

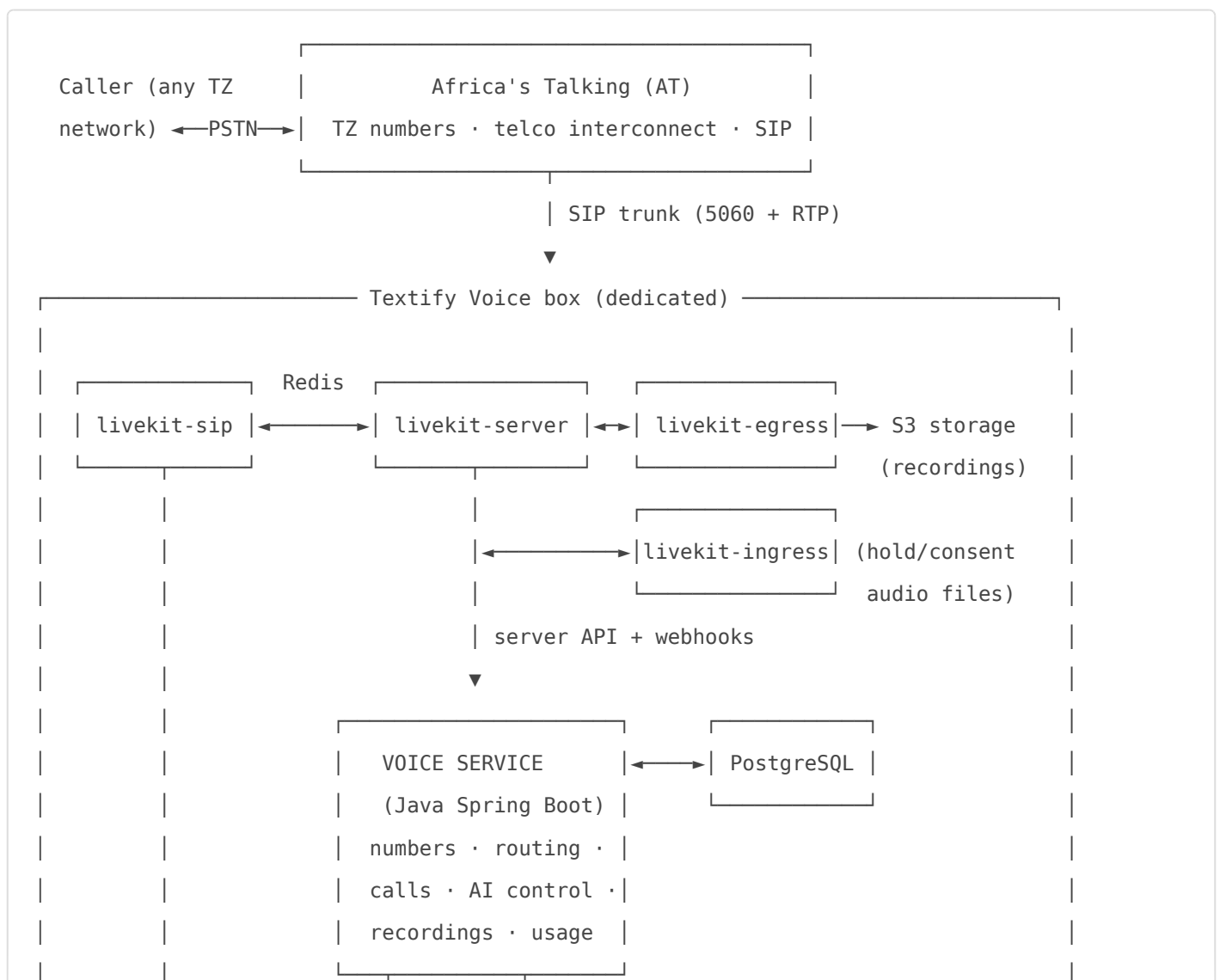
TextiFy Voice Service

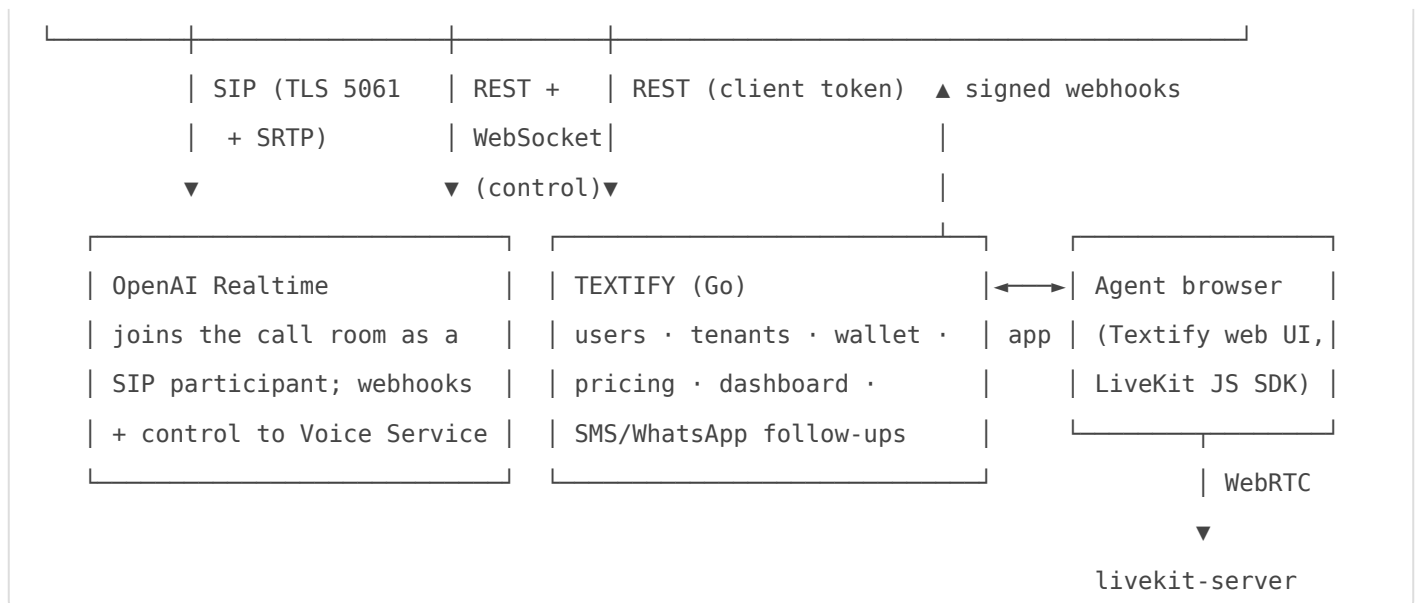
- [Textify Voice: Architecture](#)
- [Textify Voice: Product & Functional Specification](#)

Textify Voice: Architecture

“ Companion to `SPEC.md` (what and why). This file is **how**: components, boundaries, call flows, data model, APIs, events, infrastructure, security. Stack: **Textify (Go) + Voice Service (Java Spring Boot) + LiveKit (self-hosted) + OpenAI Realtime (over SIP) + Africa's Talking. No Python.** Anything marked **[VERIFY]** must be checked against current provider docs or Africa's Talking before coding against it.

1. System overview





Key idea: **every phone call is one LiveKit room**. The caller, the AI and human agents are all participants in that room:

- the **caller** arrives from AT over the SIP trunk;
- the **AI** is OpenAI Realtime, which Spring Boot **dials into the room over SIP** and controls through OpenAI's call-control REST + WebSocket;
- **human agents** join from the browser over WebRTC.

Spring Boot is the only "brain". Nothing else has business logic.

1.1 Responsibilities (hard boundary)

Component	Owns	Must NOT own
Textify (Go)	Users, login, roles, tenants, partners/white-label, wallet, pricing, invoices, dashboard UI, SMS/WhatsApp follow-ups	Any telephony, LiveKit or OpenAI logic
Voice Service (Spring Boot)	Client apps, accounts (opaque ids), numbers + config, agents (opaque refs), ring groups, call lifecycle, AI call control (OpenAI accept/tools/hangup) , recordings, transcripts, summaries, usage events	Users, passwords, roles, plans, prices, money
LiveKit (server + SIP + egress + ingress)	Real-time media, SIP termination, rooms, recording to storage, playing audio files into rooms	Business logic
OpenAI Realtime	Listening, thinking and speaking during AI calls (speech-to-speech); transcription + summaries after calls	State, routing decisions
Africa's Talking	Numbers, PSTN, SIP trunk	Anything else

Rule of thumb: if a feature mentions **people, plans, roles or money**, it belongs in Textify. If it mentions **calls, audio, numbers or AI behaviour on a call**, it belongs in the Voice Service.

1.2 Why a separate Voice Service

- **Reusable:** JikoXpress, Mauzodukani and NexGate can become clients later (each is a `client_app` with accounts).
 - **Failure isolation:** a portal outage or deploy doesn't drop calls, and the other way round.
 - **Security:** the box with SIP ports open holds no user credentials.
 - **One language for the call logic:** Java (team strength). Textify stays Go.
-

2. Component details

2.1 Voice Service (Java Spring Boot)

Stack: Java 21, Spring Boot 3.x, PostgreSQL + Flyway, Redis (locks, presence, short-lived call state), `@Scheduled` jobs (no Kafka at this scale), virtual threads (one OpenAI WebSocket per active AI call is fine).

Libraries

- LiveKit server SDK for JVM: `io.livekit:livekit-server` (Kotlin, usable from Java). Room service, SIP service (trunks, dispatch rules, `CreateSIPParticipant`, `TransferSIPParticipant`), egress, ingress, `AccessToken`, `WebhookReceiver`. **[VERIFY]** exact class/method names in the current version.
- OpenAI: plain HTTPS (`RestClient`) for call accept/reject/refer/hangup, transcription and summaries; Java WebSocket client (`java.net.http.WebSocket` or Spring `WebSocketClient`) for live call control.
- AWS SDK v2 S3 client (S3-compatible storage) for recordings.

Package layout

```
com.qbitspark.voice
├─ api           // client-facing REST (/v1/...), client-token auth
├─ hooks         // inbound webhooks: /hooks/livekit, /hooks/openai (signature-verified)
├─ livekit       // SDK wrapper, tokens, trunk/dispatch bootstrap, ingress (hold audio),
egress
├─ ai            // OpenAI call control: accept + session config, WebSocket session handler,
|               //   tool execution, AI usage capture; post-call transcription + summary
└─ calls         // call lifecycle state machine, call timeline
```

└ routing	// ring groups, agent presence, accept/claim logic
└ recording	// egress start/stop, storage, retention job
└ events	// outbox table + dispatcher (signed webhooks to client apps)
└ accounts	// client apps, accounts, numbers, number config, agents
└ common	// errors, ids, time, config

2.2 AI on calls: OpenAI Realtime over SIP, controlled from Java

```

Caller —AT→ LiveKit room "call-..." (caller = SIP participant)

      ▲
      | (1) Spring Boot: CreateSIPParticipant(outbound trunk "openai",
      |                                     sip_call_to = <OpenAI project SIP URI>,
      |                                     header X-Voice-Call-Id = <callId>)
      ▼
      OpenAI Realtime (SIP, TLS 5061 + SRTP)
      | (2) webhook realtime.call.incoming {call_id, sip_headers} → POST
/hooks/openai
      | (3) Spring Boot: POST /v1/realtime/calls/{call_id}/accept
      |           {model, voice, instructions, tools, transcription on}
      | (4) Spring Boot: WebSocket
wss://api.openai.com/v1/realtime?call_id={call_id}
      |           ← function calls, transcripts, usage
      |           → tool results, "say this" responses
      | (5) Spring Boot: POST /v1/realtime/calls/{call_id}/hangup (AI leaves;
room stays)

```

- **Correlation:** pass our `callId` as a custom SIP header on `CreateSIPParticipant`; read it back from `data.sip_headers` in OpenAI's webhook. **[VERIFY]** LiveKit supports custom headers on outbound SIP participants and OpenAI forwards them. Treat headers as untrusted: only accept calls whose `callId` exists in our DB with status `AI_DIALING`.
- **Session config on accept** (built from the number's config):
 - `instructions`: base receptionist rules (Swahili first, short spoken answers, never invent prices, offer a human when unsure) + business name + `ai_instructions` + "start by saying the recording notice and greeting: ...".
 - `voice`: chosen after the Swahili test.
 - `tools`: `transfer_to_human`, `take_message(name, phone, reason)`, `end_call`. (Phase 2: account-defined tools proxied to the client app.)
 - Input transcription on (for live transcript + summaries).
- **Tool handling** (on the WebSocket, in Java):

- `transfer_to_human` → routing rings agents (§3.2) → tool result `"ringing"` → AI tells the caller to hold → on accept, AI briefs the agent in one sentence → Spring Boot hangs up the OpenAI leg.
- `take_message` → save to `calls.message_taken` → event `call.message_taken`.
- `end_call` → wait for the AI to finish speaking → delete the LiveKit room.
- **AI usage:** capture usage from OpenAI response events and the leg duration (`participant_joined/left` of identity `ai`) → `calls.ai_seconds`, `calls.ai_usage`.
- **OpenAI API variants:** OpenAI documents both the Realtime API (`/v1/realtime/calls/...`) and a newer Live API (`/v1/live/sessions/...` , webhook `live.transport.incoming`). Pick one at build time **[VERIFY]**. The design is the same.
- **Requirements:** LiveKit outbound trunk to OpenAI uses **TLS (5061) + SRTP**; OpenAI requires SRTP, and trunks must support Opus with SDES-SRTP. LiveKit SIP supports TLS/SRTP. Interop **[VERIFY]** in the PoC.
- **Trade-off:** locked to OpenAI's realtime voices and turn detection. If OpenAI's Swahili speech isn't good enough → Appendix A.

2.3 LiveKit (self-hosted, dedicated box)

Service	Purpose
<code>livekit-server</code>	Rooms, WebRTC for browser agents, server API, webhooks to the Voice Service
<code>livekit-sip</code>	Terminates the AT trunk (caller = SIP participant); dials OpenAI over SIP
<code>livekit-egress</code>	Audio-only room recording → S3-compatible storage
<code>livekit-ingress</code>	Plays pre-recorded audio (consent notice, hold music, "no balance" message) into a room from a URL [VERIFY] URL input with audio-only files
<code>redis</code>	Required by server, SIP, egress and ingress

Telephony objects (created **once** at bootstrap, reused for every call):

- **Inbound trunk "at-inbound":** our AT numbers + AT IPs only (`numbers`, `allowed_addresses`).
- **Dispatch rule:** `dispatch_rule_individual`, `room_prefix: "call-"`, **no agent dispatch** (Spring Boot reacts to webhooks instead).
- **Outbound trunk "at-outbound":** AT SIP host + credentials, for calls to customers.
- **Outbound trunk "openai":** OpenAI SIP host, TLS transport, SRTP on, for dialing the AI into rooms.

SIP participant attributes used: `sip.phoneNumber` (caller), `sip.trunkPhoneNumber` (our dialed number), `sip.callID`. **[VERIFY]** names, and that they are included in LiveKit webhook participant info.

LiveKit SIP limitations: **no SIP REGISTER** (AT must send calls to our SIP URI directly); no video over SIP; DTMF, cold transfer (REFER), warm transfer, TLS/SRTP and noise cancellation supported. AT is

not on LiveKit's tested-provider list, so **the PoC must prove compatibility**.

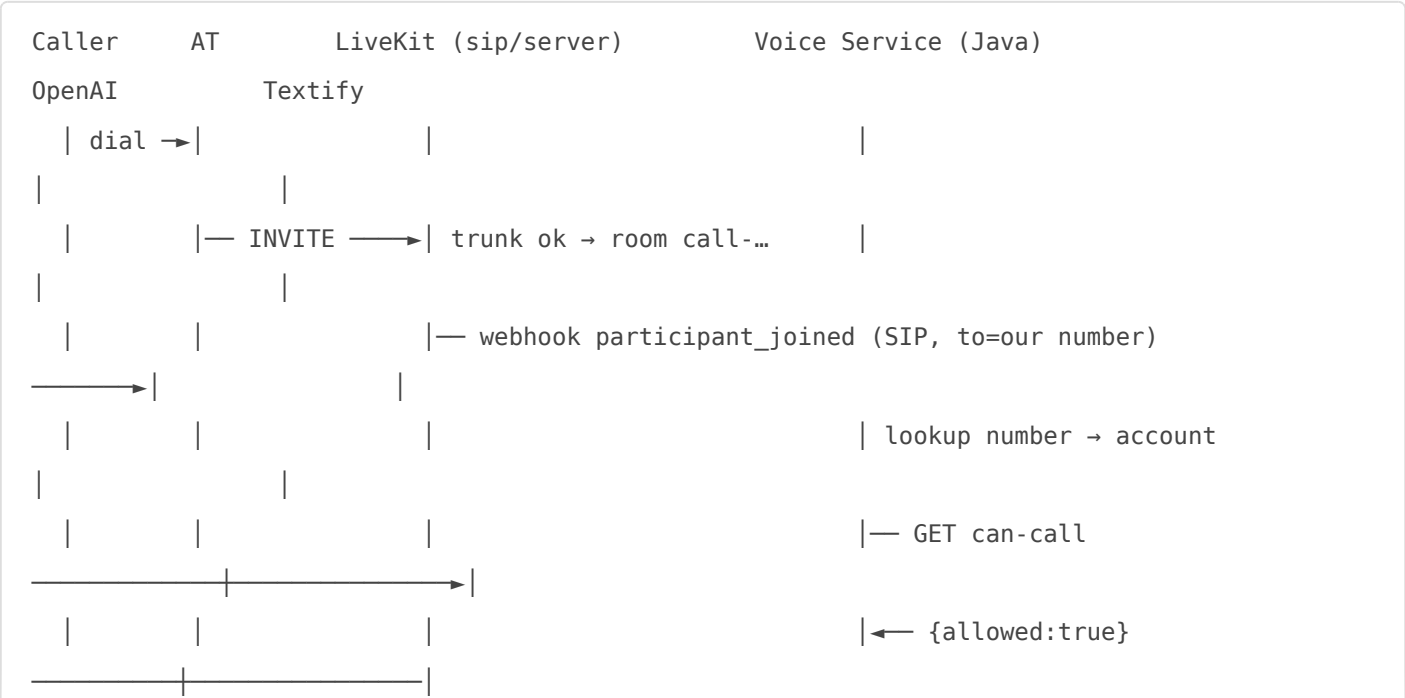
Room naming: the individual dispatch rule puts the caller's number in the room name (PII in logs). We can't control the `To` header on calls from AT, so this is accepted for Phase 1 with access-controlled logs.

2.4 Browser agent client (inside Textify's web app)

- LiveKit JS SDK. Room tokens are minted by the Voice Service and fetched **through Textify's backend**; the browser never holds Voice Service credentials.
- **Lobby room per agent:** `lobby-{accountId}-{agentRef}`, connected (mic off) while the agent is available:
 - **Presence:** lobby `participant_joined/left` webhooks = online/offline, automatically.
 - **Ringling:** the Voice Service sends a data message `{type:"ring", callId, caller, numberLabel}` into the lobby → incoming-call UI + ringtone.
- Accept → Textify backend → `POST /v1/calls/{id}/accept` → first accept wins → returns a call-room token → the browser joins with mic on.
- Losers get `409` and a `{type:"ring_cancel"}` data message.

3. Call flows

3.1 Inbound, AI_FIRST (main flow)





If `can-call` says no: play the "insufficient balance" audio via ingress → delete the room → call `REJECTED_NO_BALANCE` (no AI cost). If Textify doesn't answer within 1.5 s: **allow** (fail-open) and log it.

3.2 AI ? human handover (warm transfer)

1. AI calls tool `transfer_to_human` (caller asked for a person, is upset, or the AI can't answer).
2. Voice Service: find the number's ring group → available agents (Redis presence) → send `ring` data messages to their lobbies (ALL_AT_ONCE or ROUND_ROBIN) → tool result `{"status": "ringing"}`.
3. AI tells the caller "nakuunganisha na mhudumu sasa hivi" ("I'm connecting you to a staff member now").
4. Agent accepts → claim (`SET call:{id}:agent NX`) → call-room token → agent joins.
5. Voice Service sends the AI a "brief the agent in one sentence" instruction over the WebSocket → after it speaks → `POST .../calls/{call_id}/hangup` on OpenAI → AI leg leaves; caller and agent keep talking. AI seconds stop.

6. Timeout (default 30 s) with no accept → tool result `{"status":"no_agent"}` → AI takes a message → `call.message_taken`.

3.3 Inbound, HUMAN_ONLY

1. Same start as 3.1 (webhook → number lookup → can-call → call record).
2. **No AI dialed.** Voice Service starts ingress playing consent + hold audio into the room, and rings agents.
3. Agent accepts → stop ingress → agent joins.
4. Timeout → `noAnswerAction`:
 - `AI`: dial OpenAI into the room (as in 3.1)
 - `VOICEMAIL`: play the prompt via ingress, record via egress (Phase 2)
 - `HANGUP`: delete the room → `MISSED`
5. `MISSED` → `call.missed` event → Textify sends the SMS/WhatsApp follow-up.

3.4 Inbound, AI_ONLY

Same as 3.1, but `transfer_to_human` isn't offered; the AI takes messages instead.

3.5 Outbound / click-to-call (agent ? customer)

Agent browser → Textify → POST `/v1/calls/outbound` {accountId, numberId, toE164, agentRef}

Voice Service:

1. GET can-call (direction=OUTBOUND)
2. create call + room out-{callId}
3. return agent token → browser joins (hears ringback)
4. `CreateSIPParticipant`(at-outbound, sip_call_to=toE164, sip_number=our number, room, wait_until_answered)

AT rings the customer → answer → customer joins → talk → room_finished → call.ended

3.6 Transfer to an external mobile (fallback, off by default)

- Preferred: `TransferSIPParticipant` (SIP REFER) of the caller to `tel:+255...`. **[VERIFY]** whether AT honours REFER.
- Alternative: `CreateSIPParticipant` dials the mobile into the same room (extra outbound leg, `externalTransferSeconds`).

3.7 Recording + summary pipeline

```

call answered → start RoomComposite egress (audio_only) →
s3://recordings/{accountId}/{callId}.ogg
room_finished → egress ends → webhook egress_ended → recordings row READY
    |
    ▼ (@Scheduled job picks pending)
    transcript: AI part already captured live from the OpenAI WebSocket;
    full call (incl. human agent part) → OpenAI transcription (sw/en)
    |
    ▼
    OpenAI chat (fixed prompt, JSON output) → summary
    |
    ▼
    transcripts row DONE → event summary.ready → Textify
daily retention job → delete objects + rows where delete_after < now()

```

Summary JSON:

```

{
  "summary": "Mteja aliuliza bei ya kifurushi cha mwezi na akaomba apigiwe kesho.",
  "intent": "pricing_inquiry",
  "resolved": false,
  "followUps": ["Call back tomorrow morning with monthly package price"],
  "sentiment": "neutral",
  "topic": "sales",
  "language": "sw"
}

```

(The example summary says: "The customer asked the price of the monthly package and asked to be called back tomorrow.")

3.8 Call state machine

```

RINGING → AI_DIALING → ANSWERED → COMPLETED
    |
    |
    |— (no balance) → REJECTED_NO_BALANCE
    |— (nobody handled / caller hung up) → MISSED
    |— (SIP/system error) → FAILED      (AI_DIALING/ANSWERED can also → FAILED)

```

`handled_by`: `NONE` | `AI` | `AGENT` | `AI_THEN_AGENT`. If the OpenAI leg fails to connect within 5 s: fall back to ringing agents (HUMAN_ONLY path) and log `AI_UNAVAILABLE`.

4. Data model (PostgreSQL)

```
client_apps (  
  id uuid pk, name text, token_hash text,  
  webhook_url text, webhook_secret text, can_call_url text,  
  created_at timestamptz  
)  
  
accounts (  
  id uuid pk, client_app_id uuid fk, external_ref text,          -- e.g. Textify tenant id  
  status text,                                                  -- ACTIVE | SUSPENDED  
  default_language text default 'sw',  
  recording_retention_days int default 90,  
  created_at timestamptz,  
  unique (client_app_id, external_ref)  
)  
  
phone_numbers (  
  id uuid pk, e164 text unique, provider text default 'AFRICAS_TALKING',  
  type text,                                                    -- REGULAR | TOLL_FREE  
  status text,                                                  -- UNASSIGNED | ACTIVE | RELEASED  
  account_id uuid fk null,  
  label text,                                                  -- "Sales line"  
  answer_mode text,                                             -- AI_FIRST | AI_ONLY | HUMAN_ONLY  
  greeting text,  
  ai_instructions text,                                         -- business info, FAQs, tone  
  ai_voice text null,                                           -- override default voice  
  recording_enabled boolean default true,  
  ring_group_id uuid fk null,  
  ring_timeout_sec int default 30,  
  no_answer_action text,                                         -- AI | VOICEMAIL | HANGUP  
  external_transfer_e164 text null,  
  updated_at timestamptz  
)  
  
agents (  
  id uuid pk, account_id uuid fk, external_ref text,          -- e.g. Textify user id  
  display_name text,
```

```

    status text,                                -- AVAILABLE | BUSY | OFFLINE (live presence also in
Redis)
    last_seen_at timestamptz,
    unique (account_id, external_ref)
)

ring_groups (id uuid pk, account_id uuid fk, name text, strategy text)  -- ALL_AT_ONCE |
ROUND_ROBIN
ring_group_members (ring_group_id uuid fk, agent_id uuid fk, position int,
                    primary key (ring_group_id, agent_id))

calls (
    id uuid pk, account_id uuid fk, number_id uuid fk,
    direction text,                                -- INBOUND | OUTBOUND
    from_e164 text, to_e164 text,
    livekit_room text, sip_call_id text,
    openai_call_id text null,
    status text, handled_by text, agent_id uuid null,
    started_at timestamptz, answered_at timestamptz, ended_at timestamptz,
    end_reason text,
    inbound_seconds int default 0, outbound_seconds int default 0,
    trunk_seconds int default 0, ai_seconds int default 0,
    external_transfer_seconds int default 0,
    message_taken jsonb null,                      -- {name, phone, reason}
    ai_usage jsonb null                            -- model, token/audio counters (cost measurement)
)

call_events (id bigserial pk, call_id uuid fk, type text, at timestamptz, data jsonb)  --
timeline

recordings (
    id uuid pk, call_id uuid fk unique, egress_id text, storage_key text,
    duration_sec int, size_bytes bigint,
    status text,                                -- RECORDING | READY | FAILED | DELETED
    delete_after timestamptz
)

transcripts (
    call_id uuid pk fk, status text,  -- PENDING | DONE | FAILED
    language text, text text, segments jsonb, summary jsonb, model text, created_at timestamptz

```

```

)

outbox_events (
  id uuid pk, client_app_id uuid fk, type text, payload jsonb,
  attempts int default 0, next_attempt_at timestampz, delivered_at timestampz null,
  last_error text
)

```

Redis keys:

- `presence:{accountId}:{agentRef}` → `AVAILABLE|BUSY` (TTL, refreshed by lobby webhooks)
- `call:{callId}:agent` → claim lock (SET NX, TTL)
- `call:{callId}:ai` → OpenAI call id + WebSocket owner node
- `rr:{ringGroupId}` → round-robin pointer

5. APIs

5.1 Client-facing API (Textify ? Voice Service)

Auth: `Authorization: Bearer <client_app_token>`. Every request is scoped to that client app's accounts. JSON. Base `https://voice-api.textify.africa/v1` **[VERIFY domain]**.

Method & path	Purpose
<code>POST /accounts</code>	Create/upsert account <code>{externalRef}</code>
<code>PATCH /accounts/{id}</code>	Status, language, retention days
<code>GET /numbers?status=UNASSIGNED</code>	Numbers available to assign
<code>POST /numbers/{id}/assign</code>	<code>{accountId, label}</code>
<code>POST /numbers/{id}/release</code>	Unassign
<code>PUT /numbers/{id}/config</code>	answerMode, greeting, aiInstructions, aiVoice, recordingEnabled, ringGroupId, ringTimeoutSec, noAnswerAction, externalTransferE164
<code>PUT /accounts/{id}/agents/{agentRef}</code>	Upsert agent <code>{displayName}</code>
<code>DELETE /accounts/{id}/agents/{agentRef}</code>	Remove agent
<code>PUT /accounts/{id}/agents/{agentRef}/status</code>	<code>AVAILABLE / BUSY / OFFLINE</code>
<code>POST /accounts/{id}/agents/{agentRef}/lobby-token</code>	LiveKit token for the agent's lobby
<code>POST/PUT/DELETE /accounts/{id}/ring-groups[/gid]</code>	Manage groups + members
<code>POST /calls/{id}/accept</code>	<code>{agentRef}</code> → <code>{roomToken, wsUrl}</code> or 409

Method & path	Purpose
POST /calls/{id}/hangup	End a call
POST /calls/{id}/transfer	{toAgentRef} or {toE164}
POST /calls/outbound	{accountId, numberId, toE164, agentRef} → {callId, roomToken, wsUrl}
GET /calls?accountId&from&to&status&page	Call log
GET /calls/{id}	Detail incl. timeline, message, summary
GET /calls/{id}/recording	Short-lived signed URL
GET /calls/{id}/transcript	Transcript + summary

Admin-only (QBIT ops token): POST /numbers (register a number bought from AT), POST /client-apps.

5.2 Voice Service ? Textify: balance check

GET {can_call_url}?accountRef=...&direction=INBOUND|OUTBOUND → {allowed: bool, maxMinutes?: int}.
Timeout 1.5 s → **allow** (fail-open), log it.

5.3 Webhooks: Voice Service ? Textify

- POST {webhook_url}; headers X-Voice-Event-Id, X-Voice-Event-Type, X-Voice-Timestamp, X-Voice-Signature: sha256=HMAC(secret, timestamp + "." + body).
- Delivered via the **outbox** with exponential backoff (1 m, 5 m, 30 m, 2 h, ... up to 24 h).
The receiver must be **idempotent** on event id.

Event	Payload highlights
call.started	callId, accountExternalRef, direction, from, to, numberId
call.answered	callId, handledBy (AI/AGENT), agentRef?
call.ended	callId, status, usage {inboundSeconds, outboundSeconds, trunkSeconds, aiSeconds, externalTransferSeconds, recorded}, endReason
call.missed	callId, from, to, numberLabel. Textify sends the SMS/WhatsApp follow-up
call.message_taken	callId, name, phone, reason
recording.ready	callId, durationSec
summary.ready	callId, summary JSON

Example call.ended:

```

{
  "id": "evt_01J...", "type": "call.ended", "occurredAt": "2026-10-21T12:30:05Z",
  "data": {
    "callId": "3f2c...", "accountExternalRef": "tenant_123",
    "direction": "INBOUND", "from": "+2557XXXXXXX", "to": "+2557YYYYYYYY",
    "status": "COMPLETED", "handledBy": "AI_THEN_AGENT", "agentRef": "user_88",
    "usage": { "inboundSeconds": 245, "outboundSeconds": 0, "trunkSeconds": 245,
      "aiSeconds": 70, "externalTransferSeconds": 0, "recorded": true },
    "endReason": "CALLER_HANGUP"
  }
}

```

5.4 Inbound webhooks to the Voice Service

POST /hooks/livekit: verified with LiveKit `WebhookReceiver`.

LiveKit event	Action
<code>participant_joined</code> (SIP, room <code>call-*</code>)	Start of an inbound call : number lookup → can-call → create call → dial AI or ring agents
<code>participant_joined</code> (identity <code>ai</code>)	AI leg up → <code>ANSWERED</code> , <code>handled_by</code> AI; start egress
<code>participant_left</code> (identity <code>ai</code>)	AI seconds end
<code>participant_joined</code> (agent identity)	<code>answered_at</code> / <code>handled_by</code> AGENT; start egress if not running
<code>participant_left</code> (SIP caller)	Caller hung up → delete room
<code>participant_joined/left</code> (<code>lobby-*</code>)	Agent presence
<code>room_finished</code>	Close call, compute durations, enqueue <code>call.ended</code> / <code>call.missed</code>
<code>egress_ended</code>	Recording READY/FAILED → enqueue transcription

POST /hooks/openai: verify OpenAI webhook signature, dedupe by event id.

OpenAI event	Action
<code>realtime.call.incoming</code> (or <code>live.transport.incoming</code>)	Match <code>X-Voice-Call-Id</code> header to a call in <code>AI_DIALING</code> → accept with session config → open control WebSocket. Unknown → reject

Durations are always computed from server-side webhook timestamps.

6. Infrastructure

6.1 Box (Phase 0/1: single dedicated VPS, separate from NexGate)

Docker Compose, **host networking** for media services (SIP/RTP/WebRTC need real ports).

Service	Image / runtime	Notes
caddy (or nginx)	reverse proxy	TLS for voice-api... (Spring Boot, incl. /hooks/*) and wss://voice-rtc... (LiveKit 7880)
livekit	livekit/livekit-server	bind 127.0.0.1:7880 behind proxy; keys via LIVEKIT_KEYS env
sip	livekit/sip	config via SIP_CONFIG_BODY (api_key, api_secret, ws_url, redis, sip_port 5060, rtp_port 10000-20000, use_external_ip); TLS enabled for the OpenAI trunk [VERIFY] SIP TLS config keys
egress	livekit/egress	audio-only recordings to S3; CPU-heavy
ingress	livekit/ingress	URL input for hold/consent/no-balance audio
redis	redis:7	password, localhost
postgres	postgres:16	Voice Service DB
voice-service	Spring Boot jar (Java 21)	public via proxy (API + webhooks)

Storage: private S3-compatible bucket (e.g. Linode Object Storage or MinIO). Signed URLs only. Audio prompt files (consent, hold, no-balance) in a separate bucket/path, readable by ingress.

Sizing (start, then measure): 4-8 vCPU, 8-16 GB RAM. Choose the region with the lowest RTT to Africa's Talking's SIP servers (measure in the PoC); OpenAI SIP is GeoIP-routed (sip.api.openai.com, EU: sip-eu.api.openai.com).

6.2 Ports & firewall

Port	Proto	Direction / open to	Purpose
22	TCP	in, admin IPs	SSH
80, 443	TCP	in, world	TLS issuance, API, webhooks, WSS

Port	Proto	Direction / open to	Purpose
7881	TCP	in, world	WebRTC TCP fallback (browser agents)
50000-60000	UDP	in, world	WebRTC media (browser agents)
5060	UDP+TCP	in, AT IPs only	SIP signalling from AT
10000-20000	UDP	AT IPs + OpenAI media CIDs only	SIP RTP media
5061	TCP	out, to <code>sip.api.openai.com</code>	SIP over TLS to OpenAI
7880, 6379, 5432, 8080	TCP	localhost only	LiveKit API, Redis, Postgres, Spring Boot

OpenAI SIP media CIDs (from OpenAI's SIP guide, Oct 2026; re-check before go-live):
`13.79.45.80/28, 23.98.140.64/28, 40.67.149.176/28, 40.83.204.240/28`. WebRTC range and SIP RTP range must not overlap.

6.3 Configuration (env)

```
LIVEKIT_URL=ws://127.0.0.1:7880
LIVEKIT_PUBLIC_WS_URL=wss://voice-rtc.textify.africa
LIVEKIT_API_KEY / LIVEKIT_API_SECRET
REDIS_URL / REDIS_PASSWORD
DB_URL / DB_USER / DB_PASSWORD
S3_ENDPOINT / S3_BUCKET / S3_ACCESS_KEY / S3_SECRET_KEY
OPENAI_API_KEY
OPENAI_WEBHOOK_SECRET
OPENAI_SIP_URI                # project SIP URI from OpenAI  [VERIFY format]
OPENAI_REALTIME_MODEL / OPENAI_DEFAULT_VOICE
AT_SIP_HOST / AT_SIP_USERNAME / AT_SIP_PASSWORD / AT_SIP_IPS
AUDIO_CONSENT_URL / AUDIO_HOLD_URL / AUDIO_NO_BALANCE_URL
```

Client app tokens, webhook secrets and can-call URLs live in `client_apps` (tokens hashed).

6.4 Bootstrap (one-time, idempotent on startup)

1. Inbound trunk `at-inbound` (numbers + AT IPs).
2. Dispatch rule (individual, `call-` prefix, no agent dispatch).
3. Outbound trunk `at-outbound` (AT host + credentials).
4. Outbound trunk `openai` (OpenAI SIP host, TLS, SRTP).
5. Configure the OpenAI project webhook → `https://voice-api.../hooks/openai`.

Store the ids. **Never create trunks/rules per call.**

7. Security

- SIP exposed **only** to AT IPs (inbound) and OpenAI media CIDRs (firewall **and** LiveKit `allowed_addresses`).
- OpenAI webhook signature verification + only accept OpenAI calls matching a call in `AI_DIALING`.
- Client app → Voice Service: bearer token (hashed at rest), scoped to its accounts.
- Voice Service → client app: HMAC-signed webhooks with timestamp (reject > 5 min old).
- LiveKit webhooks: signature verified.
- Browser gets only short-lived, room-scoped LiveKit tokens (~10 min to join).
- OpenAI API key and SIP credentials only on the server.
- Recordings: private bucket, signed URLs (~5 min), retention job, consent notice.
- Secrets in env only. Keep the CrowdSec/fail2ban practice from Textify's infra.

8. Observability

- Structured JSON logs with `callId`, `room`, `accountId`, `openaiCallId` on every line.
- Metrics: active calls, active AI sessions, calls/min, answer rate, missed rate, AI handover rate, AI dial-in time (CreateSIPParticipant → accept), AI first-response latency, egress failures, webhook backlog, can-call timeouts.
- Alerts: SIP service down, OpenAI dial-in failures > N%, zero inbound calls for X minutes in business hours, outbox backlog > N, egress failure rate > N%.

9. Testing strategy

- **PoC checklist**
 1. AT number → trunk → room created → `participant_joined` webhook received.
 2. Spring Boot dials OpenAI into the room → `call.incoming` webhook → accept → AI greets in Swahili → caller hears it.
 3. Tool call over the WebSocket (`end_call`) works → room deleted → durations correct.
 4. Handover: agent in browser joins, AI leg hangs up, caller stays.
- **Swahili quality test:** 10-20 real callers (Swahili, English, mixed). Score understanding and voice. If the voice is weak → Appendix A.
- **Cost:** log `ai_usage` per call; compute TZS per AI minute.
- **Unit:** routing (ring strategies, claim race), state machine, outbox retry, signature verification, OpenAI tool dispatch.

- Integration: LiveKit dev server + SIPp/softphone from an allowed IP; OpenAI in a test project; Textify webhook mock.

10. Repository layout (suggested)

```
textify-voice/  
├─ docs/                SPEC.md, ARCHITECTURE.md  
├─ voice-service/       Spring Boot (Java 21)  
├─ deploy/              docker-compose.yaml, livekit.yaml, sip, egress.yaml, ingress.yaml,  
Caddyfile, firewall.sh, .env.example  
├─ telephony/           trunk + dispatch-rule JSON templates  
└─ audio/               consent / hold / no-balance prompt files (sw + en)
```

11. Open items to verify before coding

1. AT trunk behaviour (SPEC §11): external SIP URI, outbound auth, IPs, billing, REFER, codecs.
2. **LiveKit SIP → OpenAI SIP** interop (TLS + SRTP, Opus/G.711), custom header pass-through, OpenAI project SIP URI format.
3. OpenAI Realtime API vs Live API choice; webhook event names; usage fields.
4. LiveKit JVM SDK method names (SIP participant create/transfer, egress, ingress, tokens).
5. LiveKit SIP participant attributes present in webhook payloads.
6. Ingress URL input for short audio prompts (latency to start playing).
7. OpenAI Swahili quality + per-minute cost; choose a voice.
8. Egress CPU usage for audio-only recordings on the chosen box.

Appendix A: Fallback if OpenAI SIP fails the PoC (Python worker)

Use only if (a) LiveKit $\nrightarrow$ OpenAI SIP can't interoperate, or (b) OpenAI's Swahili voice isn't good enough and we need a different TTS provider.

- Add a small Python service using LiveKit Agents (`livekit-agents[openai]`), dispatched into each call room by the dispatch rule (`room_config.agents`).
- It runs the voice loop (OpenAI Realtime, or STT + LLM + another Swahili TTS) and exposes the same three tools, which call the Voice Service over a localhost-only internal API (`/internal/calls/start | handoff | message | end | ai-usage`).

- No database, no business logic; about 150–300 lines. Spring Boot stays the brain; data model, client APIs and events above are unchanged.

Textify Voice: Product & Functional Specification

“ Status: draft for implementation · Owner: Joshua Saimon Sakweli (QBIT SPARK)
Companion doc: `ARCHITECTURE.md` (components, data model, APIs, infra). This file is written so a new engineer, or a new AI coding session, can pick up the work with no prior context.

1. Background

Textify Africa (textify.africa) is an existing bulk SMS + WhatsApp platform, **written in Go**, live in Tanzania, Uganda and Nigeria. Its main model is **white-label**: agencies and resellers ("partners") brand the platform as their own, set their own prices, and onboard their own customers. Customers top up a **prepaid wallet** with mobile money or card.

We are adding a third channel: a **call center / business phone line** product, similar in spirit to SemaCall (semacall.com): *"One business. One number. Every team."* Customers call one number; the call reaches the right person (or an AI answers) in the customer's language.

The voice capability is built as a **separate service ("Voice Service") in Java Spring Boot**. Textify consumes it as a client. The Voice Service must not know about Textify's users, auth or money.

2. Goals

1. A Textify customer can get a Tanzanian phone number and have every incoming call answered, by an **AI receptionist** (Swahili first, English supported) or by **human agents in the browser**.
2. AI can **hand the caller over to a human** when needed.
3. Every call is **logged, optionally recorded, transcribed and summarized**.
4. **Missed calls trigger an automatic SMS/WhatsApp** follow-up (Textify already does messaging).
5. Usage is **billed per minute from the existing Textify wallet**.

6. The Voice Service is **reusable** by other QBIT SPARK products later (JikoXpress, Mauzodukani, NexGate) without changes.

3. Non-goals (for now)

- No video calls over phone lines (LiveKit SIP does not support video over SIP anyway).
- No user accounts, roles, plans or pricing inside the Voice Service.
- No approval workflows anywhere. Keep everything self-service and simple.
- Countries other than Tanzania (Uganda/Nigeria later, using the same design).
- Building our own telco connection. We use Africa's Talking for numbers and the PSTN.

4. Key terms (plain language)

Term	Meaning
PSTN / telco network	The normal phone network (Vodacom, Airtel, Yas, Halotel, TTCL).
SIP	The protocol internet phones use to set up calls (ring, answer, hang up). Like HTTP but for calls. Voice audio travels separately over RTP .
SIP account	One login for one phone/softphone (like an email account but for calls).
SIP trunk	A connection carrying many calls at once between two systems over SIP. Here: Africa's Talking ≠ our LiveKit .
Human agent	A staff member who answers calls (in the browser app).
AI agent	A bot that talks on the phone: listens → thinks → speaks.
STT / LLM / TTS	Speech-to-text, the language model (the "brain"), and text-to-speech (the voice).
Realtime / speech-to-speech model	One model that hears and speaks directly (e.g. OpenAI Realtime).
LiveKit room	A live audio session. Each phone call = one room. The caller joins as a "SIP participant"; AI or humans join the same room.
Dispatch rule	LiveKit config that decides which room an incoming call goes to and which AI agent joins.
DTMF	Keypad presses during a call ("press 1").
Cold / warm transfer	Cold: just forward the call. Warm: AI briefs the human first, then hands over.

5. Providers we use and why

Provider	Role	Why
Africa's Talking (AT)	Tanzanian phone numbers + connection to all TZ telcos + SIP trunk	Local numbers and licensing. LiveKit's own phone numbers are US-only.
LiveKit (self-hosted, open source)	Media server: rooms, SIP, browser audio, recording, transfers	Already used in production by us for NexGate (video, spaces, audio calls). Self-hosted = no per-minute LiveKit fee.
OpenAI	AI brain + voice (Realtime model), transcription, summaries	Best overall quality. Swahili quality must be tested; TTS can be swapped.

6. Costs (Africa's Talking Tanzania, as published on africastalking.com/pricing, Oct 2026)

Item	Price
Regular number (Airtel)	Setup TZS 100,000 + 18% VAT; TZS 35,000/month + 18% VAT
Toll-free number	Same as regular, plus TZS 170,000 deposit
Regular incoming	TZS 17 / min
Regular outgoing	TZS 60 / min
Toll-free incoming	TZS 200 / min
SIP agent leg	TZS 4.30 / min
SIP trunk	TZS 8.61 / min
Conference	TZS 8.61 / min
TTS (Google, via AT)	per character

6.1 How the costs combine

Every call has **leg 1** (customer ⇌ our AT number) and, if someone answers, **leg 2** (our number ⇌ whoever answers).

Customer —leg 1—> AT number —leg 2—> answerer

17 in / 60 out SIP trunk to LiveKit: 8.61

forward to a mobile: 60 (avoid)

Scenario	Approx. cost per minute
Inbound call into LiveKit (AI or browser agent)	17 + 8.61 $\approx$ TZS 25.61 + AI cost if AI answers
Outbound call from LiveKit to a customer	60 + 8.61 $\approx$ TZS 68.61 (to confirm)
Inbound forwarded to a staff mobile (fallback only)	17 + 60 $\approx$ TZS 77

Key facts already agreed:

- **Cost follows call minutes, not the number of agents.** Idle agents cost nothing, so customers can add unlimited agents.
- **Avoid forwarding to mobiles.** Agents answer in the browser/app.
- **Design decision:** money is not the constraint, so **all calls go through the SIP trunk into LiveKit** (one call path, simpler code). We don't split IVR onto AT XML to save the 8.61.

6.2 AI cost (on top of telco)

An AI-answered minute adds OpenAI cost (Realtime, or STT + LLM + TTS). This is several times the telco cost per minute. **Exact per-minute figures must be measured during the PoC** from real calls (token/audio usage logged per call).

6.3 Indicative selling prices (suggestion, to be finalized after PoC)

Item	Suggested price
Number rental	TZS 60-75k / month
Inbound minute, human-answered	TZS 40-50 / min
Outbound minute	TZS 90-100 / min
AI-answered minute	TZS 150-250 / min
AI summary add-on	+ TZS X / min (set after measuring)

Pricing lives **in Textify only**. The Voice Service only reports usage.

7. Users & personas

- **Business owner / admin** (Textify customer): gets a number, sets the AI greeting and business info, adds agents, sees calls, recordings and summaries.

- **Human agent:** logs into the Textify web app, sets themselves available, answers calls in the browser, makes click-to-call outbound calls.
- **Caller:** a member of the public calling the business number from any TZ network.
- **Partner (reseller):** later phase. Resells voice under their own brand (same white-label model as SMS).

8. Functional requirements

8.1 Numbers

- FR-N1: Textify admin (QBIT/Textify ops) obtains numbers from AT and registers them in the Voice Service.
- FR-N2: A number is assigned to exactly one **account** (Textify's tenant id, opaque to the Voice Service).
- FR-N3: Each number has an **answer mode**: `AI_FIRST` (AI answers, can hand over), `HUMAN_ONLY` (ring agents), `AI_ONLY`.
- FR-N4: Reassigning or releasing a number is a single API call. No approval.

8.2 Inbound calls

- FR-I1: Caller dials the number → call arrives over the SIP trunk → LiveKit creates a room → Voice Service records a call `RINGING`.
- FR-I2: Before answering, the Voice Service asks Textify "**can this account take a call?**" (prepaid balance). If Textify is unreachable, **allow the call** (fail-open) and bill later.
- FR-I3: If the account can't pay: play a short polite message and end the call (no AI cost).
- FR-I4: Play a **recording consent notice** when recording is on ("Simu hii inaweza kurekodiwa kwa ajili ya ubora wa huduma"; in English: "This call may be recorded for quality of service").
- FR-I5: `AI_FIRST` / `AI_ONLY` → AI receptionist joins and greets with the account's greeting.
- FR-I6: `HUMAN_ONLY` → ring the account's available agents (ring group). If nobody answers within N seconds (default 30) → voicemail **or** AI fallback (configurable) → mark `MISSED` if nobody handled it.
- FR-I7: Caller keypresses (DTMF) are supported for simple menus ("press 1 for sales").

8.3 AI receptionist

- FR-A1: Speaks **Swahili first**, switches to English if the caller does. Handles code-switching (Swahili and English mixed).
- FR-A2: Uses the account's **instructions** (business info, FAQs, hours, tone) supplied by Textify.
- FR-A3: Short, spoken-style answers. Never invents prices or policies. If unsure → offer a human.

- FR-A4: Tools the AI can use (v1): `transfer_to_human`, `take_message`, `end_call`. Later: account-defined tools (check order, book appointment) via webhooks to the client app.
- FR-A5: Handover: AI calls `transfer_to_human` → Voice Service rings available agents → first to accept joins the room → AI briefly introduces the context (warm transfer), then leaves.
- FR-A6: If no agent is available: AI takes a message (name, number, reason) → stored on the call → event to Textify.
- FR-A7: Barge-in/interruptions and turn detection are handled by LiveKit Agents.

8.4 Human agents (browser)

- FR-H1: Agents are identified only by an opaque `agentRef` from Textify. Textify owns the user, login and role.
- FR-H2: Agent sets status `AVAILABLE` / `BUSY` / `OFFLINE` through Textify, which forwards it to the Voice Service.
- FR-H3: Incoming call → available agents in the group get a **ring** in the browser (data message over LiveKit). Accept → join the call room. Reject or timeout → next.
- FR-H4: Ring strategy: `ALL_AT_ONCE` (default) or `ROUND_ROBIN`.
- FR-H5: Agent can mute, hold, hang up, and transfer to another agent.
- FR-H6: **Click-to-call** outbound from the dashboard: agent clicks a number → Voice Service dials out via the trunk → agent and customer in one room.
- FR-H7: Fallback transfer to an external mobile is allowed (costs about TZS 60/min). Off by default.

8.5 Recording, transcription, summary

- FR-R1: Recording on/off per number (default **on**, with consent notice).
- FR-R2: After the call: audio file stored in our object storage (never rely on provider links).
- FR-R3: Transcription (Swahili/English) → **summary** containing:
 - 2-3 line summary
 - customer intent, resolved yes/no
 - follow-up actions
 - sentiment (positive / neutral / negative)
 - topic tag (sales, support, complaint, inquiry, other)
- FR-R4: Transcript + summary attached to the call record; event `summary.ready` sent to the client app.
- FR-R5: Retention: recordings deleted after **90 days** by default (configurable per account). Tanzania Personal Data Protection Act applies.
- FR-R6: Summary is a billable add-on (Textify decides price; Voice Service reports `summaryMinutes`).

8.6 Missed calls & follow-ups

- FR-M1: A call is `MISSED` if no human or AI handled it (e.g. HUMAN_ONLY with nobody answering and no fallback, or the caller hung up while ringing).
- FR-M2: Voice Service emits `call.missed` with the caller number. **Textify** sends the SMS/WhatsApp follow-up ("Samahani tumekukosa, tutakupigia tena"; in English: "Sorry we missed you, we'll call you back") using its existing messaging.

8.7 Usage & billing

- FR-B1: On every call end, the Voice Service emits `call.ended` with usage broken down by leg (seconds): `inboundSeconds`, `outboundSeconds`, `trunkSeconds`, `aiSeconds`, `externalTransferSeconds`, `recorded` (bool), `summarySeconds`.
- FR-B2: Textify prices and deducts from the wallet. The Voice Service never deals with money.
- FR-B3: Events are **idempotent** (event id) and retried until Textify returns 2xx.

8.8 Dashboard (built in Textify, data from the Voice Service)

- Call log: time, direction, caller, number, handled by (AI / agent), duration, status, cost (Textify-computed).
- Call detail: audio player, transcript, summary, AI message taken.
- Number settings: answer mode, greeting, AI instructions, recording on/off, ring group, timeout, fallback.
- Agents: availability, calls handled.
- Simple stats: calls per day, answered vs missed, AI-handled vs human, average duration.

9. Non-functional requirements

Area	Requirement
Latency	AI response start < ~1 s after the caller stops speaking (target). Call connect < 3 s.
Availability	Voice Service must keep taking calls if Textify is down (fail-open balance check, queued webhooks).
Isolation	Dedicated LiveKit box for Textify Voice. Not shared with NexGate.
Security	SIP/RTP ports open only to Africa's Talking IPs. Service tokens between apps. Signed webhooks. No user PII beyond phone numbers.
Privacy	Consent notice, retention policy, recordings in our storage, access only via the client app.

Area	Requirement
Language	Swahili first-class. Test quality with real Tanzanian callers before selling AI.
Simplicity	No approval workflows, no internal OAuth, minimal moving parts.

10. Phases

Phase 0: Proof of concept (goal: one real call end-to-end)

1. Confirm with AT (see §11). Get one TZ number + SIP trunk.
2. Dedicated box: LiveKit server + LiveKit SIP + Redis. Firewall restricted to AT.
3. Inbound trunk + dispatch rule → Spring Boot dials OpenAI Realtime into the room over SIP → AI greets in Swahili.
4. Measure: latency, audio quality, OpenAI cost per minute.
5. **Swahili test**: 10-20 real people (Swahili, English, mixed). Judge understanding and voice. Swap TTS if the voice is weak.

Phase 1: MVP (sellable)

- Spring Boot Voice Service: accounts, numbers, config, calls, events, LiveKit integration.
- AI receptionist with handover to browser agents.
- Browser agent client (in Textify's web app) + ring groups + click-to-call.
- Recording + storage + retention.
- Balance check + `call.ended` usage events → wallet deduction in Textify.
- Missed-call SMS/WhatsApp.

Phase 2

- Transcripts + AI summaries (if not already in MVP).
- Account-defined AI tools (webhooks to the client app: check order, booking).
- Voicemail, business hours / after-hours behaviour.
- Stats dashboard.

Phase 3

- **Voice broadcast** (recorded message to a contact list: "bulk voice", fits Textify's bulk DNA). Check TZ calls-per-second limits.

- **AI outbound campaigns** (reminders that talk back).
- **White-label voice for partners.**
- Other countries (Uganda/Nigeria) through AT.
- Other QBIT products as clients of the Voice Service.

11. Open questions for Africa's Talking (must answer before Phase 1)

1. Can the TZ SIP trunk **route inbound calls for our number to an external SIP URI** (our LiveKit SIP endpoint)? Note: LiveKit SIP does **not support SIP REGISTER**, so AT must send calls to us directly.
2. Can it **accept outbound calls from our server IP** (and what auth: IP allowlist or username/password)?
3. What are **AT's SIP signalling and media IP addresses** (for our firewall allowlist)?
4. Exact billing for **inbound to trunk**: 17 + 8.61? Any extra?
5. Exact billing for **outbound via trunk**: 60 + 8.61? Any extra?
6. Is the **SIP agent TZS 4.30** charged on inbound calls too? Any setup/monthly fee for SIP agents?
7. Are **unanswered rings** to SIP endpoints billed? 7b. The listed regular number is **Airtel**. How do callers on Vodacom, Yas, Halotel and TTCL reach it, and at what cost to them and to us?
8. **Calls-per-second (CPS)** limits and pricing in Tanzania (needed for broadcast).
9. Supported **codecs** (G.711 PCMU/PCMA expected) and DTMF method (RFC 2833/4733).
10. Does reselling voice to our customers need any **TCRA licence** on our side, or is AT's licence enough? Get it in writing.
11. Number porting / getting several numbers quickly; toll-free availability.

AT is **not** on LiveKit's list of tested SIP providers (Twilio, Telnyx, Exotel, Plivo, Wavix, Sinch, didlogic). LiveKit is designed to work with any SIP provider, so the PoC must prove compatibility.

12. Risks

Risk	Mitigation
AT trunk can't deliver to an external SIP URI	Verify first (\$11 Q1). Fallback: AT Voice XML + AT SIP agents for humans; AI limited.
Swahili STT/TTS quality poor	Early real-user test; swap TTS provider; keep human fallback prominent.
AI cost higher than expected	Measure in PoC; short answers; pre-recorded greetings; cheap model for simple turns; price AI minutes higher.

Risk	Mitigation
SIP fraud/attacks	Firewall to AT IPs only; LiveKit inbound trunk <code>allowed_addresses</code> + numbers.
Textify outage blocks calls	Fail-open balance check; queued + retried webhooks.
Regulatory (TCRA, PDPA)	Written confirmation from AT; consent notice; retention policy.
Single box failure	Phase 1 single box is OK; add a second node (LiveKit supports multi-node via Redis) when volume grows.

13. Decisions log

#	Decision
D1	Numbers and telco via Africa's Talking ; media via self-hosted LiveKit ; AI via OpenAI .
D2	All calls go through the SIP trunk into LiveKit (no split IVR on AT XML), for one call path.
D3	Agents answer in the browser (no forwarding to mobiles by default).
D4	Dedicated LiveKit box for Textify Voice, separate from NexGate's LiveKit.
D5	Separate Voice Service in Java Spring Boot . Knows no users/auth/money; owns numbers, routing, calls, recordings, usage events.
D6	Textify (Go) owns users, tenants, partners, wallet, pricing, dashboard, messaging follow-ups.
D7	Preferred: no Python. OpenAI Realtime joins the LiveKit room as a SIP participant, controlled from Spring Boot via OpenAI's call-control REST + WebSocket (ARCHITECTURE §2.5). Fallback: thin Python LiveKit Agents worker (§2.2) if OpenAI SIP interop or Swahili voice quality fails the PoC.
D8	Billing is per minute from the Textify wallet; Voice Service emits usage events only.
D9	Recording default on with consent notice; 90-day retention default.
D10	No approval workflows anywhere.