**OAI 4.1** -- Here’s a \*\*practical draft\*\* for replacing the mock DB/message functions with real logic and a \*\*WhatsApp API (Baileys/Node bridge) integration checklist\*\*. This will take you from simulation to a live, working WOTSON agent.

---

# 1️⃣ Real Database Functions (SQLite Example)

Assume you have a working `db/database.py` with a connection function like `get\_conn()`.

Here are Python drafts for the real data access functions you’ll need:

---

### a. Fetch Unanswered Questions

```python

from db.database import get\_conn

def get\_unanswered\_questions() -> list:

"""

Fetches all messages in WhatsAppMessages table marked as questions that haven't been answered.

"""

with get\_conn() as conn:

# Assumes 'category' is NULL for unanswered, or another logic you prefer

rows = conn.execute("""

SELECT \* FROM WhatsAppMessages

WHERE content LIKE '%?'

AND category IS NULL

AND (datetime('now') - datetime(timestamp)) < 86400 -- last 24h, adjust as needed

""")

return [dict(row) for row in rows.fetchall()]

```

---

### b. Fetch Upcoming Events

```python

from db.database import get\_conn

from datetime import datetime, timedelta

def get\_upcoming\_events(within\_days=3) -> list:

with get\_conn() as conn:

now = datetime.now()

upcoming = (now + timedelta(days=within\_days)).strftime('%Y-%m-%d')

rows = conn.execute("""

SELECT E.\*, G.id as group\_id

FROM Events E

JOIN WhatsAppGroups G ON G.event\_id = E.id

WHERE E.date BETWEEN ? AND ?

""", (now.strftime('%Y-%m-%d'), upcoming))

return [dict(row) for row in rows.fetchall()]

```

---

### c. Detect Inactive Groups

```python

def get\_group\_silence\_state(hours=8) -> list:

with get\_conn() as conn:

rows = conn.execute("""

SELECT G.id as group\_id, E.name as event\_name, MAX(M.timestamp) as last\_msg

FROM WhatsAppGroups G

JOIN WhatsAppMessages M ON G.id = M.group\_id

JOIN Events E ON G.event\_id = E.id

GROUP BY G.id

HAVING (strftime('%s', 'now') - strftime('%s', last\_msg)) > (? \* 3600)

""", (hours,))

return [dict(row) for row in rows.fetchall()]

```

---

# 2️⃣ WhatsApp Integration: Real Send Functions

Assuming your \*\*WhatsApp bridge\*\* (Node.js with Baileys) exposes an HTTP API or Webhook endpoint,

replace the Python mocks with actual API calls using `requests`:

```python

import requests

from utils.config import CONFIG

WHATSAPP\_API = CONFIG['whatsapp'].get('webhook\_url', 'http://localhost:5001/send')

def send\_group\_message(group\_id: str, message: str):

"""Send a message to a WhatsApp group via the bridge."""

payload = {

"to": group\_id,

"type": "group",

"message": message,

}

try:

res = requests.post(WHATSAPP\_API, json=payload, timeout=10)

res.raise\_for\_status()

print(f"Sent to group {group\_id}: {message}")

except Exception as e:

print(f"Failed to send group message: {e}")

def send\_private\_message(user\_id: str, message: str):

"""Send a direct message to a user via the bridge."""

payload = {

"to": user\_id,

"type": "private",

"message": message,

}

try:

res = requests.post(WHATSAPP\_API, json=payload, timeout=10)

res.raise\_for\_status()

print(f"Sent DM to {user\_id}: {message}")

except Exception as e:

print(f"Failed to send DM: {e}")

```

---

# 3️⃣ WhatsApp Baileys/Node Bridge Integration Checklist

\*\*1. Install and set up [Baileys](https://github.com/WhiskeySockets/Baileys) Node.js WhatsApp bridge:\*\*

\* On your server:

```bash

git clone https://github.com/WhiskeySockets/Baileys

cd Baileys

npm install

```

\* Write a `bridge.js` that:

\* Authenticates via WhatsApp Web (QR code)

\* Listens for incoming messages and forwards them to your Python webhook (Flask/FastAPI endpoint)

\* Exposes an HTTP endpoint (e.g., `/send`) that your Python can POST to for sending messages

\*\*2. In Python, point your `send\_\*\_message` functions to the Node bridge API URL\*\* (see example above).

\*\*3. Test end-to-end:\*\*

\* Start Node bridge (scan QR code, get WhatsApp logged in)

\* Start your Python scheduler

\* Send a test message in a group and see if your pipeline can auto-reply or DM admin

\*\*4. Harden and log:\*\*

\* Log all incoming and outgoing messages for debugging

\* Make sure errors in sending are caught and reported (so your agent never “silently fails”)

---

## 🟢 Pro Tips

\* Always back up your WhatsApp session files and DB.

\* Restrict API endpoints on the Node bridge to localhost or with a token, to avoid abuse.

\* Update `requirements.txt` to include `requests`.

---

## \*\*Summary\*\*

\* Swap mock DB/API calls with real functions as above.

\* Integrate the Node.js bridge for WhatsApp.

\* Test end-to-end, watching the logs at each step.

\*\*Let me know if you want the actual Node.js bridge sample code, or if you want to see what a full “bridge.js” might look like!\*\*