```python
import pytest
import allure
from pywinauto import Application
from pywinauto.timings import wait_until_passes
import subprocess
import time
import io

PYTHON_EXE = r"<path-to-python-executable>"
APP_SCRIPT = r"D:\Coding\PyWinAuto\Application.py"

@pytest.fixture(scope="module")
def app_fixture():
    process = subprocess.Popen([PYTHON_EXE, APP_SCRIPT])
    # creates a PyWinAuto application object using UI Automation backend and allows magic lookup for controls
    app = Application(backend="uia", allow_magic_lookup=True)
    main_window = wait_until_passes(
        timeout=100,
        retry_interval=0.5,
        func=lambda: app.connect(title_re="Simple Controls Demo.*")
                        .window(title_re="Simple Controls Demo.*")
    )
    # Wait for the main window to be visible and ready for interaction
    main_window.wait("visible", timeout=10)
    # Set focus to the main window to ensure it is active for interactions
    main_window.set_focus()
    yield app, main_window
    # Close the application after tests are done
    main_window.close()
    process.terminate()

@allure.title("Button click updates status label")
def test_button_click(app_fixture):
    app, window = app_fixture
    # Find the button and status label controls
    btn = window.child_window(title="Click Me", control_type="Button").wrapper_object()
    status_label = window.child_window(title_re="Status: .*", control_type="Text").wrapper_object()

    # Use allure steps to capture screenshots and attach status text
    with allure.step("Capture before snapshot"):
        img_bytes = io.BytesIO()
        window.capture_as_image().save(img_bytes, format='JPEG')
        allure.attach(img_bytes.getvalue(), name="Before snapshot", attachment_type=allure.attachment_type.JPG)

    # Click the button and wait for the status label to update
    with allure.step("Click the button"):
        btn.click_input()
    status_text = wait_until_passes(
        timeout=5,
        retry_interval=0.5,
        func=lambda: status_label.window_text() if "Status: Button clicked" in status_label.window_text() else (_ for _ in ()).throw(AssertionError("waiting")),
    )

    # Capture after snapshot and attach status text
    with allure.step("Capture after snapshot"):
        img_bytes = io.BytesIO()
        window.capture_as_image().save(img_bytes, format='JPEG')
        allure.attach(img_bytes.getvalue(), name="After snapshot", attachment_type=allure.attachment_type.JPG)

    # Attach the status text to Allure report and assert the expected value
    allure.attach(status_text, name="Status label text", attachment_type=allure.attachment_type.TEXT)
    assert "Status: Button clicked" in status_text

```

```
64  @allure.title("Toggle checkbox updates status label")
65  def test_checkbox_toggle(app_fixture):
66      app, window = app_fixture
67      # Find the checkbox and status label controls
68      # Using object names for more reliable control identification
69      checkbox = window["checkbox"].wrapper_object()
70      # Using regex to find the status label that starts with "Status: " to ensure we get
        the correct control
71      status_label = window.child_window(title_re="Status: .*",
        control_type="Text").wrapper_object()
72      # Use allure steps to capture screenshots and attach status text
73      with allure.step("Enable checkbox"):
74          # Capture before enabling the checkbox
75          with allure.step("Capture before enabling"):
76              img_bytes = io.BytesIO()
77              window.capture_as_image().save(img_bytes, format='JPEG')
78              allure.attach(img_bytes.getvalue(), name="Before enable snapshot",
                attachment_type=allure.attachment_type.JPG)
79          # Click the checkbox to enable it
80          checkbox.click_input()
81          # Wait for the status label to update to "Checkbox on"
82          status_on = wait_until_passes(
83              timeout=5,
84              retry_interval=0.5,
85              func=lambda: status_label.window_text() if "Checkbox on" in
                status_label.window_text() else (_ for _ in
                ()).throw(AssertionError("waiting")),
86          )
87          # Capture after enabling the checkbox
88          with allure.step("Capture after enabling"):
89              img_bytes = io.BytesIO()
90              window.capture_as_image().save(img_bytes, format='JPEG')
91              allure.attach(img_bytes.getvalue(), name="After enable snapshot",
                attachment_type=allure.attachment_type.JPG)
92          # Attach the status text to Allure report and assert the expected value
93          allure.attach(status_on, name="Checkbox on state",
            attachment_type=allure.attachment_type.TEXT)
94          assert "Checkbox on" in status_on
95      # Now disable the checkbox and verify the status update
96      with allure.step("Disable checkbox"):
97          # Capture before disabling the checkbox
98          with allure.step("Capture before disabling"):
99              img_bytes = io.BytesIO()
100             window.capture_as_image().save(img_bytes, format='JPEG')
101             allure.attach(img_bytes.getvalue(), name="Before disable snapshot",
                attachment_type=allure.attachment_type.JPG)
102         # Click the checkbox to disable it
103         checkbox.click_input()
104         # Wait for the status label to update to "Checkbox off"
105         status_off = wait_until_passes(
106             timeout=5,
107             retry_interval=0.5,
108             func=lambda: status_label.window_text() if "Checkbox off" in
                status_label.window_text() else (_ for _ in
                ()).throw(AssertionError("waiting")),
109         )
110         # Capture after disabling the checkbox
111         with allure.step("Capture after disabling"):
112             img_bytes = io.BytesIO()
113             window.capture_as_image().save(img_bytes, format='JPEG')
114             allure.attach(img_bytes.getvalue(), name="After disable snapshot",
                attachment_type=allure.attachment_type.JPG)
115         # Attach the status text to Allure report and assert the expected value
116         allure.attach(status_off, name="Checkbox off state",
            attachment_type=allure.attachment_type.TEXT)
117         assert "Checkbox off" in status_off
118
119 @allure.title("Select combo option and verify status")
120 def test_combo_selection(app_fixture):
```

```
121       app, window = app_fixture
122       # Find the combo box and status label controls
123       combo = window["combo"].wrapper_object()
124       status_label = window.child_window(title_re="Status: .*",
          control_type="Text").wrapper_object()
125
126       # Use allure steps to capture screenshots and attach status text
127       with allure.step("Capture before snapshot"):
128           img_bytes = io.BytesIO()
129           window.capture_as_image().save(img_bytes, format='JPEG')
130           allure.attach(img_bytes.getvalue(), name="Before snapshot",
              attachment_type=allure.attachment_type.JPG)
131
132       # Select "Option 2" from the combo box and wait for the status label to update
133       with allure.step("Select Option 2"):
134           combo.select("Option 2")
135
136       status_text = wait_until_passes(
137           timeout=5,
138           retry_interval=0.5,
139           func=lambda: status_label.window_text() if "Selected Option 2" in
              status_label.window_text() else (_ for _ in ()).throw(AssertionError("waiting")),
140       )
141
142       # Capture after snapshot and attach status text
143       with allure.step("Capture after snapshot"):
144           img_bytes = io.BytesIO()
145           window.capture_as_image().save(img_bytes, format='JPEG')
146           allure.attach(img_bytes.getvalue(), name="After snapshot",
              attachment_type=allure.attachment_type.JPG)
147
148       # Attach the status text to Allure report and assert the expected value
149       allure.attach(status_text, name="Combo selection status",
          attachment_type=allure.attachment_type.TEXT)
150       assert "Selected Option 2" in status_text
151
152   @allure.title("Switch radio modes and verify status")
153   def test_radio_buttons(app_fixture):
154       app, window = app_fixture
155       # Find the radio buttons and status label controls
156       radio_a = window.child_window(title="Mode A",
          control_type="RadioButton").wrapper_object()
157       radio_b = window.child_window(title="Mode B",
          control_type="RadioButton").wrapper_object()
158       status_label = window.child_window(title_re="Status: .*",
          control_type="Text").wrapper_object()
159
160       # Use allure steps to capture screenshots and attach status text
161       with allure.step("Select Mode B"):
162           # Capture before selecting Mode B
163           with allure.step("Capture before selecting Mode B"):
164               img_bytes = io.BytesIO()
165               window.capture_as_image().save(img_bytes, format='JPEG')
166               allure.attach(img_bytes.getvalue(), name="Before Mode B snapshot",
                  attachment_type=allure.attachment_type.JPG)
167           # Click the Mode B radio button
168           radio_b.click_input()
169           # Wait for the status label to update to "Mode B selected"
170           mode_b_text = wait_until_passes(
171               timeout=5,
172               retry_interval=0.5,
173               func=lambda: status_label.window_text() if "Mode B selected" in
                  status_label.window_text() else (_ for _ in
                  ()).throw(AssertionError("waiting")),
174           )
175           # Capture after selecting Mode B
176           with allure.step("Capture after selecting Mode B"):
177               img_bytes = io.BytesIO()
178               window.capture_as_image().save(img_bytes, format='JPEG')
```

```
179             allure.attach(img_bytes.getvalue(), name="After Mode B snapshot",
                attachment_type=allure.attachment_type.JPG)
180         # Attach the status text to Allure report and assert the expected value
181         allure.attach(mode_b_text, name="Mode B status",
            attachment_type=allure.attachment_type.TEXT)
182         assert "Mode B selected" in mode_b_text
183
184     # Now select Mode A and verify the status update
185     with allure.step("Select Mode A"):
186         # Capture before selecting Mode A
187         with allure.step("Capture before selecting Mode A"):
188             img_bytes = io.BytesIO()
189             window.capture_as_image().save(img_bytes, format='JPEG')
190             allure.attach(img_bytes.getvalue(), name="Before Mode A snapshot",
                attachment_type=allure.attachment_type.JPG)
191
192         # Click the Mode A radio button
193         radio_a.click_input()
194
195         # Wait for the status label to update to "Mode A selected"
196         mode_a_text = wait_until_passes(
197             timeout=5,
198             retry_interval=0.5,
199             func=lambda: status_label.window_text() if "Mode A selected" in
                status_label.window_text() else (_ for _ in
                ()).throw(AssertionError("waiting")),
200         )
201
202         # Capture after selecting Mode A
203         with allure.step("Capture after selecting Mode A"):
204             img_bytes = io.BytesIO()
205             window.capture_as_image().save(img_bytes, format='JPEG')
206             allure.attach(img_bytes.getvalue(), name="After Mode A snapshot",
                attachment_type=allure.attachment_type.JPG)
207
208         # Attach the status text to Allure report and assert the expected value
209         allure.attach(mode_a_text, name="Mode A status",
            attachment_type=allure.attachment_type.TEXT)
210         assert "Mode A selected" in mode_a_text
211
212 @allure.title("Enter text and verify status update")
213 def test_text_input(app_fixture):
214     app, window = app_fixture
215     # Find the text box and status label controls
216     text_box = window.child_window(control_type="Edit").wrapper_object()
217     status_label = window.child_window(title_re="Status: .*",
        control_type="Text").wrapper_object()
218     # Use allure steps to capture screenshots and attach status text
219     with allure.step("Capture before snapshot"):
220         img_bytes = io.BytesIO()
221         window.capture_as_image().save(img_bytes, format='JPEG')
222         allure.attach(img_bytes.getvalue(), name="Before snapshot",
            attachment_type=allure.attachment_type.JPG)
223     # Enter text into the text box and wait for the status label to update
224     with allure.step("Enter text into edit box"):
225         text_box.set_text("Hello World")
226         allure.attach("Hello World", name="Entered text",
            attachment_type=allure.attachment_type.TEXT)
227
228     status_text = wait_until_passes(
229         timeout=5,
230         retry_interval=0.5,
231         func=lambda: status_label.window_text() if "Status: Text box contains 'Hello
            World'" in status_label.window_text() else (_ for _ in
            ()).throw(AssertionError("waiting")),
232     )
233     # Capture after snapshot and attach status text
234     with allure.step("Capture after snapshot"):
235         img_bytes = io.BytesIO()
```

```
236             window.capture_as_image().save(img_bytes, format='JPEG')
237             allure.attach(img_bytes.getvalue(), name="After snapshot",
                attachment_type=allure.attachment_type.JPG)
238         # Attach the status text to Allure report and assert the expected value
239         allure.attach(status_text, name="Text input status",
            attachment_type=allure.attachment_type.TEXT)
240         assert "Status: Text box contains 'Hello World'" in status_text
```