✅ BOT METRICS - STANDARD APPROACH USING SHARED PYTHON FILE (Rocketbot-Compatible)

🔧 Objective:

To track every bot's metrics like bot name, start/end time, status, transactions, etc., **without duplicating logic across bots** — using **a single, centralized bot_metric.py file**.

🧩 Tools Used:

- **Rocketbot** – no changes to Rocketbot architecture required
- **Shared Python Script** (e.g., C:/RPA/CommonModules/bot_metric.py)
- **Rocketbot command**: Execute Python File

🛠 Files Involved:

File	Location	Purpose
bot_metric.py	Shared (e.g. C:/RPA/CommonModules/)	Reusable DB logger, all bots call this
bot_variables.robot	Inside each bot	Define bot-specific values to send to bot_metric.py

---

🔄 BOT EXECUTION FLOW:

1. 🚀 Start of Bot

- Set variable: bot_name, process_name, country, start_time

- Use "Execute File Python" to call: **bot_metric.py**

→ with method: log_start()

🛠 Example Code in bot_metric.py:

```
import sys
import datetime


# Receive values from Rocketbot
bot_name = sys.argv[1]
country = sys.argv[2]
```

```
process_name = sys.argv[3]
start_time = sys.argv[4]
end_time = sys.argv[5]
total_tx = int(sys.argv[6])
success_tx = int(sys.argv[7])
failed_tx = int(sys.argv[8])
log_path = sys.argv[9]

# Now log this data into DB or status file
# Example: Save to log file
with open("C:/RPA/Metrics/metrics_log.txt", "a") as log:
    log.write(f"{bot_name}, {country}, {process_name}, {start_time}, {end_time}, {total_tx}, {success_tx}, {failed_tx}, {log_path}\n")
```

2. 🔁 During Bot Execution

- Keep tracking transactions and failures inside Rocketbot

3. ✅ At End of Bot

- Set variables: end_time, success_count, fail_count, etc.

- Use "Execute File Python" again to call: bot_metric.py

→ with method: log_end()

📄 Example of variables to pass:

```
bot_name = "NCU-Bot"
country = "Chile"
process_name = "Contract Approval"
job_run_date = "2025-05-22"
start_time = "2025-05-22 09:00:00"
end_time = "2025-05-22 09:10:00"
total_transactions = 50
success_count = 48
failed_count = 2
status_file_path = "C:/RPA/Logs/NCU/status-22052025.txt"
```

4. ❌ On Error (Optional)

- If error, call: bot_metric.py

→ with method: log_error(reason)

📦 What Goes Inside bot_metric.py?

```
def log_start (bot_name, process_name, start_time, ...):
    # Insert into DB: bot start


def log_end (bot_name, end_time, total_txn, success_txn, failure_txn, ...):
    # Update DB row with execution results


def log_error (bot_name, reason, timestamp):
    # Insert error logs
```

📈 If You Add a New Field (like application_name)?

✅ Just update bot_metric.py in shared location
✅ Add that field to Execute File Python in bots
❌ No need to rewrite code in each bot

📌 Benefits Recap:

Feature	Benefit
🔁 Reusable	One shared Python file across all bots
🧩 Rocketbot-friendly	Uses built-in Rocketbot commands
🔧 Easy Updates	Add a field once, it applies everywhere
📉 Less Errors	Central logic = easier debugging

💡 **Pro Tip:**

You can even version bot_metric.py (e.g., v1.0, v1.1) so developers know when a change was made.