```python
import pyautogui
import keyboard
import json
import time
import os
import cv2
import numpy as np

COORDS_FILE = 'click_coords.json'
REGION_FILE = 'image_region.json'
IMAGE_PATH = 'confirm.png'  # Update if needed
IMAGE_PATH_FALLBACK = 'confirm 2.png'
MATCH_THRESHOLD = 0.99  # strict match

# ---------- Utilities ----------
def load_json(file, default):
    if os.path.exists(file):
        with open(file, 'r') as f:
            return json.load(f)
    return default

def save_json(file, data):
    with open(file, 'w') as f:
        json.dump(data, f, indent=4)

# ---------- Step 1: Record Point ----------
def record_point():
    x, y = pyautogui.position()
    print(f"Captured position: ({x}, {y})")

    print("Press 1 for 'Gem' or 2 for 'Slots'")
    while True:
        if keyboard.is_pressed('1'):
            label = 'Gem'
            break
        elif keyboard.is_pressed('2'):
            label = 'Slots'
            break

    data = load_json(COORDS_FILE, [])
    entry = {'x': x, 'y': y, 'label': label}
    data.append(entry)
    save_json(COORDS_FILE, data)
    print(f"Saved: {entry}")
    time.sleep(0.5)

# ---------- Step 2: Define Image Region ----------
def define_image_region():
    print("Move cursor to top-left of image region and press ENTER.")
    while not keyboard.is_pressed('enter'):
        time.sleep(0.01)
    x1, y1 = pyautogui.position()
```

```python
    print(f"Top-left corner: ({x1}, {y1})")
    time.sleep(0.5)

    print("Move cursor to bottom-right of image region and press ENTER.")
    while not keyboard.is_pressed('enter'):
        time.sleep(0.01)
    x2, y2 = pyautogui.position()
    print(f"Bottom-right corner: ({x2}, {y2})")

    left = min(x1, x2)
    top = min(y1, y2)
    width = abs(x2 - x1)
    height = abs(y2 - y1)
    region = [left, top, width, height]

    save_json(REGION_FILE, region)
    print(f"Region saved: {region}")
    time.sleep(0.5)

# ---------- Step 3: Wait for Exact Image Match with OpenCV ----------
def wait_for_image(region):
    print("Waiting for exact confirmation image...")
    template1 = cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE)
    template2 = cv2.imread(IMAGE_PATH_FALLBACK, cv2.IMREAD_GRAYSCALE)
    if template1 is None and template2 is None:
        print(f"Error: Could not read {IMAGE_PATH} or {IMAGE_PATH_FALLBACK}")
        return
    w1, h1 = template1.shape[::-1] if template1 is not None else (None, None)
    w2, h2 = template2.shape[::-1] if template2 is not None else (None, None)
    while True:
        screenshot = pyautogui.screenshot(region=region)
        screenshot = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2GRAY)
        match_found = False
        if template1 is not None:
            res1 = cv2.matchTemplate(screenshot, template1, cv2.TM_CCOEFF_NORMED)
            min_val1, max_val1, min_loc1, max_loc1 = cv2.minMaxLoc(res1)
            if max_val1 >= MATCH_THRESHOLD:
                print("Primary image match found.")
                match_found = True
            else:
                print(f"Primary match value: {max_val1:.4f} (waiting...)")
        if not match_found and template2 is not None:
            res2 = cv2.matchTemplate(screenshot, template2, cv2.TM_CCOEFF_NORMED)
            min_val2, max_val2, min_loc2, max_loc2 = cv2.minMaxLoc(res2)
            if max_val2 >= MATCH_THRESHOLD:
                print("Fallback image match found.")
                match_found = True
            else:
                print(f"Fallback match value: {max_val2:.4f} (waiting...)")
        if match_found:
            break
        time.sleep(0.2)

# ---------- Step 4: Perform Clicks ----------
def perform_clicks():
```

```python
    coords = load_json(COORDS_FILE, [])
    region = load_json(REGION_FILE, None)

    if not coords:
        print("No saved coordinates.")
        return
    if not region:
        print("Image region not defined.")
        return

    for entry in coords:
        x, y = entry['x'], entry['y']
        label = entry['label']

        pyautogui.moveTo(x, y)

        if label.lower() == 'slots':
            print(f"Left-clicking on Slots at ({x}, {y})")
            pyautogui.click(button='left')
        elif label.lower() == 'gem':
            print(f"Right-clicking on Gem at ({x}, {y})")
            pyautogui.rightClick()
            wait_for_image(region)
        else:
            print(f"Unknown label '{label}', skipping...")

        time.sleep(0.3)

# ---------- Main Listener ----------
print("Script Ready")
print("- : Capture point")
print("p : Define image region")
print("ENTER : Start automation")
print("ESC : Quit")

while True:
    if keyboard.is_pressed('-'):
        record_point()
    elif keyboard.is_pressed('p'):
        define_image_region()
    elif keyboard.is_pressed('enter'):
        perform_clicks()
        print("Automation finished. Press ENTER again to restart or ESC to quit.")
        # Wait for ENTER to be released before allowing another run
        while keyboard.is_pressed('enter'):
            pass
    elif keyboard.is_pressed('esc'):
        print("Exiting...")
        break
    time.sleep(0.1)
```