**Lab Exercise 5 - Capturing Screenshots Using Python for RPA**

**Objective**

Learn how to capture, save, and manipulate screenshots using Python for RPA tasks. This exercise will cover taking full-screen screenshots, capturing specific regions, and naming and organizing screenshots dynamically.

**Prerequisites**

1. Python 3.x installed on your system.
2. Install the **pyautogui** library:

pip install pyautogui

1. Optionally, install the **pillow** library for advanced image handling:

pip install pillow

**Exercise Steps**

**Step 1: Capturing a Full-Screen Screenshot**

1. **Objective**: Capture and save a screenshot of the entire screen.
2. **Code**:

import pyautogui

# Capture the entire screen

screenshot = pyautogui.screenshot()

# Save the screenshot

screenshot.save("full\_screen.png")

print("Screenshot saved as full\_screen.png.")

1. **Expected Output**:  
   A file named full\_screen.png will be saved in the current working directory.

**Step 2: Capturing a Specific Region**

1. **Objective**: Capture a specific rectangular region of the screen.
2. **Code**:

import pyautogui

# Define the region (x, y, width, height)

region = (100, 100, 500, 400)

# Capture the region

screenshot = pyautogui.screenshot(region=region)

# Save the screenshot

screenshot.save("specific\_region.png")

print("Screenshot of specific region saved as specific\_region.png.")

1. **Expected Output**:  
   A file named specific\_region.png will contain the specified area of the screen.

**Step 3: Adding a Timestamp to the Screenshot Name**

1. **Objective**: Save the screenshot with a unique name including the current timestamp.
2. **Code**:

import pyautogui

from datetime import datetime

# Capture the screenshot

screenshot = pyautogui.screenshot()

# Generate a timestamped filename

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

filename = f"screenshot\_{timestamp}.png"

# Save the screenshot

screenshot.save(filename)

print(f"Screenshot saved as {filename}.")

1. **Expected Output**:  
   A screenshot with a filename like screenshot\_20231207\_103045.png will be saved.

**Step 4: Capturing and Displaying the Screenshot**

1. **Objective**: Display the screenshot after capturing it.
2. **Code**:

import pyautogui

from PIL import Image

# Capture the screenshot

screenshot = pyautogui.screenshot()

# Display the screenshot

screenshot.show()

print("Screenshot captured and displayed.")

1. **Expected Output**:  
   The screenshot will open in the default image viewer on your system.

**Step 5: Automating Periodic Screenshots**

1. **Objective**: Capture screenshots at regular intervals.
2. **Code**:

import pyautogui

from datetime import datetime

import time

# Set the interval (in seconds)

interval = 5

for i in range(3): # Take 3 screenshots

# Capture the screenshot

screenshot = pyautogui.screenshot()

# Generate a timestamped filename

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

filename = f"screenshot\_{timestamp}.png"

# Save the screenshot

screenshot.save(filename)

print(f"Screenshot {i+1} saved as {filename}.")

# Wait for the next interval

time.sleep(interval)

1. **Expected Output**:  
   Three screenshots will be taken at 5-second intervals and saved with timestamped filenames.

**Step 6: Capturing and Cropping a Screenshot**

1. **Objective**: Crop a screenshot to focus on a specific area.
2. **Code**:

import pyautogui

from PIL import Image

# Capture the entire screen

screenshot = pyautogui.screenshot()

# Define the cropping box (left, upper, right, lower)

crop\_box = (100, 100, 600, 500)

cropped\_screenshot = screenshot.crop(crop\_box)

# Save the cropped screenshot

cropped\_screenshot.save("cropped\_screenshot.png")

print("Cropped screenshot saved as cropped\_screenshot.png.")

1. **Expected Output**:  
   A file named cropped\_screenshot.png will contain the cropped region of the screen.

**Step 7: Capturing a Screenshot Based on User Input**

1. **Objective**: Allow the user to capture a screenshot by pressing a key.
2. **Code**:

import pyautogui

import keyboard

print("Press 's' to take a screenshot and 'q' to quit.")

while True:

if keyboard.is\_pressed("s"):

screenshot = pyautogui.screenshot()

screenshot.save("user\_screenshot.png")

print("Screenshot saved as user\_screenshot.png.")

break

elif keyboard.is\_pressed("q"):

print("Exiting without capturing a screenshot.")

break

1. **Expected Output**:  
   Pressing s saves a screenshot, while pressing q exits the program.