Multi-Agent Automation: Meta Insights to WhatsApp

This plan outlines the "Process and Answer" for building a 4-agent pipeline to automate business reporting.

High-Level Architecture

```
graph TD
    A[Trigger: Cron/Webhook] --> B[Data Fetch Agent]
    B -->|Raw JSON Data| C[Insight Analysis Agent]
    C -->|Summary & Metrics| D[Report Builder Agent]
    D -->|PDF/Image Report| E[WhatsApp Delivery Agent]
    E -->|Message Sent| F[Done]
```

Proposed Components

1. Data Fetch Agent (Meta Business Suite)

Responsibility: Authenticate and pull metrics from Meta Graph API.

- **Tech:** Python (`facebook_business` SDK) or Node.js.
- **Key Metrics:** Impressions, Clicks, Spend, CPC, CTR, Reach.
- **Process:**
  - Authenticate using a **System User Access Token** (Long-lived).
  - Call the `/insights` endpoint of the specific Ad Account or Page.
  - Store data in a clean JSON format for the next agent.

2. Insight Analysis Agent

Responsibility: Interpret data and extract actionable trends.

- **Tech:** LLM (GPT-4o / Claude 3.5 / Gemini 1.5 Pro) + Logic.
- **Process:**
  - Receive raw JSON data.
  - Compare current metrics with "Previous Period" (if provided).
  - Calculate MoM (Month-over-Month) or WoW growth.
  - Generate a structured "Insight Context" (e.g., "CPCs are up 20%, recommend reducing budget on Ad Set X").

3. Report Builder Agent

Responsibility: Render the analysis into a professional visual format.

- **Tech:** Jinja2 (Python) / EJS (Node.js) + `Puppeteer` or `ReportLab` .
- **Process:**
  - Load a report template (HTML/CSS).
  - Inject metrics and insights into the template.
  - Generate a **PDF** or a high-quality **PNG** image (images are often better for quick WhatsApp viewing).

4. WhatsApp Delivery Agent

Responsibility: Send the final report to the brand's WhatsApp Group.

- **Tech:** WhatsApp Business Cloud API or unofficial library (`whatsapp-web.js`).
- **Process:**
  - Authenticate with WhatsApp Business API.
  - Obtain the `group_id` of the target brand.
  - Send the Media Message (PDF/Image) with a summary caption using the `/messages` endpoint.

Orchestration Example (CrewAI)

```
from crewai import Agent, Task, Crew, Process

# Agent 1: The Fetcher
data_fetcher = Agent(
  role='Meta Data Analyst',
  goal='Fetch last 7 days metrics for {brand_name}',
  backstory='Expert in Meta Graph API and data extraction.',
  tools=[MetaApiTool()], # Custom tool
  verbose=True
)

# Agent 2: The Analyzer
insight_analyst = Agent(
  role='Business Intelligence Specialist',
  goal='Analyze trends and identify performance blockers',
  backstory='Former data scientist at Meta, knows what metrics matter.',
  verbose=True
)

# ... Define Report Builder & WhatsApp Agents ...

# Tasks
fetch_task = Task(description='Fetch metrics...', agent=data_fetcher)
analyze_task = Task(description='Analyze data...', agent=insight_analyst)
# ...

# The Crew
reporting_crew = Crew(
  agents=[data_fetcher, insight_analyst, ...],
  tasks=[fetch_task, analyze_task, ...],
  process=Process.sequential
)

result = reporting_crew.kickoff()
```

Setup & Process Steps

1. **Meta Dev Portal**: Create app -> Add Marketing API -> Generate System User Token.
2. **WhatsApp Cloud API**: Set up Meta Business Account -> Get Phone Number ID -> Get Access Token.
3. **Environment**: Use Python/Node.js environment with `dotenv` for secret management.

4. **Trigger**: Use a Github Action or a Lambda Cron job to run the script daily/weekly.

Verification Plan

Automated Tests

- Run each agent in isolation with mock data.
- Validate Meta API connectivity with a test access token.
- Verify PDF generation layout.
- Send a test message to a private WhatsApp group.