business travel as of 1 January 20242025 (GERMAN ONLY)).

Accommodation allowances are reimbursed as detailed in the specification of inputs below.

With special justification, additional Accommodation costs up to a reasonable amount can be reimbursed against evidence.

Accommodation costs which exceed this up to a reasonable amount and the cost of flights and other main forms of transport can be reimbursed against evidence.

All business trips must be agreed in advance by the officer responsible for the project.

7.2 Sustainability aspects for travel

GIZ would like to reduce greenhouse gas emissions (CO_2 emissions) caused by travel. When preparing your tender, please incorporate options for reducing emissions, such as selecting the lowest emission booking class (economy) and using means of transport, airlines and flight routes with a higher CO_2 efficiency. For short distances, travel by train (second class) or e-mobility should be the preferred option.

If they cannot be avoided, CO_2 emissions caused by air travel should be offset. GIZ specifies a budget for this, through which the carbon offsets can be settled against evidence.

There are many different providers in the market for emissions certificates, and they have different climate impact ambitions. The Development and Climate Alliance (German only) has published a list of standards (German only). GIZ recommends using the standards specified there.

7.2.1 Specification of inputs

Fee days	Number of experts	Number of days per expert	Total	Comments
Team Leader (Project Lead)	1	133	133	Responsible for overall coordination, planning, stakeholder engagement, delivery tracking, and risk mitigation across all work packages.
Key expert 1: AI Chatbot Engineer	1	124	124	Leads implementation of core AI services including RAG, NLP model integration, embedding models, and multilingual language support.
Key expert 2 : Backend Developer	1	175	175	Develops backend services, APIs, and integrates core

				modules of the ChatFed framework.
Key expert 3 : Web Frontend Developer	1	90	90	Builds user-facing interfaces for ChatFed admin portal, feedback dashboards, and web widget. Works closely with backend dev and technical writer
Key expert 4 : Conversation Design, User Experience	1	60	60	Leads the design and optimization of user interactions across the chatbot solution ensuring that conversations are clear, accessible, and accurate.
Short term expert 1: Business analyst	1	60	60	Translate stakeholder needs into clear technical and functional requirements to guide the design, development, and deployment of the chatbot framework in real-world service delivery contexts
Short term expert 2: Technical Writer	1	60	60	Develops all documentation including system architecture, training guides, API references, SOPs, and user manuals.
Travel expenses	**Quantity**	**Number per expert**	**Total**	**Comments**
Fixed travel budget	1	7.800.000 RWF	7.800.000 RWF	A budget is earmarked for travel in the following country: **Rwanda**. A fixed budget of **7.800.000 RWF** is earmarked for settling travel expenses against evidence. The travel budget contains ● Per-diem allowances and overnight allowances in the country of assignment

				● Travels within the country of assignment, transfer to/from places of work of product owners (outside of Kigali), on-ground data collection, etc., You can find the further information on the travel expense budget in the 'Price Schedule' document. Please use the 'Explanations' column in the price schedule to break down the individual items. Settlement is possible only until the budget is depleted.
Other costs	**Number**	**Price**	**Total**	**Comments**
Flexible remuneration	1	28,000.000 RWF	28,000.000 RWF	A budget of 28,000.000 RWF is foreseen for flexible remuneration. Please incorporate this budget into the price schedule. Use of the flexible remuneration item requires prior written approval from GIZ.
Other costs The bidder is required to incorporate costs of **hosting the staging environment** (for the duration of the contract) as well as production environment up to **a period of 18 months**. These environments should generally accommodate: ● High computing capacity for AI workloads	1	25.000.000 RWF	25.000.000 RWF	The budget contains the following costs: Hosting the staging environment (including computing capacity and storage) for up to 25.000.000 RWF

• Ability so scale storage as data needs grows • High uptime and availability of all services – with a reputable global infrastructure provider • Industry-grade security of the servers				
Other Costs The bidder is required to incorporate costs of the **3rd party services to access large language models (LLM credits) and language model development costs (AI training)**	1	13.400.000 RWF	13.400.000 RWF	Costs for using LLM services provided via 3rd party APIs such as OpenAI, Groq, etc., Costs for finetuning of large language models (LLMs) to Kinyarwanda and/or French.

7.2.2 Travel

Most meetings are to be implemented as virtual workshops. However, for more buy-in from decision-makers and in-person user research, physical meetings may need to be arranged – especially at project kick-off, any specific high-level engagements and user research and testing events, including the launch event.

All travels must be approved by GIZ and settled against provision of evidence (e.g. vouchers, receipts, invoices, boarding passes).

If the circumstances allow for travel, the bidder is required to calculate the travel by the specified experts and the experts it has proposed based on the places of performance stipulated in Chapter 2 and list the expenses separately by daily allowance, accommodation expenses, flight costs and other travel expenses.

Please note: National accommodation and allowance will only be paid on a need's basis (= if workshop is not in same town in which person is located which fulfils this assignment).

Workshops, training

Expenses for workshop and training logistics (e.g. venue and catering) will be directly covered by GIZ and should not be included in the financial offer.

8 Inputs of GIZ or other actors

GIZ and/or other actors are expected to make the following available:

- Logistics for workshops: venue, catering, and daily subsistence for participants.

9 Requirements on the format of the bid

The structure of the tender must correspond to the structure of the ToR. It must be legible (font size 11 or larger) and clearly formulated. The language in which the tender must be written is English

The technical-methodological concept of the tender (Section 5 of the ToR) is not to exceed 20 pages (not including the cover page, list of abbreviations, table of contents and brief introduction).

The CVs of the staff proposed in accordance with Section 6 of the ToR must be in the EU format and not exceed four pages in length. The CVs must clearly show what position the proposed person held, which tasks he or she performed and how many expert days he or she worked during which period in the specified references. The CVs can also be submitted in a different language, namely English.

We strongly request that you do not exceed the number of pages specified.

10 Requirements regarding the development of AI systems

The service provider commits to not place any AI systems, which will be requested by a GIZ partner and developed in context of this contract, on the European Union market or put the AI system into service in the European Union. Additionally, the service provider shall not use any outputs produced by the AI system in the European Union in any manner.

Furthermore, the service provider commits to ensure that ALL AI systems, which will be developed in context of this contract, do not fall under the prohibited or high-risk classification of the EU AI Act (respectively Article 5 and Article 6 as well as Annex 3).

If, exceptionally, individual use cases that are explicitly requested by GIZ could result in a linkage to the European Union market, GIZ reserves the right to conduct an internal assessment to decide whether the EU AI Act may apply. In this case, the service provider commits to complying with respective articles of the EU AI Act. Additionally for these AI systems, GIZ and the service provider commit to neither requesting nor developing AI systems that are either classified as prohibited or high-risk under the EU AI Act.

11 Data Protection

The data protection and information security provisions set out in the most recent GIZ AVB (sections 1.10 Data Protection and 1.6. Confidentiality) apply.

The performance of the contract may involve the processing of personal data by the contractor. In such cases, the contractor shall act as an independent Data Controller and shall comply with ALL applicable data protection obligations, including those arising from regional and local laws.

The contractor may process personal data only if the goal to be achieved cannot be achieved without such data. The data protection principles such as lawfulness, data minimization, accuracy, purpose limitation, storage limitation, transparency, integrity and confidentiality and accountability as well as the numerous rights of the data subjects must be considered. GIZ is in no way responsible for such processing.

In cases where the contractor follows the instructions of a GIZ partner, the partner is the data controller. The laws and standards applicable to the GIZ partner and the contractor must be complied with and implemented.

GIZ reserves the right to conduct a risk assessment before an AI system will be developed through this service contract. This might be the case if a proposed AI system might entail high-risks for the fundamental rights or freedom of local beneficiaries or high risks for GIZ itself. Should the assessment conclude a high residual risk even after the consideration of mitigation measures, GIZ reserves the right to not finance the development of this use case via this service contract.

If the contractor is not subject to the GDPR and the applicable laws do not provide explanations of the data protection principles and rights mentioned here, the definitions of the GDPR (Regulation (EU) 2016/679) should be used.

If, exceptionally, individual use cases require the processing of personal data on behalf of GIZ, an agreement of outsourcing the processing of personal data (AuV) must be concluded with the Contractor in accordance with Art. 28 GDPR for these services. In these cases, the technical and organizational measures (TOM) for compliance with the data protection requirements by GIZ must be presented before the contract for the short ToRs are concluded.

12 Submission of the offer

11.1 Technical offer

Technical offer must include the following documents:

Eligibility documents:

- Self-declaration of eligibility for the award
- Company administrative documents: registration certificate (RDB), VAT registration certificate and Valid Tax clearance certificate
- Three Company References for the completion of similar assignments in the last 3 years, as described in the eligibility assessment grid

Technical proposal:

- Technical Proposal (**attached template for technical proposal MUST be used)**
- Up to date CVs of proposed experts

11.2 Financial offer:

Financial offer indicates the all-inclusive total contract price, supported by a breakdown of all costs as described in the specification of inputs. The costs **must be in RWF and VAT** excluded (**Price sheet must be used**).

Your EoI has to be submitted in **2 separated emails** to **RW_Quotation@giz.de** until latest **28.11.2025:**

1. **The technical offer** has to be submitted in 2 **PDF file format (eligibility/technical)** and **as attachment to the email** with the subject: **83493986 -Technical offer.**
2. **The financial offer** has to be submitted in **PDF format** and **as attachment to the email** with the subject: **83493986 - Financial offer.**

If the emails exceed the default email size of **30MB**, offers can be exceptionally submitted through **https://filetransfer.giz.de/, as indicated. The subject of recipient notification must be edited with the subject indicated above** and the notification message **must include the password to access the files.**

Offers submitted through any other sharing platform, as google documents or similar will not be considered.

Without the subject mentioned, your offer may not be considered

Offers submitted in hard copy will not be considered.

GIZ reserves all rights.

Abbreviations

AI	Artificial Intelligence
DPG	Digital Public Good
DTC	Digital Transformation Center
GIZ	Deutsche Gesellschaft für Internationale Zusammenarbeit
IVR	Interactive Voice Response
NLP	Natural Language Processing
NLU	Natural Language Understanding
RAG	Retrieval Augmented Generation
ChatFed	Chatbot Federation
RISA	Rwanda Information Society Authority
MINICT	Ministry of ICT and Innovation
DPO	Data Protection Office
NAIS	Rwanda's National Agriculture Insurance Scheme
MINAGRI	Ministry of Agriculture and Animal Resources
WP	Work Package
ASR	Automatic Speech Recognition
TTS	Text to Speech
RBDS	Rule-Based Dialog System
DPI	Digital Public Infrastructure