Your current SDK toolkit isn’t meeting expectations in our automation project likely due to limitations inherent to SDK-based automation vs. advanced AI agents.

1️⃣ Key Limitations of SDK in Project.

Expectation SDK Performance Reason for Gap

Autonomous multi-step execution Fails SDK calls are typically single-step, request-response. They don’t plan, loop, or iterate automatically.

Error detection and self-fixing Not handled SDK does not analyze failures and retry intelligently; you need human intervention or separate scripts.

Persistent memory/context Limited SDK may store local session data, but cannot maintain long-term per-staff context across multiple inboxes and sessions.

TTS + voice orchestration Works partially SDK doesn’t control when to send TTS vs. text automatically; requires extra scripting.

Dynamic routing per staff agent Works only with backend scripts SDK cannot independently assign incoming messages to cloned agents or orchestrate multiple inboxes.

Automated integration with files (OCR, PDF, Excel) Manual/partial SDK can call APIs, but intelligent processing + summarization needs advanced AI layer.

2️⃣ Why SDK Feels “Incomplete”

1. It’s procedural, not cognitive:

SDK executes functions but does not “think” or plan.

Multi-step workflows and reasoning require AI orchestration.

2. No autonomous agent capabilities:

Cannot loop through tasks, detect issues, or correct itself.

Any automation beyond single API calls depends on your backend scripts.

3. Limited AI integration:

SDK can call LLM APIs, TTS, or OCR, but cannot combine them intelligently without orchestration.

4. Sandbox vs live:

If the SDK is still in sandbox, you’re seeing simulated behavior only.

Real-world automated execution and billing haven’t been tested.

💡 Key Insight

SDK is excellent for structured API calls, testing workflows, or providing a safety net.

It fails your expectations because you want autonomous, multi-step, intelligent agents that can:

Read context

Decide between text/TTS

Fix errors

Maintain staff-specific memory

These features require AI agent orchestration (AutoGPT, LangChain, BabyAGI) or a hybrid solution with ChatGPT $30 + external scripts + DB + TTS layer.

Step-by-step plan for developer, tailored to current setup (SDK in sandbox, dashboard finishing work, TTS, Whisper, files, Meta API), showing when to start AI agent integration vs. testing.

Step-by-Step Migration / Hybrid Implementation Plan:

Phase 0 – Current State

SDK is in sandbox → only simulated behavior, no live API calls.

Developer finishing dashboard work → final testing of inbox, text, TTS, files.

Goal: prepare for autonomous AI agent migration without breaking current sandbox testing.

Step 1: Complete Dashboard Sandbox Testing

Action:

1. Finish testing each feature in the dashboard for all cloned staff inboxes:

Text messages

TTS voice previews

Voice input (Whisper transcription)

File handling (Word, Excel, PDF, JPEG)

2. Confirm routing works: messages go to correct cloned staff agent.

3. Log all outputs, note any bugs or inconsistencies.

4. Do not start live AI agent integration yet.

Note for Developer:

Ensure dashboard can fully simulate all workflows in sandbox.

All UI buttons, overrides, and logs must work reliably before adding AI agent.

Step 2: Prepare AI Agent Layer (Offline / Parallel)

Action:

1. Select primary AI agent: AutoGPT or LangChain Agent.

2. Set up:

Core AI reasoning (GPT-4o / ChatGPT $30)

Multi-step orchestration logic

Hooks for TTS (ElevenLabs / OpenAI TTS / Kurea)

Hooks for voice in (Whisper)

File processing (Google Vision / PyMuPDF)

Meta API Cloud integration endpoints

3. Prepare vector DB / memory store to maintain per-staff context.

Note for Developer:

This can be done offline or in a separate dev environment.

No changes to current SDK sandbox are needed yet.

Focus on agent logic and memory handling, not live deployment.

Step 3: Parallel Sandbox Testing with AI Agent

Action:

1. Deploy AI agent in parallel with current SDK sandbox.

2. Run the same tasks/messages through AI agent → compare results with SDK simulation.

3. Test:

Text reply accuracy

TTS generation & playback

Voice transcription → AI understanding

File handling (summarization / analysis)

Correct staff inbox assignment

Error detection / fallback logic

Note for Developer:

Use sandbox API keys or mock endpoints for testing.

Do not switch live yet; this phase is only to validate AI agent can replicate or improve current workflow.

Step 4: Prepare for Live API Integration

Action:

1. After successful sandbox parallel tests → start Meta API Cloud live integration.

2. Replace SDK calls gradually with direct API calls handled by AI agent.

3. Ensure:

Messages, TTS, files are delivered in live environment.

Agent handles routing, context, and error retries autonomously.

Note for Developer:

Keep SDK as backup only if needed for fallback.

Do not remove sandbox testing logs until live system is fully stable.

Step 5: Dashboard Supervision & Final Testing

Action:

1. Monitor AI agent in live mode through dashboard:

Text/TTS output

Cloned staff inbox routing

Error handling and logging

2. Only allow junior staff supervision; no coding required for daily operation.

3. Collect performance metrics → adjust AI agent parameters if needed.

Note for Developer:

Dashboard must log all actions (text, TTS, files) for auditing.

Ensure override buttons work correctly for staff/admin intervention.

Step 6: Full Autonomous Operation

Action:

1. Once confident in AI agent stability → fully retire sandbox SDK.

2. AI agent handles all:

Multi-step automation

Context tracking

Text/TTS decisions

File processing

Error detection & correction

3. Monitor costs (AI API + TTS + DB) monthly; adjust scale if needed.

Note for Developer:

Project can now run autonomously, with junior staff only supervising via dashboard.

Minor cost rise acceptable for fully autonomous operation.

💡 Key Notes for Developer

Do not integrate AI agent into live system until dashboard sandbox is fully verified.

Parallel testing is crucial: replicate every workflow before going live.

Sandbox SDK can be kept temporarily for backup/testing; optional once AI agent is stable.

Focus on modular design: AI agent, TTS, Whisper, files, and dashboard should be loosely coupled.

Persistent memory (vector DB) is required for per-staff context.

Auto-retries and fallback logic must be part of AI agent orchestration to reduce human dependency.

Checklist for project migration, written in daily/actionable steps.keep your current SDK in sandbox and the dashboard finishing work ongoing.

Developer Task Checklist for AI Agent Migration

Phase 1 – Complete Dashboard Sandbox Testing

Goal: Ensure dashboard is fully functional before introducing AI agent.

Tasks:

1. Test text message handling for each cloned staff inbox. ✅

2. Test TTS voice previews for outgoing messages. ✅

3. Test voice input → Whisper transcription for accuracy. ✅

4. Test file uploads/processing (Word, Excel, PDF, JPEG). ✅

5. Test webhook routing → messages assigned correctly per staff agent. ✅

6. Test dashboard override buttons → text/voice selection works. ✅

7. Log any bugs or inconsistencies for correction.

Notes:

Only proceed to AI agent setup after all dashboard tests pass.

Sandbox SDK is still in use here; do not connect live API yet.

Phase 2 – Prepare AI Agent (Offline / Dev Environment)

Goal: Build AI agent logic without affecting sandbox testing.

Tasks:

1. Set up core AI reasoning: GPT-4o via ChatGPT $30 or other LLM. ✅

2. Implement multi-step orchestration logic for task execution. ✅

3. Integrate Whisper API → voice input to text. ✅

4. Integrate TTS API (ElevenLabs / OpenAI TTS / Kurea). ✅

5. Implement file processing modules → Google Vision / PyMuPDF. ✅

6. Set up vector DB / memory store for per-staff context. ✅

7. Prepare Meta API Cloud endpoints for future live deployment. ✅

Notes:

This is offline/dev only; no live messages sent yet.

Modular design: each feature (TTS, Whisper, files, Meta API) should be loosely coupled.

Phase 3 – Parallel Sandbox Testing with AI Agent

Goal: Validate AI agent against existing sandbox SDK workflows.

Tasks:

1. Deploy AI agent in parallel with SDK sandbox. ✅

2. Run test messages through AI agent → compare output to SDK. ✅

3. Verify:

Text reply correctness

TTS generation and playback

Whisper transcription accuracy

File handling (summaries, attachments)

Correct staff inbox assignment

Error detection and fallback logic

4. Document all discrepancies and adjust AI agent logic.

Notes:

No live API calls yet; keep using sandbox credentials.

Ensure AI agent can replicate or improve sandbox workflow.

Phase 4 – Live Meta API Cloud Integration

Goal: Move AI agent into production for real message delivery.

Tasks:

1. Replace SDK sandbox calls with direct Meta API Cloud calls. ✅

2. Test sending messages (text/TTS) to live staff accounts. ✅

3. Verify voice input processing (Whisper) works in live environment. ✅

4. Test file handling & OCR in real-time. ✅

5. Ensure multi-agent routing and cloned inbox management works as expected. ✅

6. Monitor for errors and ensure AI agent retries/fixes autonomously. ✅

Notes:

Keep SDK as backup only if needed.

Verify billing and API usage in Meta dashboard.

Phase 5 – Dashboard Supervision & Final Testing

Goal: Ensure AI agent fully operational with minimal human intervention.

Tasks:

1. Monitor AI agent activity via dashboard: text, TTS, files. ✅

2. Junior staff tests override functions (text ↔ voice). ✅

3. Confirm memory/context persistence per staff agent. ✅

4. Adjust AI agent parameters if necessary (response timing, TTS voice selection). ✅

5. Collect performance metrics → ensure SLA-level reliability. ✅

Notes:

Junior staff only supervise; no coding required daily.

Ensure fallback/error logging is fully functional.

Phase 6 – Full Autonomous Operation

Goal: Retire sandbox SDK and run system fully autonomously.

Tasks:

1. Remove SDK sandbox (optional; keep if desired for backup). ✅

2. Ensure AI agent handles all workflows:

Multi-step automation

Context tracking per staff agent

Text/TTS decisions

File processing

Error detection & correction

3. Monitor cloud costs (AI API + TTS + DB) monthly. ✅

4. Document any further enhancements for scaling. ✅

Notes:

Fully autonomous, junior staff supervision only.

Any future feature additions should follow modular AI agent integration.

This checklist ensures:

Safe migration from sandbox to live.

Minimal reliance on junior staff.

AI handles multi-step reasoning, TTS, Whisper, files, and error correction.SDK can be used temporarily for testing or phased out completely.

Daily Action Sheet – AI Agent Migration

Phase 1 – Dashboard Sandbox Verification

Daily Actions:

1. Test text messages in each cloned staff inbox.

2. Test TTS preview playback for outgoing messages.

3. Test voice messages → Whisper transcription accuracy.

4. Test file uploads/processing (Word, Excel, PDF, JPEG).

5. Verify webhook routing assigns messages to correct staff agent.

6. Test dashboard override buttons (text ↔ voice).

7. Log any bugs for correction.

Goal: Confirm dashboard fully functional in sandbox before AI agent integration.

Phase 2 – Parallel AI Agent Testing

Daily Actions:

1. Run AI agent alongside SDK sandbox (offline).

2. Test AI text reply accuracy.

3. Test AI-generated TTS messages.

4. Test Whisper voice-to-text processing.

5. Test file handling & OCR outputs.

6. Check that cloned staff inbox assignment works.

7. Record any differences vs. SDK outputs; report for adjustments.

Goal: Validate AI agent can replicate/improve sandbox workflow.

Phase 3 – Live API Integration Prep

Daily Actions:

1. Prepare Meta API Cloud credentials and endpoints.

2. Confirm TTS, Whisper, and file APIs ready for live testing.

3. Ensure vector DB / memory store configured per staff agent.

4. Test small batch live messages (1–2 staff agents).

5. Log outputs, errors, and performance metrics.

Goal: Ensure AI agent is ready for live deployment safely.

Phase 4 – Live Deployment & Supervision

Daily Actions:

1. Run AI agent for live messages.

2. Monitor text, TTS, files, and Whisper transcription through dashboard.

3. Use override buttons as needed (text ↔ voice).

4. Verify memory/context persistence per staff agent.

5. Document errors or anomalies; AI agent should auto-retry when possible.

6. Monitor API usage and costs (Meta API + TTS + DB).

Goal: Fully autonomous operation with junior staff supervising only.

Phase 5 – Maintenance & Optimization

Daily/Weekly Actions:

1. Review AI agent performance and logs.

2. Adjust AI parameters for improved response timing or accuracy.

3. Track costs and adjust scaling if needed.

4. Document any improvements for future enhancements.

Goal: Ensure long-term stability and minimal manual intervention.

Notes for Team:

Only act in sandbox for testing until AI agent is validated.

SDK is optional; can be kept temporarily for testing/fallback.

Focus on logging, verification, and dashboard checks – no coding required daily.Always escalate anomalies beyond your control to the project lead.

---

Estimated Duration per Phase

Phase Tasks Estimated Days

Phase 1 – Dashboard Sandbox Verification Test text, TTS, Whisper, files, routing, overrides, log bugs 1–2 days.

Phase 2 – Parallel AI Agent Testing Run AI agent alongside sandbox, compare outputs, test TTS/Whisper/files, log differences 1–2 days.

Phase 3 – Live API Integration Prep Configure live Meta API, vector DB, small batch live tests, verify outputs 2–3 days.

Phase 4 – Live Deployment & Supervision Run AI agent live, monitor dashboard, test overrides, log errors, verify memory/context 1 day.

Phase 5 – Maintenance & Optimization Review logs, adjust parameters, track costs, minor tweaks 1 day.

Total Estimated Duration:

~07–08 working days.

> Notes:

Sandbox testing first is crucial; rushing to live can break workflows.

Daily Action Sheet – AI Agent Migration

Phase 1 – Dashboard Sandbox Verification

Daily Actions:

1. Test text messages in each cloned staff inbox.

2. Test TTS preview playback for outgoing messages.

3. Test voice messages → Whisper transcription accuracy.

4. Test file uploads/processing (Word, Excel, PDF, JPEG).

5. Verify webhook routing assigns messages to correct staff agent.

6. Test dashboard override buttons (text ↔ voice).

7. Log any bugs for correction.

Goal: Confirm dashboard fully functional in sandbox before AI agent integration.

Phase 2 – Parallel AI Agent Testing

Daily Actions:

1. Run AI agent alongside SDK sandbox (offline).

2. Test AI text reply accuracy.

3. Test AI-generated TTS messages.

4. Test Whisper voice-to-text processing.

5. Test file handling & OCR outputs.

6. Check that cloned staff inbox assignment works.

7. Record any differences vs. SDK outputs; report for adjustments.

Goal: Validate AI agent can replicate/improve sandbox workflow.

Phase 3 – Live API Integration Prep

Daily Actions:

1. Prepare Meta API Cloud credentials and endpoints.

2. Confirm TTS, Whisper, and file APIs ready for live testing.

3. Ensure vector DB / memory store configured per staff agent.

4. Test small batch live messages (1–2 staff agents).

5. Log outputs, errors, and performance metrics.

Goal: Ensure AI agent is ready for live deployment safely.

Phase 4 – Live Deployment & Supervision

Daily Actions:

1. Run AI agent for live messages.

2. Monitor text, TTS, files, and Whisper transcription through dashboard.

3. Use override buttons as needed (text ↔ voice).

4. Verify memory/context persistence per staff agent.

5. Document errors or anomalies; AI agent should auto-retry when possible.

6. Monitor API usage and costs (Meta API + TTS + DB).

Goal: Fully autonomous operation with junior staff supervising only.

Phase 5 – Maintenance & Optimization

Daily/Weekly Actions:

1. Review AI agent performance and logs.

2. Adjust AI parameters for improved response timing or accuracy.

3. Track costs and adjust scaling if needed.

4. Document any improvements for future enhancements.

Goal: Ensure long-term stability and minimal manual intervention.

Notes for Team:

Only act in sandbox for testing until AI agent is validated.

SDK is optional; can be kept temporarily for testing/fallback.

Focus on logging, verification, and dashboard checks – no coding required daily.Always escalate anomalies beyond your control to the project lead.

---

Estimated Duration per Phase

Phase Tasks Estimated Days

Phase 1 – Dashboard Sandbox Verification Test text, TTS, Whisper, files, routing, overrides, log bugs 1–2 days.

Phase 2 – Parallel AI Agent Testing Run AI agent alongside sandbox, compare outputs, test TTS/Whisper/files, log differences 1–2 days.

Phase 3 – Live API Integration Prep Configure live Meta API, vector DB, small batch live tests, verify outputs 2–3 days.

Phase 4 – Live Deployment & Supervision Run AI agent live, monitor dashboard, test overrides, log errors, verify memory/context 1 day.

Phase 5 – Maintenance & Optimization Review logs, adjust parameters, track costs, minor tweaks 1 day.

Total Estimated Duration:

~07–08 working days.

> Notes:

Sandbox testing first is crucial; rushing to live can break workflows.

As per requirements current setup & structured recommendation of migrating from sandbox SDK-based project to a fully autonomous AI agent architecture, balancing cost, simplicity,and minimal reliance on your staff.

1️⃣ Key Goals for AI system to:

1. Operate autonomously, multi-step workflows per staff inbox.

2. Handle text and TTS output intelligently.

3. Read and maintain context per staff (memory persistence).

4. Detect and fix errors automatically.

5. Orchestrate multiple features: Whisper (voice in), TTS (voice out), OCR/files, Meta API Cloud.

6. Minimize human intervention — only junior staff needed for supervision.

2️⃣ Recommended AI Agent Options

Option Strengths Weakness / Notes Cost Estimate

AutoGPT (cloud-based) Full autonomy, can plan multi-step tasks, integrate APIs, maintain context with external DB Needs orchestration scripts for TTS, Whisper, and dashboard $50–$200/month (cloud hosting + API calls)

LangChain + LLM Agent Highly modular, supports memory, multi-agent orchestration, integrates tools & APIs, best for hybrid setups Slightly more complex to set up than AutoGPT; needs minimal coding for TTS/voice orchestration $50–$250/month

BabyAGI / SuperAGI Autonomous task execution, memory, can loop for error handling, lightweight for cloud deployment Less mature than LangChain; may need developer supervision for very complex workflows $50–$150/month

ChatGPT $30 + External Scripts + DB + TTS Low cost, simple setup, can handle text reasoning + voice orchestration via scripts; you control cloning, memory storage, error handling Not fully autonomous — requires script orchestration for looping, retries, and multiple inboxes $30 + $50–$100/month for cloud compute / DB / TTS usage.

3️⃣ Recommended Approach

Since your SDK is still in sandbox, you have two options:

1. Keep SDK as safety net (optional)

Pros: Can run tests and verify API calls.

Cons: Sandbox only; doesn’t provide real automation.

Recommendation: Keep temporarily for testing, then remove once AI agent handles live workflows.

2. Full migration to autonomous AI agent

Use AutoGPT or LangChain Agent as primary orchestrator.

Integrate:

Whisper → convert incoming voice to text

TTS → convert AI replies to audio for Meta API Cloud

Vector DB / Memory store → maintain per-staff context

Dashboard → supervise cloned agent inboxes.

4️⃣ Hybrid Cost-Effective Setup (Recommended)

Layer Tool / Service

Core AI reasoning ChatGPT $30 or GPT-4o via AutoGPT/LangChain

Multi-step orchestration AutoGPT / LangChain agent

Context memory Pinecone / Weaviate / Supabase (vector DB)

Voice in Whisper API

Voice out ElevenLabs / OpenAI TTS / Kurea

File OCR Google Vision API / PyMuPDF

Meta API Cloud integration Direct API calls, optional SDK only for testing

Dashboard Python/Node backend with admin override & agent inboxes

Total estimated cost: $150–$400/month (AI agent cloud + API usage + TTS + DB)

You get: fully autonomous multi-agent system, minimal human oversight, real-time TTS/text integration, error handling, and persistent memory.

5️⃣ Suggested Migration Plan

1. Phase 1: Parallel Sandbox Testing

Keep SDK in sandbox.

Deploy AI agent (AutoGPT/LangChain) in parallel to test tasks, TTS, Whisper, and files.

2. Phase 2: Memory & Multi-Agent Setup

Implement vector DB for per-staff context.

Clone agents per staff inbox.

3. Phase 3: Live API Integration

Switch from SDK sandbox to direct Meta API Cloud calls.

AI agent controls task execution fully.

4. Phase 4: Dashboard & Supervision

Use dashboard for monitoring AI actions, overrides, and logs.

staff supervises only; no coding required for daily operation.

✅ Why This Works for You

Fully autonomous → no dependence on staff for multi-step logic.

Handles all features: context, TTS, Whisper, OCR/files, Meta API.

Sandbox SDK can be phased out once stable.Slight cost rise,but saves human effort and reduces risk of errors in live deployment.

Concerns & Analysis

From the report.

Moiz have concerns center on backend access limitations for Sukkur & Mehak (cloud Oracle Forms), preventing direct integration with WATI.

Previosuly You successfully Multi Project steps (Python-oracledb scripts,WATI API auth,env vars, backups).

but the "same model" software differs in deployment (cloud UI-only vs. local DB in Multi), blocking full replication.

Why Asking for Credentials (Unlike Multi)

Concerns:

No backend/DB credentials provided—only web UI login (e.g., http://ca74.att-apps.com:8889/forms/frmservlet).

This blocks live data fetch/triggering (e.g., payments/receipts for notifications).

Exports are image PDFs (repc.pdf), unusable for automation (OCR inefficient/unreliable).

Unlike Multi: Multi had full DB access (oracledb connection worked);

Sukkur & Mehak is cloud Forms (UI-focused,no backend exposure).

EX Developer ABSENCE means no handover for creds,stalling real-time integration.

Impact:

Can't poll tables or send automated messages without creds;

prepared components (scripts) are ready but idle.

Why Can't Do It Like Earlier

Barriers:

Cloud Forms restricts to UI (no SQL queries/inserts);PDF exports lack structured data.

Multi was DB-direct (staging tables, poller feasible); here,no equivalent without creds.

Developer Leave:

No escalation path for creds, forcing feasibility analysis over execution.

Solutions:

NEXT Step-by-Step Plan WITHOUT creds while implementing 95% workaround (middleware with manual/UI staging) Todays to MVP.

Try to get Credentials (Today, wait 1 Hour):

Who: You/team lead (contact developer/manager for DB creds/service name).

Request backend access (user/pass,DSN like Multi's "XE") via secure channel (encrypted email).

If dont gets above in 1 Hour:

Proceed to workaround; developer leave = no block—

use UI exports as input.

Prepare/Verify Components (Today, 1 Hour):

Who: Abdul Moiz (lead).

What: Reuse Multi scripts (Python-oracledb for connection, WATI API auth).

Test env vars/backup on dev PC.

If No Access: Confirm read-only UI login works (web URL).

Build Staging & Middleware 1-2 Hours):

Who: Abdul Moiz (70%—poller/worker);

back-end dev (20%—webhook);

AI/junior dev (10%—templates).

Create staging table (whatsapp\_outbox) in local SQLite (adapt Multi DDL).

Build poller:

Fetch "pending" data from UI exports (manual PDF upload or Selenium scraping).

Send to WATI sandbox (requests.post), update status/logging, SFTP backup.

Code Adaptation (Abdul Moiz):

Use Multi's wati\_send.py; add OCR (pytesseract) for PDFs if needed.

import sqlite3

import requests

import pytesseract # For PDF OCR if exports used

from PIL import Image # pip install pillow pytesseract

# Local staging (SQLite)

conn = sqlite3.connect('sukkur\_staging.db')

conn.execute('''CREATE TABLE IF NOT EXISTS whatsapp\_outbox

(id INTEGER PRIMARY KEY, phone TEXT, template\_name TEXT, status TEXT DEFAULT 'NEW')''')

conn.commit()

# Manual/UI input (e.g., parse PDF export)

def parse\_pdf\_export(pdf\_path):

image = Image.open(pdf\_path) # Assume PDF to image

text = pytesseract.image\_to\_string(image)

# Extract phone/template (custom logic)

return {'phone': '923XXXXXXXXX', 'template\_name': 'PAYMENT\_UPDATE'}

# Poller (fetch from staging or manual)

def worker():

row = conn.execute("SELECT \* FROM whatsapp\_outbox WHERE status = 'NEW'").fetchone()

if row:

data = parse\_pdf\_export('repc.pdf') if no\_db else row # Fallback to PDF

body = {'phone': data['phone'], 'template\_name': data['template\_name']}

response = requests.post('https://api.wati.io/api/v1/sendTemplateMessage', json=body, headers={'Authorization': f'Bearer {os.getenv("WATI\_KEY")}'})

if response.status\_code == 200:

conn.execute("UPDATE whatsapp\_outbox SET status = 'SENT' WHERE id = ?", (row[0],))

# Backup

upload\_to\_sftp(data)

conn.commit()

while True:

worker()

time.sleep(5)

If No Access: Use manual PDF uploads as "pending" input (95% functional).

Test End-to-End (Day 1 Hour):

Who: Abdul Moiz (test flow); back-end dev (webhook for inbound);

AI dev (templates/payloads).

Simulate:

Manual export → staging → WATI send → status update. Test webhook (receive receipts). Verify cloud sync (real-time/end-of-day exports).

If No Access:

Mock with sample PDFs; achieve 95% (no live DB trigger).

Deploy & Monitor:

Who:

Full team (Abdul Moiz deploys; Fahad verifies UI).

Deploy middleware to server;monitor quota/logs. Migrate to live DB post-creds.

If No Access:

Run on dev PC; manual sync exports daily.

Critical: 95% Middleware Plan (If No OS/DB Access)

Why: Developer leave = no creds; UI-only = no direct DB.

How:

Stage manual UI exports (PDFs) in SQLite; parse/extract data (OCR/Selenium for automation); poll/send via middleware. Full flow: Export → staging → WATI → backup/cloud sync.

Who: Abdul Moiz leads (adapt Multi code); Fahad provides UI exports; back-end/AI support testing.

Timeline: 1 day to 95% (functional notifications from manual input); upgrade to 100% post-creds.

Benefits: Delivers value now (e.g., test notifications); no block.

Recommendation:

("Good Job yesterday,now repeat on scripts!").

"Fahad arrange DB creds for Sukkur/Mehak now.

Moiz:Stage PDFs in SQLite (OCR test). Back-end: Webhook. AI: Templates.

Without No creds 95% plan is workable = go."

Recommendation:

NEW REDESIGN joint instruction message (it's actionable and team-wide), but customize it with my previous suggestions

(e.g., PDF OCR code for workaround, escalation for creds).

This addresses the report's concerns (no DB creds,PDF issues) while repeating Multi steps—unlike Multi, Sukkur/Mehak is cloud Forms (UI-only),so emphasize 95% plan (stage PDFs in SQLite, parse with OCR/Selenium).

SUMMARY OF MY 1ST MSG OF TODAY & Refined WhatsApp Draft (Copy-Paste Ready)

Moiz!

Great report—scripts from Multi are ready,but Sukkur/Mehak is cloud Forms (UI-only, no DB creds like Multi). Repeat Multi steps with 95% workaround:

Stage manual PDF exports in SQLite, OCR parse data (phone/balance),send to WATI sandbox.

TELL FAHAD TO ARRANGE creds meanwhile.Full DB later.

Roles:

Fahad: ARRANGE DB creds (user/pass/DSN) to manager;share 2-3 PDF samples (repc.pdf).

Abdul Moiz (Lead):

Build staging/poller NOW (adapt Multi code for PDFs).

Back-end: Webhook for inbound.

AI/Junior: Templates for payments/receipts.

Before Start:

1. Fahad: Confirm SSH for reset (if creds include it); backup PDFs.

2. Test UI: Load http://ca74.att-apps.com:8889/forms/frmservlet; export sample PDF.

95% Workaround (No Creds Needed—Start Immediately)

1. Local setup: python3 -m venv venv; source venv/bin/activate; pip install oracledb requests pytesseract pillow flask

2. Staging (SQLite):

```python

import sqlite3

conn = sqlite3.connect('sukkur\_staging.db')

conn.execute('''CREATE TABLE IF NOT EXISTS whatsapp\_outbox (id INTEGER PRIMARY KEY, phone TEXT, template\_name TEXT, status TEXT DEFAULT 'NEW')''')

conn.commit()

Parse PDF (OCR for data):

from PIL import Image

import pytesseract

def parse\_pdf(pdf\_path):

image = Image.open(pdf\_path) # PDF to image

text = pytesseract.image\_to\_string(image)

# Extract (custom regex for phone/balance)

return {'phone': '923XXXXXXXXX', 'template\_name': 'PAYMENT\_UPDATE'}

row = parse\_pdf('repc.pdf')

conn.execute("INSERT INTO whatsapp\_outbox (phone, template\_name) VALUES (?, ?)", (row['phone'], row['template\_name']))

conn.commit()

Poller/send to WATI:

import requests

def worker():

row = conn.execute("SELECT \* FROM whatsapp\_outbox WHERE status = 'NEW'").fetchone()

if row:

body = {'phone': row[1], 'template\_name': row[2]}

response = requests.post('https://api.wati.io/api/v1/sendTemplateMessage', json=body, headers={'Authorization': f'Bearer {os.getenv("WATI\_KEY")}'})

if response.status\_code == 200:

conn.execute("UPDATE whatsapp\_outbox SET status = 'SENT' WHERE id = ?", (row[0],))

# SFTP backup (adapt Multi)

conn.commit()

while True:

worker()

time.sleep(5)

Test: Run poller with sample PDF; verify send/update.

Cautions: No DB?

Use manual PDFs/Selenium for scraping (if OCR fails).

Backup PDFs first;mask data in logs.

Checklist: [ ]

Fahad: Creds escalation + PDF samples by EOD.

[ ] MOIZ: Staging/poller test by EOD. [ ] Back-end: Webhook.

[ ] AI: Templates tomorrow. [ ]

EVEN WITHOUT creds: 95% = PDF → WATI; full DB post-CREDS.

let's get 95% live today.

#### \*\*Why This NEW?\*\*

- \*\*MY MULTI JOINT INSTRUCTION Draft (70%)\*\*: Keeps structure (roles, steps, checklist, cautions)—motivational and junior-friendly.

- \*\*My Previous (30%)\*\*:

Adds report-specific code (OCR for PDFs, SQLite staging)—solves

"why can't like Multi"

(UI exports vs. DB). Escalates creds explicitly.

- \*\*Rationale\*\*:

Report shows YOU needs guidance on workaround (PDF) JUST repeats Multi but adapts for cloud Forms.

Developer leave = no block—95% delivers value (test notifications from exports).

it's concise, code-ready,and workable to progress.

Recommendation:

NEW REDESIGN joint instruction message (it's actionable and team-wide), but customize it with my previous suggestions

(e.g., PDF OCR code for workaround, escalation for creds).

This addresses the report's concerns (no DB creds,PDF issues) while repeating Multi steps—unlike Multi, Sukkur/Mehak is cloud Forms (UI-only),so emphasize 95% plan (stage PDFs in SQLite, parse with OCR/Selenium).

SUMMARY OF MY 1ST MSG OF TODAY & Refined WhatsApp Draft (Copy-Paste Ready)

Moiz!

Great report—scripts from Multi are ready,but Sukkur/Mehak is cloud Forms (UI-only, no DB creds like Multi). Repeat Multi steps with 95% workaround:

Stage manual PDF exports in SQLite, OCR parse data (phone/balance),send to WATI sandbox.

TELL FAHAD TO ARRANGE creds meanwhile.Full DB later.

Roles:

Fahad: ARRANGE DB creds (user/pass/DSN) to manager;share 2-3 PDF samples (repc.pdf).

Abdul Moiz (Lead):

Build staging/poller NOW (adapt Multi code for PDFs).

Back-end: Webhook for inbound.

AI/Junior: Templates for payments/receipts.

Before Start:

1. Fahad: Confirm SSH for reset (if creds include it); backup PDFs.

2. Test UI: Load http://ca74.att-apps.com:8889/forms/frmservlet; export sample PDF.

95% Workaround (No Creds Needed—Start Immediately)

1. Local setup: python3 -m venv venv; source venv/bin/activate; pip install oracledb requests pytesseract pillow flask

2. Staging (SQLite):

```python

import sqlite3

conn = sqlite3.connect('sukkur\_staging.db')

conn.execute('''CREATE TABLE IF NOT EXISTS whatsapp\_outbox (id INTEGER PRIMARY KEY, phone TEXT, template\_name TEXT, status TEXT DEFAULT 'NEW')''')

conn.commit()

Parse PDF (OCR for data):

from PIL import Image

import pytesseract

def parse\_pdf(pdf\_path):

image = Image.open(pdf\_path) # PDF to image

text = pytesseract.image\_to\_string(image)

# Extract (custom regex for phone/balance)

return {'phone': '923XXXXXXXXX', 'template\_name': 'PAYMENT\_UPDATE'}

row = parse\_pdf('repc.pdf')

conn.execute("INSERT INTO whatsapp\_outbox (phone, template\_name) VALUES (?, ?)", (row['phone'], row['template\_name']))

conn.commit()

Poller/send to WATI:

import requests

def worker():

row = conn.execute("SELECT \* FROM whatsapp\_outbox WHERE status = 'NEW'").fetchone()

if row:

body = {'phone': row[1], 'template\_name': row[2]}

response = requests.post('https://api.wati.io/api/v1/sendTemplateMessage', json=body, headers={'Authorization': f'Bearer {os.getenv("WATI\_KEY")}'})

if response.status\_code == 200:

conn.execute("UPDATE whatsapp\_outbox SET status = 'SENT' WHERE id = ?", (row[0],))

# SFTP backup (adapt Multi)

conn.commit()

while True:

worker()

time.sleep(5)

Test: Run poller with sample PDF; verify send/update.

Cautions: No DB?

Use manual PDFs/Selenium for scraping (if OCR fails).

Backup PDFs first;mask data in logs.

Checklist: [ ]

Fahad: Creds escalation + PDF samples by EOD.

[ ] MOIZ: Staging/poller test by EOD. [ ] Back-end: Webhook.

[ ] AI: Templates tomorrow. [ ]

EVEN WITHOUT creds: 95% = PDF → WATI; full DB post-CREDS.

let's get 95% live today.

#### \*\*Why This NEW?\*\*

- \*\*MY MULTI JOINT INSTRUCTION Draft (70%)\*\*: Keeps structure (roles, steps, checklist, cautions)—motivational and junior-friendly.

- \*\*My Previous (30%)\*\*:

Adds report-specific code (OCR for PDFs, SQLite staging)—solves

"why can't like Multi"

(UI exports vs. DB). Escalates creds explicitly.

- \*\*Rationale\*\*:

Report shows YOU needs guidance on workaround (PDF) JUST repeats Multi but adapts for cloud Forms.

Developer leave = no block—95% delivers value (test notifications from exports).

it's concise, code-ready,and workable to progress.

Moiz ! As per Anaysis & Recommendatio msgs of today,proceed steps by step on our Main Project Sukkur & Mehak Software.

Aap & Ummama Pahly,Asharib/Umamma ky Previous msgs ke Working Final krain.Phr New Msgs ko Follow & INTETNAL OFFICE COMMUNICATION ky same AI Project mei finishing Work kro.

Aaj ky New msgs same project ky AI ko AUTONONMOUS AI AGENT adopt/apply krny kylye hai.pahly read krky samjho/discuss kro phr execute krna.previous working mei need ho tu Backend & Frontend ka finishing work Amaan/Shazaib sy & UI/UX Summaya sy Final krwao.Aaj internal Office Project ka Aapny kaam final/sab Staff ky Mobile # sy msgs krky confirm krlo.

&

Haadi sy AGETICS AI LEAD EXTRACTION PROJECT & Fahad sy REVILANCE AI KA ACCESS LYLO & BACKGROUND/CUSTOMER SUPPORT Project Samjho.Air Tel,Retel,Vapi per AI CALL CENTRE

Ky Proposals ke Files & Analysis msgs lykr study krlo.

AI Based CUSTOMER SUPPORT Project on Air Tel,Retel ya Vapi per UPTO 1,000 USERS PER DAY YA IP PHONE/AI Call Centre ka ready made limited 50-100 Calls per day ka IP Telephonic communication setup ON ISP per Work START kro.