**Step-by-Step Strategy**

**1. Modular Python Automation Design**

Split the automation into modules:

* **launcher.py** → Starts ERA app
* **checker.py** → Detects green ticks using image detection or OCR
* **click\_proceed.py** → Clicks Proceed only when ready
* **login.py** → Enters user credentials
* **session\_handler.py** → Completes 2 daily sessions (with activity simulation)
* **logout.py** → Logs out current user
* **user\_loop.py** → Repeats the process for each user
* import pyautogui
* import time
* import subprocess
* from PIL import ImageGrab
* import pytesseract
* *# Set path to tesseract executable if not already in PATH*
* pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
* *# --- Step 1: Launch ERA Explorer ---*
* era\_path = r"C:\Program Files\ERA Explorer\ERA Explorer.exe"
* subprocess.Popen(era\_path)
* print("Launching ERA Explorer...")
* *# --- Step 2: Wait a few seconds for window to open ---*
* time.sleep(5)
* *# --- Step 3: Define correct tick region ---*
* TICK\_REGION = (487, 642, 904, 808)  *# Corrected coordinates (x1, y1, x2, y2)*
* *# --- Step 4: Wait for green ticks to appear ---*
* def wait\_for\_ready\_text():
* print("Waiting for green ticks using OCR...")
* while True:
* screenshot = ImageGrab.grab(bbox=TICK\_REGION)
* text = pytesseract.image\_to\_string(screenshot).lower()
* print("OCR Result:", text)
* if "ready" in text or "proceed" in text:
* print("System Ready Detected!")
* break
* time.sleep(1)
* *# --- Step 5: Click Proceed button ---*
* def click\_proceed():
* proceed\_btn\_coords = (712, 615)
* pyautogui.moveTo(proceed\_btn\_coords[0], proceed\_btn\_coords[1], duration=1)
* pyautogui.click()
* print("Clicked Proceed button ✅")
* *# --- Step 6: Wait and click "Skip Wizard" ---*
* def wait\_and\_skip\_wizard():
* print("Waiting for 'Skip Wizard' button...")
* while True:
* screenshot = ImageGrab.grab()
* text = pytesseract.image\_to\_string(screenshot)
* if "Skip Wizard" in text or "SKIP WIZARD" in text:
* print("Found 'Skip Wizard'... clicking.")
* pyautogui.click(x=1011, y=161)  *# Update if needed based on actual button location*
* time.sleep(2)
* break
* time.sleep(1)
* *# --- Execution starts here ---*
* wait\_for\_ready\_text()
* click\_proceed()
* wait\_and\_skip\_wizard()

Era Automation updated code

import pyautogui

import time

import subprocess

from PIL import ImageGrab

import pytesseract

import csv

*# --- Configuration ---*

pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

ERA\_PATH = r"C:\Program Files\ERA Explorer\ERA Explorer.exe"

TICK\_REGION = (487, 642, 904, 808)

PROCEED\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\proceed\_button.png"

SKIP\_WIZARD\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\skip wizard interface.PNG"

LOGIN\_AGAIN\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\loginAgain.PNG"

USERPASS\_INTERFACE\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\UserPassInterface.png"

LOGIN\_SUCCESS\_INTERFACE\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\LoginSuccessInterface.png"

RESUME\_LEARNING\_BUTTON\_IMAGE = r"C:\Users\Administrator\Desktop\era automation\ResumeLearningButton.PNG"

LOGIN\_AGAIN\_COORDS = (232, 657)

RESUME\_LEARNING\_COORDS = (365, 387)

USER\_FIELD = (280, 427)

PASS\_FIELD = (286, 493)

LOGIN\_BUTTON = (245, 614)

CSV\_PATH = r"C:\Users\Administrator\Desktop\era automation\users.csv"

*# --- Functions ---*

def load\_users(csv\_path):

    users = []

    with open(csv\_path, mode='r', newline='') as file:

        reader = csv.DictReader(file)

        for row in reader:

            users.append({'username': row['username'], 'password': row['password']})

    return users

def launch\_era():

    print("🔄 Launching ERA Explorer...")

    subprocess.Popen(ERA\_PATH)

    time.sleep(5)

def wait\_for\_ready\_ticks():

    print("🕵️‍♂️ Waiting for green ticks using OCR...")

    while True:

        screenshot = ImageGrab.grab(bbox=TICK\_REGION)

        text = pytesseract.image\_to\_string(screenshot).lower()

        print("🔍 OCR Text:", text)

        if "ready" in text or "proceed" in text:

            print("✅ Green ticks detected!")

            break

        time.sleep(1)

def wait\_for\_button(image\_path, button\_name):

    print(f"⌛ Waiting for '{button\_name}' button to appear...")

    while True:

        try:

            location = pyautogui.locateOnScreen(image\_path, confidence=0.8)

            if location:

                center = pyautogui.center(location)

                print(f"✅ '{button\_name}' button found at {center}")

                return center

        except pyautogui.ImageNotFoundException:

            pass

        time.sleep(0.5)

def click\_button(center, name="button"):

    pyautogui.moveTo(center.x, center.y, duration=0.5)

    pyautogui.click()

    print(f"🖱️ Clicked '{name}'")

def wait\_for\_login\_interface():

    print("⌛ Waiting for login screen interface to appear...")

    while True:

        try:

            location = pyautogui.locateOnScreen(USERPASS\_INTERFACE\_IMAGE, confidence=0.85)

            if location:

                print(f"✅ Login interface detected at {location}")

                return

        except Exception as e:

            print(f"⚠️ Error detecting login screen: {e}")

        time.sleep(0.5)

def login\_user(username, password):

    print(f"🔐 Logging in with username: {username}")

    wait\_for\_login\_interface()

    pyautogui.click(USER\_FIELD)

    pyautogui.hotkey("ctrl", "a")

    pyautogui.typewrite(username, interval=0.05)

    pyautogui.click(PASS\_FIELD)

    pyautogui.hotkey("ctrl", "a")

    pyautogui.typewrite(password, interval=0.05)

    pyautogui.click(LOGIN\_BUTTON)

    print("🔓 Login button clicked.")

    time.sleep(3)

    handle\_active\_session(LOGIN\_AGAIN\_IMAGE, LOGIN\_AGAIN\_COORDS)

def handle\_active\_session(image\_path, button\_coords):

    print("🔎 Checking for 'Found another active session' message...")

    try:

        location = pyautogui.locateOnScreen(image\_path, confidence=0.8)

        if location:

            print("⚠️ Active session found. Clicking 'Login Again'...")

            pyautogui.moveTo(button\_coords[0], button\_coords[1], duration=0.5)

            pyautogui.click()

            print("✅ Clicked 'Login Again'")

        else:

            print("✔️ No active session warning detected.")

    except Exception as e:

        print("❌ Error checking active session:", e)

def wait\_for\_login\_success\_and\_click\_resume():

    print("🧠 Waiting for login success interface...")

    while True:

        try:

            location = pyautogui.locateOnScreen(LOGIN\_SUCCESS\_INTERFACE\_IMAGE, confidence=0.85)

            if location:

                print("✅ Login success interface detected!")

                time.sleep(1)

                pyautogui.moveTo(RESUME\_LEARNING\_COORDS[0], RESUME\_LEARNING\_COORDS[1], duration=0.5)

                pyautogui.click()

                print("🎯 Clicked 'Resume Learning'")

                return

        except Exception as e:

            print("⚠️ Error detecting login success interface:", e)

        time.sleep(0.5)

*# --- Main Execution ---*

launch\_era()

wait\_for\_ready\_ticks()

*# Step 1: Click "Proceed"*

proceed\_center = wait\_for\_button(PROCEED\_IMAGE, "Proceed")

click\_button(proceed\_center, "Proceed")

*# Step 2: Skip Wizard*

time.sleep(3)

skip\_center = wait\_for\_button(SKIP\_WIZARD\_IMAGE, "Skip Wizard")

click\_button(skip\_center, "Skip Wizard")

*# Step 3: Login with user from CSV*

users = load\_users(CSV\_PATH)

if users:

    login\_user(users[0]['username'], users[0]['password'])

    wait\_for\_login\_success\_and\_click\_resume()

else:

    print("⚠️ No users found in CSV!")

print("✅ Full automation completed successfully.")

**✨ Highlights:**

* Waits for login screen before typing
* Waits for successful login screen before clicking “Resume Learning”
* Uses your exact coordinates
* Very stable & readable

Let me know when you want to **loop through all users** in the CSV one by one with login + logout flow — we can plug that in next.

You're doing awesome work with this automation!