🧭 PHASE 2: DYNAMIC ANTIGRAVITY INPUT MAPPING

(UI moves, CLAWBOLT adapts)

This phase solves the core risk you described:

> The “Ask anything” box moves, relocates, and changes behavior.

We will not rely on fixed coordinates. Ever.

🎯 PHASE 2 GOAL (LOCKED)

✅ Reliably locate the Antigravity “Ask anything” input box

✅ Re-map it every time before typing

✅ Fail safely and notify Telegram when uncertain

✅ Prepare hooks for password/keyring detection later

🧠 STRATEGY (WHY THIS WORKS)

We use 3-tier detection, descending in cost:

Tier 1 — Window Targeting (FAST)

- Find Antigravity window
- Bring to foreground

- Narrow search area

Tier 2 — OCR Anchor Scan (SMART)

- Screenshot window
- OCR for keywords:
    - Ask anything
    - Type your question
- Get bounding box

Tier 3 — Heuristic Input Offset (SAFE)

- Input box is below the label
- Click center-bottom offset
- Verify cursor focus

If any tier fails → screenshot + Telegram alert.

📁 NEW FILES (PHASE 2)

```
antigravity/
├── window.py       # Window detection & focus
├── mapper.py       # OCR + bounding box logic
├── injector.py     # Smart text injection (upgrade)
```

1️⃣ WINDOW DETECTION

antigravity/window.py

```
import subprocess
```

```
def focus_antigravity():
    try:
        subprocess.run(
            ["wmctrl", "-a", "Antigravity"],
            check=True
        )
        return True
    except subprocess.CalledProcessError:
        return False
```

Works on Linux Mint with wmctrl
Window title can be tuned later

2️⃣ OCR MAPPING (CORE LOGIC)

antigravity/mapper.py

```
import pytesseract
import cv2
import pyautogui
import numpy as np

KEYWORDS = ["ask anything", "type your question"]

def locate_input_box():
    screenshot = pyautogui.screenshot()
    img = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)

    data = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT)

    for i, text in enumerate(data["text"]):
        if text.lower().strip() in KEYWORDS:
            x = data["left"][i]
            y = data["top"][i]
            w = data["width"][i]
            h = data["height"][i]

            # Input box is BELOW label
            input_x = x + w // 2
            input_y = y + h + 40

            return (input_x, input_y)
```

```
    return None
```

3️⃣ SMART INJECTION (UPGRADED)

antigravity/injector.py

```
import pyautogui
import time
from antigravity.window import focus_antigravity
from antigravity.mapper import locate_input_box

CONTEXT = """Context:
Your response will be sent via send_telegram.py.
Show progress if processing.
Explain errors clearly.
"""

def send_to_antigravity(user_text):
    if not focus_antigravity():
        return False, "Antigravity window not found"

    location = locate_input_box()
    if not location:
        return False, "Input box not detected"

    x, y = location
    pyautogui.click(x, y)
    time.sleep(0.2)

    full_text = f"{CONTEXT}\n\n{user_text}"
    pyautogui.typewrite(full_text, interval=0.01)
    pyautogui.press("enter")

    return True, "Message sent"
```

4️⃣ BOT UPDATE (ERROR-AWARE)

Update in

telegram/bot.py

```
success, status = send_to_antigravity(user_text)

if not success:
    await update.message.reply_text(f"⚠️ {status}")
    image_path = capture_screen()
    await send_image(update, image_path)
    return
```

Now failure is visible, not silent.

🧪 EXPECTED BEHAVIOR (THIS IS GOOD)

Scenario	Result
Antigravity moved	Re-mapped
Input box changed	OCR retry
Antigravity closed	Telegram alert
UI lag	Screenshot shows it

This gives you observability, which most bots never have.