测试用例编号：TSGL\_CSYL\_001

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_001 "检查输入的账号是否存在数据库中; 确保账号符合特定格式要求（例如长度、字符类型）;" TSGLXT-XQ-0001 输入账号长度小于指定范围，应提示账号长度不足; TSGLXT-CSD-0001 "系统已启动并运行正常 用户已经登录系统" 测试系统：Windows 10，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" abcd1234\_ 提示账号验证通过 "如果提示信息与预期结果一致，则认为该测试用例通过 如果提示信息与预期结果不一致，则认为该测试用例失败" 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  username\_input.clear()  password\_input.clear()  username\_input.send\_keys(username)  password\_input.send\_keys(password)  submit\_button.click()  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]")))  assert expected\_error\_message in error\_message.text  else:  success\_page = WebDriverWait(driver, 10).until(EC.url\_contains("success\_page\_url")) # Replace with actual URL  assert success\_page  print("测试用例1通过")  except AssertionError as e:  print(f"测试用例1失败: {e}")  except Exception as e:  print(f"测试用例1错误: {e}")  finally:  try:  confirm\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]")))  confirm\_button.click()  time.sleep(1)  except Exception as e:  print(f"Error closing popup in test case 1: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)  driver.switch\_to.frame("view\_frame")  run\_test\_case(driver, "abcd1234\_", "91874", "账号长度不足")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_002

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_002 "检查输入的账号是否存在数据库中; 确保账号符合特定格式要求（例如长度、字符类型）;" TSGLXT-XQ-0001 输入账号长度大于指定范围，应提示账号长度过长; TSGLXT-CSD-0002 "系统已启动并运行正常 用户已经访问过系统" 测试系统：Windows 11，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" 1234567890123456789012345678901234567890 提示账号长度过长 "如果提示信息与预期结果一致，则认为该测试用例通过 如果提示信息与预期结果不一致，则认为该测试用例失败" 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例2通过")  except AssertionError as e:  print(f"测试用例2失败: {e}")  except Exception as e:  print(f"测试用例2错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 2: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp")  driver.switch\_to.frame("view\_frame")  # 等待账号输入框元素加载完成  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  # 等待密码输入框元素加载完成  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  # 等待提交按钮元素加载完成  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1234567890123456789012345678901234567890", "13847", "账号长度过长")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_003

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_003 "检查输入的账号是否存在数据库中; 确保账号符合特定格式要求（例如长度、字符类型）;" TSGLXT-XQ-0001 输入账号包含非法字符，应提示账号包含非法字符; TSGLXT-CSD-0003 "系统已启动并运行正常 用户已经访问过系统" 测试系统：Windows 12，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" abcdefghijklmnopqrstuvwxyz 提示账号只包含字母 "如果提示信息与预期结果一致，则认为该测试用例通过 如果提示信息与预期结果不一致，则认为该测试用例失败" 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例3通过")  except AssertionError as e:  print(f"测试用例3失败: {e}")  except Exception as e:  print(f"测试用例3错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 3: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "abcdefghijklmnopqrstuvwxyz", "1384743819", "账号包含非法字符")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_004

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_004 "检查输入的账号是否存在数据库中; 确保账号符合特定格式要求（例如长度、字符类型）;" TSGLXT-XQ-0001 输入符合要求的账号，应通过格式检查; TSGLXT-CSD-0004 "系统已启动并运行正常 用户已经访问过系统" 测试系统：Windows 13，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" 12345678901234567890 提示账号只包含数字 "如果提示信息与预期结果一致，则认为该测试用例通过 如果提示信息与预期结果不一致，则认为该测试用例失败" 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例4通过")  except AssertionError as e:  print(f"测试用例4失败: {e}")  except Exception as e:  print(f"测试用例4错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 4: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "12345678901234567890", "2189364", "账号或密码错误")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_005

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_005 将密码进行验证、若是正确密码则成功登录 TSGLXT-XQ-0002 正确密码验证：输入正确的密码，与数据库中的密码匹配; TSGLXT-CSD-0005 系统已启动并运行正常；用户已经登录系统 测试系统：Windows 14，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" "| T001 | 1234567890123456789012345678901234567890 | | T002 | abcd1234\_ | | T003 | 12345678901234567890 | | T004 | abcdefghijklmnopqrstuvwxyz" "| T001 | 提示密码长度过长 | | T002 | 提示密码验证通过 | | T003 | 提示密码只包含数字 | | T004 | 提示密码只包含字母 |" 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例5通过")  except AssertionError as e:  print(f"测试用例5失败: {e}")  except Exception as e:  print(f"测试用例5错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 5: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "T001", "12345678901234567890", "密码长度过长")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_006

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_006 将密码进行验证、若是正确密码则成功登录 TSGLXT-XQ-0002 错误密码验证：输入错误的密码，与数据库中的密码匹配; TSGLXT-CSD-0006 系统已启动并运行正常；用户已经登录系统 测试系统：Windows 15，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" abcd1234\_ | 提示密码验证不通过 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例6通过")  except AssertionError as e:  print(f"测试用例6失败: {e}")  except Exception as e:  print(f"测试用例6错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 6: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "abcd1234\_", "467823918274", "密码验证不通过")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_007

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_007 将密码进行验证、若是正确密码则成功登录 TSGLXT-XQ-0002 特殊字符密码验证：输入包含特殊字符的密码，验证过程应正确处理; TSGLXT-CSD-0007 系统已启动并运行正常；用户已经登录系统 测试系统：Windows 16，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" 1234567890123 abcdefghijklmnopqrstuvwxyz 提示密码只包含数字 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例7通过")  except AssertionError as e:  print(f"测试用例7失败: {e}")  except Exception as e:  print(f"测试用例7错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 7: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1234567890123", "abcdefghijklmnopqrstuvwxyz", "密码只包含数字")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_008

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_008 将密码进行验证、若是正确密码则成功登录 TSGLXT-XQ-0002 成功登录：验证正确的密码后，用户应能成功登录并接收到成功消息; TSGLXT-CSD-0008 系统已启动并运行正常；用户已经登录系统 测试系统：Windows 17，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" 正确的用户名和密码 成功接收消息 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例8通过")  except AssertionError as e:  print(f"测试用例8失败: {e}")  except Exception as e:  print(f"测试用例8错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 8: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1805010220", "0123457", None)  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_009

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_009 将密码进行验证、若是正确密码则成功登录 TSGLXT-XQ-0002 登录后操作：测试登录成功后是否正确执行了后续操作（如页面跳转）; TSGLXT-CSD-0009 系统已启动并运行正常；用户已经登录系统 测试系统：Windows 18，浏览器：Chrome 最新版本 "1.打开系统界面 2.在账号输入框中输入指定的账号 3.点击“提交”按钮" 正确的用户名和密码 页面正常跳转 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例9通过")  except AssertionError as e:  print(f"测试用例9失败: {e}")  except Exception as e:  print(f"测试用例9错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 9: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1532764571243", "216485", "账号或密码错误")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_010

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_010 "提供明确的错误提示，当账号或密码不正确时; 避免透露具体的错误原因（即不区分是账号错误还是密码错误）;" TSGLXT-XQ-0003 账号错误：输入错误的账号，检查是否显示通用错误提示; TSGLXT-CSD-0010 提供明确的错误提示，当账号或密码不正确时; 测试系统：Windows 19，浏览器：Chrome 最新版本 避免透露具体的错误原因（即不分区是账号错误还是密码错误） 输入错误的账号 显示通用错误提示; 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例10通过")  except AssertionError as e:  print(f"测试用例10失败: {e}")  except Exception as e:  print(f"测试用例10错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 10: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1805010220", "168884723", "账号或密码错误")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_011

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_011 "提供明确的错误提示，当账号或密码不正确时; 避免透露具体的错误原因（即不区分是账号错误还是密码错误）;" TSGLXT-XQ-0003 密码错误：输入正确的账号但错误的密码，检查是否显示通用错误提示; TSGLXT-CSD-0011 提供明确的错误提示，当账号或密码不正确时; 测试系统：Windows 20，浏览器：Chrome 最新版本 避免透露具体的错误原因（即不分区是账号错误还是密码错误） 输入错误的密码 检查是否显示通用错误提示; 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性。 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例11通过")  except AssertionError as e:  print(f"测试用例11失败: {e}")  except Exception as e:  print(f"测试用例11错误: {e}")  finally:  # 尝试关闭弹窗  try:  # 等待确定按钮出现  confirm\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.XPATH, "/html/body/div[5]"))  )  confirm\_button.click()  time.sleep(1)  except Exception as e:  # 如果没有找到确定按钮，忽略异常  print(f"Error closing popup in test case 11: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']"))))  run\_test\_case(driver, "178245867", "72165467321", "账号或密码错误")  driver.switch\_to.default\_content()  driver.quit() |

测试用例编号：TSGL\_CSYL\_012

|  |  |
| --- | --- |
| User | TSGL\_CSYL\_012 "提供明确的错误提示，当账号或密码不正确时; 避免透露具体的错误原因（即不区分是账号错误还是密码错误）;" TSGLXT-XQ-0003 账号和密码都错误：输入错误的账号和密码，检查是否显示通用错误提示 TSGLXT-CSD-0012 提供明确的错误提示，当账号或密码不正确时; 测试系统：Windows 21，浏览器：Chrome 最新版本 避免透露具体的错误原因（即不分区是账号错误还是密码错误） 输入错误的账号和密码 显示通用错误提示; 如果提示信息与预期结果一致，则认为该测试用例通过；如果提示信息与预期结果不一致，则认为该测试用例失败; 无 这是图书管理系统的测试用例，生成selenium脚本 |
| LLM | 在使用 Selenium 之前，你需要确保已经安装了 Selenium 库，并且下载了对应浏览器的 WebDriver。WebDriver 是一个浏览器的驱动程序，它允许 Selenium 与浏览器进行交互。  请注意，这只是一个基础示例，实际使用中你可能需要处理更复杂的逻辑，比如等待页面加载、处理弹窗、使用框架切换等。此外，Selenium 还支持远程WebDriver，可以跨浏览器和平台进行测试。根据您提供的测试用例，我将为每个用例编写一个基本的 Selenium 测试脚本。请注意，这些脚本假设您已经有一个运行的图书管理系统，并且已经设置了相应的测试环境（如数据库、后端服务等）。此外，这些脚本是基于 Python 语言编写的。请注意，您需要根据实际情况替换脚本中的 URL、元素定位器（如 name 属性、XPath 等）以及测试用例数据。此外，这个脚本只是一个基础的示例，实际的测试脚本可能需要更复杂的逻辑来处理各种测试场景，比如等待页面加载、处理弹窗、使用框架切换等。  每个测试用例的逻辑都需要根据实际的测试需求来编写，包括但不限于输入数据、点击按钮、验证提示信息等。您可能还需要添加异常处理来确保脚本的健壮性 |
| User | 附：这是图书管理系统的网址：http://120.26.37.204:8089/library 这是 这是图书管理系统的登录页面URL：http://120.26.37.204:8089/library/reader/04readerFrame.jsp 这是账号输入框：<input type="text" name="user" class="from-control my\_input" placeholder="请输入账号" required="required"> 这是密码输入框：<input type="password" name="psw" class="from-control my\_input" placeholder="请输入密码" required="required"> 这是提交按钮"登录"：<input type="submit" class="form-control" value="登录" onclick="javascript:void(0);"> 这是错误提示元素的XPath：/html/body/div[5] |
| LLM | from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  def run\_test\_case(driver, username, password, expected\_error\_message):  try:  # 清空输入框  username\_input.clear()  password\_input.clear()  time.sleep(1)  # 输入账号和密码  username\_input.send\_keys(username)  password\_input.send\_keys(password)  time.sleep(1)  # 点击提交按钮  submit\_button.click()  time.sleep(1)  # 等待错误消息元素加载完成  if expected\_error\_message:  error\_message = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, "/html/body/div[5]"))  )  assert expected\_error\_message in error\_message.text  else:  # 如果没有预期错误消息，检查页面是否成功跳转  success\_page = WebDriverWait(driver, 10).until(  EC.url\_contains("success\_page\_url") # 替换为实际的成功登录后的页面URL  )  assert success\_page  print("测试用例12通过")  except AssertionError as e:  print(f"测试用例12失败: {e}")  except Exception as e:  print(f"测试用例12错误: {e}")  if \_\_name\_\_ == "\_\_main\_\_":  driver = webdriver.Chrome()  driver.maximize\_window()  driver.get("http://120.26.37.204:8089/library/reader/04readerFrame.jsp&#34;)")  driver.switch\_to.frame("view\_frame")  # 等待账号输入框元素加载完成  username\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "user")))  # 等待密码输入框元素加载完成  password\_input = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.NAME, "psw")))  # 等待提交按钮元素加载完成  submit\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@value='登录']")))  run\_test\_case(driver, "1805010220", "0123456", None)  driver.switch\_to.default\_content()  driver.quit() |