测试用例编号：WYYYY-CSYL-0001

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,输入手机号对话框id是com.netease.cloudmusic:id/cellphone，手机号是159896666，勾选“我已阅读并同意”复选框id为com.netease.cloudmusic:id/agreeCheckbox，点击“验证码登录”按钮id是com.netease.cloudmusic:id/newLoginBtn |
| LLM | 根据你提供的测试用例和截图，以下是一个简单的Appium自动化测试脚本示例，使用Python语言编写。这个脚本会模拟在登录页面输入一个格式错误的手机号，并检查是否出现了手机号格式错误的提示。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.touch\_action import TouchAction  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 输入格式错误的手机号  incorrect\_phone\_number = "12345"  phone\_input = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/cellphone")  phone\_input.send\_keys(incorrect\_phone\_number)  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击“验证码登录”按钮  login\_button = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/newLoginBtn")  login\_button.click()  # 等待并验证错误提示是否出现  time.sleep(2)  error\_message = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/error\_message").text  assert "请输入11位数字的手机号" in error\_message, "手机号格式错误提示未出现"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，错误提示的ID（`com.netease.cloudmusic:id/error\_message`）可能需要根据实际应用的布局进行调整。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0007

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,输入手机号对话框id是com.netease.cloudmusic:id/cellphone，手机号是159896666，不勾选“我已阅读并同意”复选框，点击“验证码登录”按钮id是com.netease.cloudmusic:id/newLoginBtn，系统阻止并弹出服务协议和隐私政策等指引 |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟在登录页面不勾选“我已阅读并同意”复选框，然后尝试点击“验证码登录”按钮，验证系统是否会阻止登录并弹出服务协议和隐私政策的提示。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 输入手机号  phone\_number = "159896666"  phone\_input = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/cellphone")  phone\_input.send\_keys(phone\_number)  # 不勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click() # 如果已经勾选，则点击一次可以取消勾选  # 点击“验证码登录”按钮  login\_button = driver.find\_element(MobileBy.ID, "com.netease.cloudmusic:id/newLoginBtn")  login\_button.click()  # 等待并验证是否弹出服务协议和隐私政策的提示  time.sleep(2)  popup = driver.find\_element(MobileBy.XPATH, "//\*[contains(text(), '服务协议和隐私政策等指引')]")  assert popup is not None, "未弹出服务协议和隐私政策的提示"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，弹出提示的XPATH（`"/\*[contains(text(), '服务协议和隐私政策等指引')]"`）可能需要根据实际应用的布局进行调整。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0008

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,输入手机号对话框id是com.netease.cloudmusic:id/cellphone，手机号是159896666，不勾选“我已阅读并同意”复选框，点击“验证码登录”按钮id是com.netease.cloudmusic:id/newLoginBtn，系统阻止并弹出服务协议和隐私政策等指引，用户点击不同意后，返回登录页面，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟在登录页面不勾选“我已阅读并同意”复选框，然后尝试点击“验证码登录”按钮，验证系统是否会阻止登录并弹出服务协议和隐私政策的提示。如果用户点击不同意，脚本将验证是否返回到登录页面。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 输入手机号  phone\_number = "159896666"  phone\_input = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/cellphone")  phone\_input.send\_keys(phone\_number)  # 不勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  if agree\_checkbox.is\_selected():  agree\_checkbox.click()  # 点击“验证码登录”按钮  login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/newLoginBtn")  login\_button.click()  # 等待并验证是否弹出服务协议和隐私政策的提示  time.sleep(2)  popup = driver.find\_element(AppiumBy.XPATH, "//\*[contains(text(), '服务协议和隐私政策等指引')]")  assert popup is not None, "未弹出服务协议和隐私政策的提示"  # 点击不同意  disagree\_button = driver.find\_element(AppiumBy.XPATH, "//\*[contains(text(), '不同意')]")  disagree\_button.click()  # 验证是否返回到登录页面  time.sleep(2)  login\_page = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/cellphone")  assert login\_page is not None, "未返回到登录页面"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，弹出提示的XPATH和不同意按钮的XPATH可能需要根据实际应用的布局进行调整。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0010

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，进入主页后点击我的//android.widget.TextView[@resource-id="com.netease.cloudmusic:id/desc" and @text="我的"]，再点击本地//android.widget.TextView[@resource-id="com.netease.cloudmusic:id/nameDragonBall" and @text="本地"]，用户能够看到按文件名、艺术家、专辑等属性分类的本地音乐列表即可，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟在登录页面不勾选“我已阅读并同意”复选框，然后尝试点击“验证码登录”按钮，验证系统是否会阻止登录并弹出服务协议和隐私政策的提示。如果用户点击不同意，脚本将验证是否返回到登录页面。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 输入手机号  phone\_number = "159896666"  phone\_input = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/cellphone")  phone\_input.send\_keys(phone\_number)  # 不勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  if agree\_checkbox.is\_selected():  agree\_checkbox.click()  # 点击“验证码登录”按钮  login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/newLoginBtn")  login\_button.click()  # 等待并验证是否弹出服务协议和隐私政策的提示  time.sleep(2)  popup = driver.find\_element(AppiumBy.XPATH, "//\*[contains(text(), '服务协议和隐私政策等指引')]")  assert popup is not None, "未弹出服务协议和隐私政策的提示"  # 点击不同意  disagree\_button = driver.find\_element(AppiumBy.XPATH, "//\*[contains(text(), '不同意')]")  disagree\_button.click()  # 验证是否返回到登录页面  time.sleep(2)  login\_page = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/cellphone")  assert login\_page is not None, "未返回到登录页面"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，弹出提示的XPATH和不同意按钮的XPATH可能需要根据实际应用的布局进行调整。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0014

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击//androidx.recyclerview.widget.RecyclerView[@resource-id="com.netease.cloudmusic:id/mixContainerRecyclerView"]/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup播放音乐，再点击暂停com.netease.cloudmusic:id/minPlayBtn，查看音乐播放是否暂停，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后播放音乐并验证是否能够暂停。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击播放音乐  play\_music\_button = driver.find\_element(AppiumBy.XPATH, "//androidx.recyclerview.widget.RecyclerView[@resource-id='com.netease.cloudmusic:id/mixContainerRecyclerView']/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup")  play\_music\_button.click()  # 等待音乐开始播放  time.sleep(2)  # 点击暂停音乐  pause\_music\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/minPlayBtn")  pause\_music\_button.click()  # 验证音乐是否暂停  time.sleep(2)  assert pause\_music\_button.is\_displayed(), "音乐未暂停"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0016

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击//androidx.recyclerview.widget.RecyclerView[@resource-id="com.netease.cloudmusic:id/mixContainerRecyclerView"]/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup播放音乐，再点击根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击//androidx.recyclerview.widget.RecyclerView[@resource-id="com.netease.cloudmusic:id/mixContainerRecyclerView"]/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup播放音乐，再点击进入播放页面com.netease.cloudmusic:id/minPlayBtn，点击右下三个点//android.widget.RelativeLayout[@resource-id="com.netease.cloudmusic:id/moreButtonContainer1"]/android.widget.FrameLayout，查看用户查看当前播放歌曲的详细信息，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后播放音乐，进入播放页面，点击右下的三个点，查看当前播放歌曲的详细信息。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击播放音乐  play\_music\_button = driver.find\_element(AppiumBy.XPATH, "//androidx.recyclerview.widget.RecyclerView[@resource-id='com.netease.cloudmusic:id/mixContainerRecyclerView']/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup")  play\_music\_button.click()  # 等待音乐开始播放  time.sleep(2)  # 点击进入播放页面  play\_page\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/minPlayBtn")  play\_page\_button.click()  # 点击右下的三个点查看详细信息  more\_info\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.RelativeLayout[@resource-id='com.netease.cloudmusic:id/moreButtonContainer1']/android.widget.FrameLayout")  more\_info\_button.click()  # 验证是否显示歌曲详细信息  song\_info = driver.find\_element(AppiumBy.XPATH, "//\*[contains(text(), '歌曲名：')]")  assert song\_info is not None, "未显示歌曲详细信息"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0036

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击//androidx.recyclerview.widget.RecyclerView[@resource-id="com.netease.cloudmusic:id/mixContainerRecyclerView"]/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup播放音乐，再点击根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击//androidx.recyclerview.widget.RecyclerView[@resource-id="com.netease.cloudmusic:id/mixContainerRecyclerView"]/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup播放音乐，再点击进入播放页面com.netease.cloudmusic:id/minPlayBtn，点击右下三个点//android.widget.RelativeLayout[@resource-id="com.netease.cloudmusic:id/moreButtonContainer1"]/android.widget.FrameLayout，点击定时关闭//android.widget.TextView[@resource-id="com.netease.cloudmusic:id/bs\_list\_title" and @text="定时关闭"]，选择时间为10 //android.widget.TextView[@text="10"]，此时//android.widget.TextView[@text="选择时间"]改变则设定成功，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后播放音乐，进入播放页面，点击右下的三个点，点击“定时关闭”，选择时间为10分钟，并验证设置是否成功。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击播放音乐  play\_music\_button = driver.find\_element(AppiumBy.XPATH, "//androidx.recyclerview.widget.RecyclerView[@resource-id='com.netease.cloudmusic:id/mixContainerRecyclerView']/android.widget.FrameLayout[4]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/androidx.recyclerview.widget.RecyclerView/android.widget.LinearLayout[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup[2]/android.view.ViewGroup")  play\_music\_button.click()  # 等待音乐开始播放  time.sleep(2)  # 点击进入播放页面  play\_page\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/minPlayBtn")  play\_page\_button.click()  # 点击右下的三个点  more\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.RelativeLayout[@resource-id='com.netease.cloudmusic:id/moreButtonContainer1']/android.widget.FrameLayout")  more\_button.click()  # 点击“定时关闭”  timer\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@resource-id='com.netease.cloudmusic:id/bs\_list\_title' and @text='定时关闭']")  timer\_button.click()  # 选择时间为10分钟  timer\_option = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@text='10']")  timer\_option.click()  # 验证设置是否成功  time.sleep(2)  selected\_time = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@text='选择时间']")  assert selected\_time is not None, "定时关闭设置未成功"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0043

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击笔记//android.widget.TextView[@resource-id="com.netease.cloudmusic:id/desc" and @text="笔记"]，对推荐进行点赞com.netease.cloudmusic:id/trackLikeBtn，当属性text在原有基础上+1则成功，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后进入笔记页面，对推荐进行点赞，并验证点赞数是否增加。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击笔记  notes\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@resource-id='com.netease.cloudmusic:id/desc' and @text='笔记']")  notes\_button.click()  # 对推荐进行点赞  like\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/trackLikeBtn")  like\_count\_before = int(like\_button.get\_attribute('text').replace(' ', ''))  like\_button.click()  time.sleep(2)  like\_count\_after = int(like\_button.get\_attribute('text').replace(' ', ''))  # 验证点赞数是否增加  assert like\_count\_after == like\_count\_before + 1, "点赞数未增加"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0094

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击左上角侧边栏com.netease.cloudmusic:id/trackLikeBtn，点击我的消息进入//android.widget.TextView[@text="我的消息"]，成功进入消息列表完整展示，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后点击左上角的侧边栏图标，点击“我的消息”，验证消息列表是否完整展示。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击左上角侧边栏图标  sidebar\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/trackLikeBtn") # 注意：这里的ID可能需要根据实际情况调整  sidebar\_button.click()  # 点击“我的消息”  messages\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@text='我的消息']")  messages\_button.click()  # 验证消息列表是否完整展示  time.sleep(2)  messages\_list = driver.find\_elements(AppiumBy.XPATH, "//android.widget.TextView[contains(@text(), '恭喜你获得')]")  assert len(messages\_list) > 0, "消息列表未完整展示"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0101

|  |  |
| --- | --- |
| User | 根据这个测试用例和实际页面截图，生成对应的测试脚本appium,点击其他登录方式id是com.netease.cloudmusic:id/thirdLoginTextView，勾选“我已阅读并同意”复选框com.netease.cloudmusic:id/agreeCheckbox，点击微信id是com.netease.cloudmusic:id/wx，登录后点击左上角侧边栏com.netease.cloudmusic:id/trackLikeBtn，点击音乐黑名单进入//android.widget.TextView[@text="音乐黑名单"]，成功进入黑名单中的歌手、单曲和分类列表正确展示。  ，定位使用AppiumBy |
| LLM | 根据你提供的测试用例和截图，以下是一个Appium自动化测试脚本的示例，使用Python语言编写。这个脚本会模拟点击“其他登录方式”，勾选“我已阅读并同意”，选择微信登录，然后点击左上角的侧边栏图标，点击“音乐黑名单”，验证黑名单中的歌手、单曲和分类列表是否正确展示。  ```python  from appium import webdriver  from appium.webdriver.common.mobileby import MobileBy  from appium.webdriver.common.appiumby import AppiumBy  import time  # 设置Appium服务器地址和端口  appium\_server = 'http://localhost:4723/wd/hub'  # 设置desired capabilities  desired\_caps = {  'platformName': 'Android',  'deviceName': 'your\_device\_name',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.WelcomeActivity',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True  }  # 初始化webdriver  driver = webdriver.Remote(appium\_server, desired\_caps)  try:  # 等待元素加载  time.sleep(2)  # 点击“其他登录方式”  other\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/thirdLoginTextView")  other\_login\_button.click()  # 勾选“我已阅读并同意”  agree\_checkbox = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/agreeCheckbox")  agree\_checkbox.click()  # 点击微信登录  wx\_login\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/wx")  wx\_login\_button.click()  # 等待微信登录完成  time.sleep(10)  # 点击左上角侧边栏图标  sidebar\_button = driver.find\_element(AppiumBy.ID, "com.netease.cloudmusic:id/trackLikeBtn") # 注意：这里的ID可能需要根据实际情况调整  sidebar\_button.click()  # 点击“音乐黑名单”  blacklist\_button = driver.find\_element(AppiumBy.XPATH, "//android.widget.TextView[@text='音乐黑名单']")  blacklist\_button.click()  # 验证黑名单中的歌手、单曲和分类列表是否正确展示  time.sleep(2)  blacklist\_items = driver.find\_elements(AppiumBy.XPATH, "//android.widget.TextView[contains(@text(), '歌手') or contains(@text(), '单曲') or contains(@text(), '分类')]")  assert len(blacklist\_items) > 0, "黑名单中的歌手、单曲和分类列表未正确展示"  finally:  # 关闭webdriver  driver.quit()  ```  请注意，这个脚本假设你的Appium服务器正在本地运行，并且你的设备已经连接并被Appium识别。你需要将`your\_device\_name`替换为你的实际设备名称。此外，微信登录可能需要手动完成，因为Appium无法处理微信的登录流程。  在运行这个脚本之前，请确保你已经安装了Appium和相应的Python客户端库，并且你的设备已经开启了开发者模式并允许USB调试。如果你使用的是模拟器，确保模拟器已经启动并且可以被Appium连接。 |

测试用例编号：WYYYY-CSYL-0086

|  |  |
| --- | --- |
| User | 很好！作为一名软件测试工程师，现在需要你继续针对网易云音乐的商城中心模块继续编写自动化脚本，我将给你网易云音乐商城中心的测试用例和实际页面截图，测试用例为验证从购物车中移除商品的功能应正常；（注意：需要成功登录网易云之后，点击左上角的工具栏标识，然后点击商城中心方可进入商城中心模块后点击右上方的购物车标识进入购物车管理页面，随机勾选一张唱片，进行移除操作）现需要你仔细分析我给的测试用例和商城中心页面截图生成对应的测试脚本 |
| LLM | 根据您提供的测试用例和网易云音乐商城中心的页面截图，我们将编写一个自动化测试脚本来验证从购物车中移除商品的功能是否正常。这个脚本将使用Appium和Python语言。  首先，我们需要确定一些关键的元素定位器，比如商品的删除按钮和商品的复选框。我们将使用一些假设的定位器。在实际编写脚本时，你需要根据实际的UI元素来调整这些定位器。  以下是一个示例脚本：  python  from appium import webdriverfrom appium.webdriver.common.appiumby import AppiumByimport time  *# 初始化Appium WebDriver*  desired\_caps = {  'platformName': 'Android',  'platformVersion': '5.0',  'deviceName': 'Android Emulator',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.SplashActivity',  'automationName': 'UiAutomator2',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True,}  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  *# 等待网易云音乐启动并进入商城中心*  time.sleep(5)  driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="网易云音乐"]').click()  time.sleep(2)  driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="商城中心"]').click()  time.sleep(5)  *# 点击右上方的购物车标识进入购物车管理页面*  cart\_icon = driver.find\_element(AppiumBy.XPATH, '//android.widget.ImageView[@contentDescription="购物车"]')  cart\_icon.click()  time.sleep(5)  *# 在购物车页面选择一个商品，这里我们选择第一个商品*  first\_product = driver.find\_elements(AppiumBy.CLASS\_NAME, 'android.widget.FrameLayout')[1]  checkbox = first\_product.find\_element(AppiumBy.XPATH, './/android.widget.CheckBox')  delete\_button = first\_product.find\_element(AppiumBy.XPATH, './/android.widget.Button[@text="删除"]')  *# 勾选商品*  checkbox.click()  time.sleep(1)  *# 点击删除按钮*  delete\_button.click()  time.sleep(2)  *# 验证商品是否从购物车中移除*try:  first\_product.find\_element(AppiumBy.XPATH, './/android.widget.TextView')  print("测试失败：商品未从购物车中移除")except Exception as e:  print("测试通过：商品已从购物车中移除")  *# 结束测试，关闭Appium会话*  driver.quit()  请注意，这个脚本中的元素定位器（如XPATH）需要根据实际的UI元素来调整。在实际测试中，你可能需要使用Appium的Inspector工具来获取正确的元素定位器。此外，如果页面结构复杂，可能还需要添加更多的等待时间或异常处理逻辑来确保脚本的稳定性。  这个脚本首先进入网易云音乐的商城中心，然后点击购物车图标进入购物车管理页面。在购物车页面，它选择第一个商品并勾选，然后点击删除按钮。脚本将尝试重新定位被删除的商品，如果找不到该商品，则认为删除成功，脚本将打印“测试通过”消息。如果仍然可以定位到该商品，则认为删除失败，脚本将打印“测试失败”消息。 |
| User | 在上一个对话中，购物车商品的删除，勾选一个商品后，需要点击页面的【管理】按钮，然后才出现删除按钮，点击【删除】按钮后，会弹出一个提示框，再次点击【删除】按钮后，才会删除成功，现在根据我的描述，再次生成自动化脚本 |
| LLM | 根据您提供的新的操作步骤，我们需要更新自动化测试脚本以包括点击“管理”按钮，处理弹出的确认对话框，以及验证商品是否已从购物车中成功移除。以下是更新后的Python脚本，使用Appium进行自动化测试：  python  from appium import webdriverfrom appium.webdriver.common.appiumby import AppiumByimport time  *# 初始化Appium WebDriver*  desired\_caps = {  'platformName': 'Android',  'platformVersion': '5.0',  'deviceName': 'Android Emulator',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.SplashActivity',  'automationName': 'UiAutomator2',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True,}  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  *# 等待网易云音乐启动并进入商城中心*  time.sleep(5)  driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="网易云音乐"]').click()  time.sleep(2)  driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="商城中心"]').click()  time.sleep(5)  *# 点击右上方的购物车标识进入购物车管理页面*  cart\_icon = driver.find\_element(AppiumBy.XPATH, '//android.widget.ImageView[@contentDescription="购物车"]')  cart\_icon.click()  time.sleep(5)  *# 在购物车页面选择一个商品，这里我们选择第一个商品*  first\_product = driver.find\_elements(AppiumBy.CLASS\_NAME, 'android.widget.FrameLayout')[1]  checkbox = first\_product.find\_element(AppiumBy.XPATH, './/android.widget.CheckBox')  checkbox.click()  time.sleep(1)  *# 点击管理按钮，显示删除按钮*  management\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="管理"]')  management\_button.click()  time.sleep(2)  *# 点击删除按钮，弹出确认对话框*  delete\_button = first\_product.find\_element(AppiumBy.XPATH, './/android.widget.Button[@text="删除"]')  delete\_button.click()  time.sleep(2)  *# 确认删除操作*  confirm\_delete = driver.find\_element(AppiumBy.XPATH, '//android.widget.Button[@text="删除"]')  confirm\_delete.click()  time.sleep(2)  *# 验证商品是否从购物车中移除*try:  first\_product.find\_element(AppiumBy.XPATH, './/android.widget.TextView')  print("测试失败：商品未从购物车中移除")except Exception as e:  print("测试通过：商品已从购物车中移除")  *# 结束测试，关闭Appium会话*  driver.quit() 说明：  1. ****初始化和登录****：脚本首先初始化Appium WebDriver，然后登录网易云音乐并进入商城中心。 2. ****进入购物车****：点击购物车图标进入购物车管理页面。 3. ****选择商品****：选择第一个商品并勾选。 4. ****管理操作****：点击“管理”按钮以显示删除按钮。 5. ****删除商品****：点击删除按钮，然后在弹出的确认对话框中再次点击删除。 6. ****验证****：检查商品是否已从购物车中移除。   请确保根据实际的UI元素和结构调整元素定位器。在实际测试中，可能需要使用Appium的Inspector工具来获取正确的元素定位器，并根据需要调整等待时间以确保脚本的稳定性和可靠性。 |

测试用例编号：WYYYY-CSYL-0140

|  |  |
| --- | --- |
| User | 很好！作为一名软件测试工程师，现在需要你继续针对网易云音乐的运势中心模块继续编写自动化脚本，我将给你网易云音乐运势中心模块的测试用例和实际页面截图，测试用例的描述为提供访问运势中心的入口；（注意：需要成功登录网易云之后，点击左上角的工具栏标识，然后点击【趣测】方可进入我的运势页面，若页面显示有综合运势字样，说明页面成功跳转到正确页面，则该测试用例通过）现需要你仔细分析我给的测试用例、运势中心页面截图和我的描述生成对应的测试脚本 |
| LLM | 根据您提供的测试用例描述和截图，我们将编写一个自动化测试脚本来验证网易云音乐的运势中心模块是否提供了明显的访问入口，并且该入口是否能够正确跳转到运势中心页面。这个脚本将使用Appium和Python语言。  以下是一个示例脚本：  python  from appium import webdriverfrom appium.webdriver.common.appiumby import AppiumByimport time  *# 初始化Appium WebDriver*  desired\_caps = {  'platformName': 'Android',  'platformVersion': '5.0',  'deviceName': 'Android Emulator',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.SplashActivity',  'automationName': 'UiAutomator2',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True,}  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  *# 等待网易云音乐启动并登录*  time.sleep(5)  *# 点击左上角的工具栏标识*  menu\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="网易云音乐"]')  menu\_button.click()  time.sleep(2)  *# 点击【趣测】进入我的运势页面*  fun\_test\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="趣测"]')  fun\_test\_button.click()  time.sleep(5)  *# 检查页面是否显示有综合运势字样*if driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="综合运势"]'):  print("测试通过：页面成功跳转到正确的运势中心页面")else:  print("测试失败：页面未显示综合运势字样")  *# 结束测试，关闭Appium会话*  driver.quit() 注意事项：  1. ****元素定位器****：脚本中的元素定位器（如XPATH）需要根据实际的UI元素来调整。在实际测试中，你可能需要使用Appium的Inspector工具来获取正确的元素定位器。 2. ****等待时间****：脚本中的等待时间可能需要根据实际的网络速度和设备性能进行调整。   这个脚本首先进入网易云音乐的首页，然后点击“趣测”进入我的运势页面。在运势中心页面，脚本检查是否显示了“综合运势”字样。如果显示，则认为测试通过，页面成功跳转到正确的运势中心页面。如果没有显示，则认为测试失败。 |

测试用例编号：WYYYY-CSYL-0116

|  |  |
| --- | --- |
| User | 很好！作为一名软件测试工程师，现在需要你继续针对网易云音乐的音乐应用模块继续编写自动化脚本，我将给你网易云音乐音乐应用模块的测试用例和实际页面截图，测试用例的描述为可以对音乐应用进行编辑；（注意：需要成功登录网易云之后，点击右下方【我的】按钮，点击【音乐应用】标识进入音乐应用页面，然后点击【编辑】按钮，选择一个一个应用点击该应用上方的叉，点击叉后再点击【完成】，如果该应用从音乐页面消失，说明应用删除成功，测试用例通过）现需要你仔细分析我给的测试用例、音乐应用页面截图和我的描述生成对应的测试脚本。 |
| LLM | 根据您提供的测试用例描述和截图，我们将编写一个自动化测试脚本来验证网易云音乐的音乐应用模块的编辑功能是否正常工作。这个脚本将使用Appium和Python语言。  以下是一个示例脚本：  python  from appium import webdriverfrom appium.webdriver.common.appiumby import AppiumByimport time  *# 初始化Appium WebDriver*  desired\_caps = {  'platformName': 'Android',  'platformVersion': '5.0',  'deviceName': 'Android Emulator',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.SplashActivity',  'automationName': 'UiAutomator2',  'noReset': True,  'unicodeKeyboard': True,  'resetKeyboard': True,}  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  *# 等待网易云音乐启动并登录*  time.sleep(5)  *# 点击右下角的【我的】按钮*  my\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="我的"]')  my\_button.click()  time.sleep(2)  *# 点击【音乐应用】标识进入音乐应用页面*  music\_apps\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.TextView[@text="音乐应用"]')  music\_apps\_button.click()  time.sleep(5)  *# 点击【编辑】按钮*  edit\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.Button[@text="编辑"]')  edit\_button.click()  time.sleep(2)  *# 选择一个应用，这里我们选择第一个应用# 假设第一个应用的删除按钮的定位器*  first\_app = driver.find\_elements(AppiumBy.CLASS\_NAME, 'android.widget.FrameLayout')[0]  delete\_button = first\_app.find\_element(AppiumBy.XPATH, './/android.widget.ImageButton')  delete\_button.click()  time.sleep(1)  *# 点击【完成】按钮*  complete\_button = driver.find\_element(AppiumBy.XPATH, '//android.widget.Button[@text="完成"]')  complete\_button.click()  time.sleep(2)  *# 验证该应用是否从音乐应用页面消失*try:  first\_app.find\_element(AppiumBy.XPATH, './/android.widget.TextView')  print("测试失败：应用未从音乐应用页面消失")except Exception as e:  print("测试通过：应用已从音乐应用页面消失")  *# 结束测试，关闭Appium会话*  driver.quit() 注意事项：  1. ****元素定位器****：脚本中的元素定位器（如XPATH）需要根据实际的UI元素来调整。在实际测试中，你可能需要使用Appium的Inspector工具来获取正确的元素定位器。 2. ****等待时间****：脚本中的等待时间可能需要根据实际的网络速度和设备性能进行调整。 3. ****应用选择****：脚本假设选择第一个应用进行删除，这需要根据实际的应用列表进行调整。   这个脚本首先进入网易云音乐的“我的”页面，然后点击“音乐应用”进入音乐应用页面。在音乐应用页面，脚本点击“编辑”按钮，选择第一个应用并点击该应用的删除按钮，然后点击“完成”按钮。脚本将尝试重新定位被删除的应用，如果找不到该应用，则认为删除成功，脚本将打印“测试通过”消息。如果仍然可以定位到该应用，则认为删除失败，脚本将打印“测试失败”消息。 |

**测试脚本-1**

|  |  |
| --- | --- |
| User | 用例编号 关联需求 关联需求索引 用例描述 测试点索引 前置条件 环境配置 操作步骤 输入数据 预期结果 评判标准 其他说明  WYYYY-CSYL-0272 允许用户自定义消息通知的设置 WYYYY-XQ-0110 验证用户可以更改通知设置 WYYYY-CSD-0273 用户已进入网易云音乐APP的“消息和隐私设置”页面 "测试系统：  雷电模拟器 v9.0.73.0(64)  Android OS 5.0以上  网易云音乐 v9.1.60" 1. 点击“消息通知设置”；2. 修改通知选项（如铃声、震动等）；3. 点击保存 自定义的通知设置数据 系统保存设置，并显示更新后的通知设置 设置正确保存并生效 无  **阅读以上测试用例** |
| LLM | 这是一个关于验证用户在网易云音乐APP中更改通知设置的测试用例。下面是对该测试用例的总结和分析：  ### 测试用例分析  \*\*用例编号\*\*: WYYYY-CSYL-0272  \*\*关联需求\*\*: 允许用户自定义消息通知的设置 (WYYYY-XQ-0110)  \*\*用例描述\*\*: 验证用户可以更改通知设置  \*\*测试点索引\*\*: WYYYY-CSD-0273  #### 前置条件  - 用户已经进入网易云音乐APP的“消息和隐私设置”页面。  #### 环境配置  - 测试系统: 雷电模拟器 v9.0.73.0 (64位)  - Android OS: 5.0以上  - 网易云音乐版本: v9.1.60  #### 操作步骤  1. 点击“消息通知设置”。  2. 修改通知选项（例如：铃声、震动等）。  3. 点击保存。  #### 输入数据  - 自定义的通知设置数据。  #### 预期结果  - 系统能够保存设置，并显示更新后的通知设置。  #### 评判标准  - 设置能够正确保存并生效。  ### 总结  这个测试用例详细描述了用户在APP中更改通知设置的过程，包含了前置条件、环境配置、操作步骤及预期结果。通过验证设置是否能成功保存并应用，可以确保功能的正常运行，提升用户体验。  如果有特定的方面想要进一步探讨，或者需要修改某些内容，请告诉我！ |
| User | 作为一名软件测评师，请根据该用例设计appium-python脚本，其中：  # appium-python-client 2.10.0  # selenium 4.10  # appium 1.17.1  # 网易云音乐 9.1.60 |
| LLM | 以下是一个基于您提供的用例，使用 Appium 和 Python 编写的测试脚本示例。该脚本用于验证用户在网易云音乐APP中更改通知设置的功能。  ### 前提条件  - 确保已安装 Appium 和相关的 Python 库。  - Appium 服务器正在运行，并且设备已连接。  ### 脚本示例  ```python  from appium import webdriver  from selenium.webdriver.common.by import By  import time  # 配置 Appium 连接  desired\_caps = {  "platformName": "Android",  "platformVersion": "5.0", # 请根据实际设备的 Android 版本修改  "deviceName": "YourDeviceName", # 请替换为实际设备名称  "appPackage": "com.netease.cloudmusic", # 网易云音乐的包名  "appActivity": "com.netease.cloudmusic.activity.WelcomeActivity", # 启动活动  "noReset": True # 不重置应用数据  }  # 初始化驱动  driver = webdriver.Remote("http://localhost:4723/wd/hub", desired\_caps)  try:  # 等待应用加载  time.sleep(5)  # 前往消息和隐私设置页面  driver.find\_element(By.ACCESSIBILITY\_ID, "消息和隐私设置").click() # 假设按钮是通过可访问性ID定位的  time.sleep(2)  # 点击消息通知设置  driver.find\_element(By.ACCESSIBILITY\_ID, "消息通知设置").click()  time.sleep(2)  # 修改通知选项（假设有铃声和震动选项可供修改）  # 选择铃声选项  driver.find\_element(By.ACCESSIBILITY\_ID, "铃声").click()  time.sleep(1)  # 选择一个自定义铃声（假设列表中有一个可选铃声）  driver.find\_element(By.XPATH, "//android.widget.TextView[@text='自定义铃声']").click()  time.sleep(1)  # 开启震动  driver.find\_element(By.ACCESSIBILITY\_ID, "震动").click()  time.sleep(1)  # 点击保存按钮  driver.find\_element(By.ACCESSIBILITY\_ID, "保存").click()  time.sleep(2)  # 验证设置是否保存  # 这里可以通过查找特定的元素或者文本来验证设置是否生效  # 假设有一个元素显示当前的通知设置  current\_setting = driver.find\_element(By.ID, "current\_notification\_setting").text # 请根据实际ID修改  assert "自定义铃声" in current\_setting, "铃声设置未保存"  assert "震动开启" in current\_setting, "震动设置未保存"  print("测试通过：通知设置已成功保存")  except Exception as e:  print(f"测试失败: {e}")  finally:  # 关闭驱动  driver.quit()  ```  ### 脚本说明  1. \*\*环境配置\*\*：修改 `desired\_caps` 中的 `platformVersion` 和 `deviceName` 以匹配您的设备。  2. \*\*操作步骤\*\*：脚本模拟用户操作，包括点击设置、修改铃声和震动选项，然后保存设置。  3. \*\*验证\*\*：通过查找当前设置的元素文本来验证设置是否生效。  4. \*\*异常处理\*\*：捕获任何异常并输出错误信息。  ### 注意事项  - 请根据实际的元素可访问性 ID 和 XPATH 修改脚本中的定位器。  - 确保 Appium 服务器和设备都处于连接状态。  - 测试可能需要适配不同的设备分辨率和操作系统版本。 |

**测试脚本-2**

|  |  |
| --- | --- |
| User | 你了解python-appium |
| LLM | 是的，Python-Appium 是一个用于自动化移动应用程序的工具，它结合了 Appium 和 Python 的功能。Appium 是一个开源的测试自动化框架，可以用于 iOS 和 Android 应用程序，而 Python 则是一个易于学习和使用的编程语言。  使用 Python-Appium，你可以编写脚本来控制移动应用，执行测试用例，并进行功能验证。以下是一些主要特性：  1. \*\*跨平台支持\*\*：可以用于 iOS 和 Android 平台的自动化测试。  2. \*\*多种应用类型\*\*：支持原生应用、混合应用和移动网页。  3. \*\*灵活性\*\*：可以与其他测试框架（如 pytest、unittest）结合使用。  ### 基本步骤  1. \*\*安装 Appium 和依赖项\*\*：  ```bash  pip install Appium-Python-Client  ```  2. \*\*启动 Appium 服务器\*\*：  可以通过命令行启动 Appium 服务器：  ```bash  appium  ```  3. \*\*编写测试脚本\*\*：  你可以使用 `Appium` 提供的 API 编写测试脚本，例如：  ```python  from appium import webdriver  desired\_caps = {  'platformName': 'Android',  'platformVersion': '11',  'deviceName': 'YourDeviceName',  'app': '/path/to/your/app.apk',  }  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  # 示例操作  driver.find\_element\_by\_id('element\_id').click()  ```  4. \*\*运行和调试测试\*\*：通过运行你的脚本，可以启动应用并执行自动化测试。  如果你有特定的问题或者想了解某个方面的详细信息，随时问我！ |
| User | 很好，请阅读以下用例：  用例编号 关联需求 关联需求索引 用例描述 测试点索引 前置条件 环境配置 操作步骤 输入数据 预期结果 评判标准 其他说明  WYYYY-CSYL-0292 在状态栏显示歌词 WYYYY-XQ-0118 验证播放音乐时状态栏是否显示歌词 WYYYY-CSD-0288 用户已打开音乐应用并开始播放音乐 "测试系统：  雷电模拟器 v9.0.73.0(64)  Android OS 5.0以上  网易云音乐 v9.1.60" "1. 打开音乐应用  2. 选择一首歌曲并开始播放  3. 观察状态栏是否显示歌词" 歌曲ID（播放的歌曲） 状态栏应显示当前播放歌曲的歌词，且歌词内容应实时更新 歌词正确显示，实时更新 无 |
| LLM | 这是一个针对音乐应用中状态栏歌词显示功能的测试用例。以下是对用例的分析及其实现建议：  ### 用例分析  - \*\*用例编号\*\*：WYYYY-CSYL-0292  - \*\*关联需求\*\*：在状态栏显示歌词（WYYYY-XQ-0118）  - \*\*描述\*\*：验证在播放音乐时，状态栏是否能正确显示歌词。  - \*\*前置条件\*\*：用户需已打开音乐应用并开始播放音乐。  - \*\*环境配置\*\*：  - 模拟器：雷电模拟器 v9.0.73.0 (64)  - Android OS：5.0及以上  - 应用版本：网易云音乐 v9.1.60  - \*\*操作步骤\*\*：  1. 打开音乐应用  2. 选择一首歌曲并开始播放  3. 观察状态栏是否显示歌词  - \*\*输入数据\*\*：歌曲ID（播放的歌曲）  - \*\*预期结果\*\*：状态栏应显示当前播放歌曲的歌词，并实时更新。  - \*\*评判标准\*\*：歌词正确显示并实时更新。  ### 测试脚本实现建议（Python-Appium）  下面是一个简单的 Python-Appium 脚本示例，用于实现该用例：  ```python  from appium import webdriver  import time  # 配置  desired\_caps = {  'platformName': 'Android',  'platformVersion': '5.0',  'deviceName': 'YourDeviceName',  'appPackage': 'com.netease.cloudmusic',  'appActivity': 'com.netease.cloudmusic.activity.MainActivity',  'noReset': True  }  # 启动驱动  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  try:  # 1. 打开音乐应用  driver.launch\_app()  # 2. 选择一首歌曲并开始播放  song\_id = 'your\_song\_id' # 替换为实际歌曲ID  driver.find\_element\_by\_id(f'com.netease.cloudmusic:id/{song\_id}').click() # 找到并点击歌曲  # 等待歌曲播放  time.sleep(3) # 等待几秒以确保歌曲开始播放  # 3. 观察状态栏是否显示歌词  # 可以使用安卓的 UIAutomator 进行状态栏元素查找  # 示例：假设歌词元素的资源ID是 'com.netease.cloudmusic:id/lyric'  lyrics\_element = driver.find\_element\_by\_id('com.netease.cloudmusic:id/lyric')  displayed\_lyrics = lyrics\_element.text  # 预期结果检查  assert displayed\_lyrics != "", "歌词未显示或更新失败"  print("歌词显示正确，实时更新。")  except Exception as e:  print(f"测试失败：{e}")  finally:  driver.quit()  ```  ### 注意事项  1. \*\*元素定位\*\*：确保在代码中使用正确的元素ID或其他定位方法，可能需要根据应用的实际布局进行调整。  2. \*\*异常处理\*\*：在实际场景中，加入更多的异常处理机制以确保测试的稳定性。  3. \*\*实时更新验证\*\*：可能需要额外的逻辑来验证歌词的实时更新，例如在播放过程中定时检查歌词。  如果需要进一步调整或添加功能，请告诉我！ |
| User | 作为一名软件测评师，请根据该用例设计appium-python脚本，其中：  结合了WebDriverWait和expected\_conditions，引入了TouchAction、UiAutomator2Options，根据元素状态动态等待，提升了脚本的效率和稳定性  ，并在关键操作添加try-catch  appium-python-client 2.10.0  selenium 4.10  appium 1.17.1  网易云音乐 9.1.60 |
| LLM | 好的，这里是一个基于你的需求设计的 Appium 测试脚本，结合了 `WebDriverWait` 和 `expected\_conditions`，引入了 `TouchAction` 和 `UiAutomator2Options`，并在关键操作中添加了 `try-catch` 来提高脚本的效率和稳定性。  ### 测试脚本  ```python  from appium import webdriver  from appium.webdriver.common.touch\_action import TouchAction  from appium.webdriver.extensions.android.uiautomator import UiAutomator2Options  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  # 配置  options = UiAutomator2Options()  options.set\_capability("platformName", "Android")  options.set\_capability("platformVersion", "5.0")  options.set\_capability("deviceName", "YourDeviceName")  options.set\_capability("appPackage", "com.netease.cloudmusic")  options.set\_capability("appActivity", "com.netease.cloudmusic.activity.MainActivity")  options.set\_capability("noReset", True)  driver = webdriver.Remote('http://localhost:4723/wd/hub', options=options)  try:  # 1. 打开音乐应用  driver.launch\_app()    # 2. 选择一首歌曲并开始播放  song\_id = 'your\_song\_id' # 替换为实际歌曲ID  try:  song\_element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, f'com.netease.cloudmusic:id/{song\_id}'))  )  song\_element.click()  except Exception as e:  print(f"选择歌曲失败：{e}")  driver.quit()  exit()  # 等待歌曲播放  time.sleep(3) # 等待几秒以确保歌曲开始播放  # 3. 观察状态栏是否显示歌词  try:  lyrics\_element = WebDriverWait(driver, 10).until(  EC.visibility\_of\_element\_located((By.ID, 'com.netease.cloudmusic:id/lyric'))  )  displayed\_lyrics = lyrics\_element.text    # 预期结果检查  if displayed\_lyrics:  print("歌词显示正确，实时更新。")  else:  print("歌词未显示。")  except Exception as e:  print(f"获取歌词失败：{e}")  except Exception as e:  print(f"测试失败：{e}")  finally:  driver.quit()  ```  ### 说明  1. \*\*引入了 `WebDriverWait` 和 `expected\_conditions`\*\*：  - 在选择歌曲和获取歌词时，使用 `WebDriverWait` 等待元素变得可点击或可见，以提高脚本的稳定性。  2. \*\*使用了 `TouchAction` 和 `UiAutomator2Options`\*\*：  - `UiAutomator2Options` 用于设置所需的能力，确保与 Android 设备的兼容性。  3. \*\*动态等待\*\*：  - 使用 `visibility\_of\_element\_located` 和 `element\_to\_be\_clickable` 进行动态等待，确保在元素可用时再进行操作。  4. \*\*异常处理\*\*：  - 在关键操作中添加 `try-catch` 结构，捕获并处理可能的异常，确保脚本运行稳定。  5. \*\*最后的 `finally` 块\*\*：  - 确保在脚本结束时正确关闭 Appium 驱动。  请根据你的环境和实际应用中的元素 ID 进行必要的调整。如果还有其他需求或问题，随时告诉我！ |

**测试脚本-3**

|  |  |
| --- | --- |
| User | 阅读以下用例：用例编号 关联需求 关联需求索引 用例描述 测试点索引 前置条件 环境配置 操作步骤 输入数据 预期结果 评判标准 其他说明  WYYYY-CSYL-0306 在桌面显示歌词 WYYYY-XQ-0123 验证桌面歌词是否与音乐播放同步显示 WYYYY-CSD-0302 用户已注册并登录音乐播放软件账号，且正在播放音乐 "测试系统：  雷电模拟器 v9.0.73.0(64)  Android OS 5.0以上  网易云音乐 v9.1.60" "1. 打开音乐播放软件  2. 开始播放一首歌曲  3. 观察桌面歌词显示情况" 无需特定输入数据，但需记录音乐播放进度和桌面歌词显示情况 桌面歌词应与音乐播放进度同步显示，无延迟或超前现象 桌面歌词显示的歌词内容应与当前播放的歌曲歌词一致 无 |
| LLM | 根据您提供的用例 WYYYY-CSYL-0306，下面是一个用于验证网易云音乐应用中桌面歌词是否与音乐播放同步显示的 Appium-Python 脚本示例。该脚本结合了 `WebDriverWait` 和 `expected\_conditions`，并在关键操作中添加了 `try-catch` 处理。  ### 用例分析  - \*\*用例编号\*\*：WYYYY-CSYL-0306  - \*\*关联需求\*\*：验证桌面歌词是否与音乐播放同步显示（WYYYY-XQ-0123）  - \*\*前置条件\*\*：用户已注册并登录音乐播放软件账号，且正在播放音乐。  - \*\*环境配置\*\*：  - 模拟器：雷电模拟器 v9.0.73.0 (64)  - Android OS：5.0及以上  - 应用版本：网易云音乐 v9.1.60  - \*\*操作步骤\*\*：  1. 打开音乐播放软件。  2. 开始播放一首歌曲。  3. 观察桌面歌词显示情况。  - \*\*预期结果\*\*：桌面歌词应与音乐播放进度同步显示，无延迟或超前现象。  - \*\*评判标准\*\*：桌面歌词显示的歌词内容应与当前播放的歌曲歌词一致。  ### 测试脚本实现  以下是一个示例脚本：  ```python  from appium import webdriver  from appium.webdriver.extensions.android.uiautomator import UiAutomator2Options  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  # 配置Appium连接  options = UiAutomator2Options()  options.set\_capability("platformName", "Android")  options.set\_capability("platformVersion", "5.0") # 根据实际设备版本调整  options.set\_capability("deviceName", "YourDeviceName") # 替换为实际设备名称  options.set\_capability("appPackage", "com.netease.cloudmusic")  options.set\_capability("appActivity", "com.netease.cloudmusic.activity.MainActivity")  options.set\_capability("noReset", True)  driver = webdriver.Remote('http://localhost:4723/wd/hub', options=options)  try:  # 1. 打开音乐播放软件  driver.launch\_app()  time.sleep(2) # 等待应用加载  # 2. 开始播放一首歌曲  try:  play\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, 'com.netease.cloudmusic:id/play\_button'))  )  play\_button.click()  print("成功开始播放歌曲。")  except Exception as e:  print(f"播放歌曲失败：{e}")  driver.quit()  exit()  # 3. 观察桌面歌词显示情况  # 这里可以设定一个观察的时间段  observation\_duration = 30 # 观察时间30秒  start\_time = time.time()  while time.time() - start\_time < observation\_duration:  try:  # 获取当前歌词  current\_lyric = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.ID, 'com.netease.cloudmusic:id/current\_lyric'))  ).text  # 获取音乐播放进度  progress = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.ID, 'com.netease.cloudmusic:id/music\_progress'))  ).text  print(f"当前歌词: {current\_lyric}, 播放进度: {progress}")  # 在这里可以添加更多的逻辑来验证歌词的同步性，例如获取当前时间戳  # 并与歌词时间戳进行对比，确保歌词与进度相符    time.sleep(2) # 每2秒获取一次状态  except Exception as e:  print(f"获取歌词或进度失败：{e}")  except Exception as e:  print(f"测试失败：{e}")  finally:  driver.quit()  ```  ### 说明  1. \*\*配置部分\*\*：  - 使用 `UiAutomator2Options` 配置 Appium 连接，确保连接的设备、应用包名和活动名称正确。  2. \*\*动态等待\*\*：  - 使用 `WebDriverWait` 和 `expected\_conditions` 进行动态等待，确保在操作前元素可点击或存在。  3. \*\*观察桌面歌词\*\*：  - 在播放期间，通过循环定时获取当前歌词和播放进度，记录和打印信息以供后续验证。  4. \*\*异常处理\*\*：  - 在关键操作中添加 `try-catch`，确保在发生错误时能够友好地处理并输出错误信息。  5. \*\*时间控制\*\*：  - 设置了观察时间，定期检查歌词和进度的同步状态。  请根据您的具体环境调整设备名称、元素 ID 等信息。如果有其他需求或问题，请随时告诉我！ |
| User | 作为一名软件测评师，请根据该用例设计appium-python脚本，其中：  结合了WebDriverWait和expected\_conditions，引入了TouchAction、UiAutomator2Options，根据元素状态动态等待，提升了脚本的效率和稳定性  ，并在关键操作添加try-catch  appium-python-client 2.10.0  selenium 4.10  appium 1.17.1  网易云音乐 9.1.60  要求在开头添加以下代码：  options = UiAutomator2Options().load\_capabilities({  "deviceName": "127.0.0.1:62001",  "platformName": "Android",  "platformVersion": "7.1.2",  "appPackage": "com.netease.cloudmusic",  "appActivity": ".activity.IconChangeDefaultAlias",  #'noReset': True,  "unicodeKeyboard": True,  "resetKeyboard": True,  })  driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', options=options)  print('当前用例：桌面歌词')  size = driver.get\_window\_size()  # 获取屏幕宽度  width = size['width']  # 获取屏幕高度  height = size['height']  print(width)  print(height)  # app起始页（点同意）  try:  sleep(8)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.netease.cloudmusic:id/agree")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("起始页发生错误：", e)  sleep(1)  # app起始页（点允许）  try:  sleep(1)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.android.packageinstaller:id/permission\_allow\_button")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("app起始页发生错误：", e)  # app起始页（点击立即体验）  try:  sleep(6)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.netease.cloudmusic:id/trialT")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("app起始页发生错误：", e)  # app起始页（点击同意）  try:  sleep(1)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.netease.cloudmusic:id/positiveBtn")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("app起始页发生错误：", e)  # 进入首页（点击允许）  try:  sleep(8)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.android.packageinstaller:id/permission\_allow\_button")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("app起始页发生错误：", e)  # 首页-选项  try:  sleep(8)  # 等待元素加载完成  element = driver.find\_element(By.ID, "com.netease.cloudmusic:id/menu\_icon")  # 定位  element.click()  # 对元素进行操作  except Exception as e:  print("首页-选项发生错误：", e)  # 首页-选项-滚动到设置并点击  try:  sleep(5)  # 等待元素加载完成  # driver.tap([(340,739)]) # 原先元素的坐标  x1 = int(width \* 730 / 900)  y1 = int(height \* 1540 / 1600)  x2 = int(width \* 730 / 900)  y2 = int(height \* 416 / 1600)  TouchAction(driver).long\_press(x=x1, y=y1).move\_to(x=x2, y=y2).release().perform()  # 滚动到设置  x3 = int(width \* 676 / 900)  y3 = int(height \* 585 / 1600)  TouchAction(driver).tap(x=x3, y=y3).perform()  # 点击设置  except Exception as e:  print("首页-选项-滚动到设置点击发生错误：", e) |
| LLM | 根据您的要求，下面是一个完整的 Appium-Python 脚本，结合了您提供的代码和用例 WYYYY-CSYL-0306。这个脚本将验证网易云音乐应用中桌面歌词与音乐播放的同步显示，同时结合了 `WebDriverWait`、`expected\_conditions` 和 `TouchAction`，并在关键操作中添加了 `try-catch`。  ```python  from appium import webdriver  from appium.webdriver.common.touch\_action import TouchAction  from appium.webdriver.extensions.android.uiautomator import UiAutomator2Options  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  # 配置Appium连接  options = UiAutomator2Options().load\_capabilities({  "deviceName": "127.0.0.1:62001",  "platformName": "Android",  "platformVersion": "7.1.2",  "appPackage": "com.netease.cloudmusic",  "appActivity": ".activity.IconChangeDefaultAlias",  "unicodeKeyboard": True,  "resetKeyboard": True,  })  driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', options=options)  print('当前用例：桌面歌词')  # 获取屏幕宽度和高度  size = driver.get\_window\_size()  width = size['width']  height = size['height']  print(f"屏幕宽度: {width}, 高度: {height}")  # app起始页（点同意）  try:  time.sleep(8)  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.netease.cloudmusic:id/agree"))  )  element.click()  except Exception as e:  print("起始页发生错误：", e)  time.sleep(1)  # app起始页（点允许）  try:  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.android.packageinstaller:id/permission\_allow\_button"))  )  element.click()  except Exception as e:  print("app起始页发生错误：", e)  # app起始页（点击立即体验）  try:  time.sleep(6)  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.netease.cloudmusic:id/trialT"))  )  element.click()  except Exception as e:  print("app起始页发生错误：", e)  # app起始页（点击同意）  try:  time.sleep(1)  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.netease.cloudmusic:id/positiveBtn"))  )  element.click()  except Exception as e:  print("app起始页发生错误：", e)  # 进入首页（点击允许）  try:  time.sleep(8)  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.android.packageinstaller:id/permission\_allow\_button"))  )  element.click()  except Exception as e:  print("app起始页发生错误：", e)  # 首页-选项  try:  time.sleep(8)  element = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.netease.cloudmusic:id/menu\_icon"))  )  element.click()  except Exception as e:  print("首页-选项发生错误：", e)  # 首页-选项-滚动到设置并点击  try:  time.sleep(5)  x1 = int(width \* 730 / 900)  y1 = int(height \* 1540 / 1600)  x2 = int(width \* 730 / 900)  y2 = int(height \* 416 / 1600)  TouchAction(driver).long\_press(x=x1, y=y1).move\_to(x=x2, y=y2).release().perform()    x3 = int(width \* 676 / 900)  y3 = int(height \* 585 / 1600)  TouchAction(driver).tap(x=x3, y=y3).perform()  except Exception as e:  print("首页-选项-滚动到设置点击发生错误：", e)  # 开始播放一首歌曲  try:  time.sleep(5)  play\_button = WebDriverWait(driver, 10).until(  EC.element\_to\_be\_clickable((By.ID, "com.netease.cloudmusic:id/play\_button"))  )  play\_button.click()  print("成功开始播放歌曲。")  except Exception as e:  print(f"播放歌曲失败：{e}")  # 观察桌面歌词显示情况  try:  observation\_duration = 30 # 观察时间30秒  start\_time = time.time()  while time.time() - start\_time < observation\_duration:  current\_lyric = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.ID, 'com.netease.cloudmusic:id/current\_lyric'))  ).text    progress = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.ID, 'com.netease.cloudmusic:id/music\_progress'))  ).text  print(f"当前歌词: {current\_lyric}, 播放进度: {progress}")  time.sleep(2)  except Exception as e:  print(f"观察歌词显示情况失败：{e}")  finally:  driver.quit()  ```  ### 说明  1. \*\*起始页处理\*\*：使用 `WebDriverWait` 动态等待各个操作元素的可点击状态，增强稳定性。  2. \*\*使用 `TouchAction`\*\*：在滚动和点击设置时使用 `TouchAction`，以适应屏幕尺寸。  3. \*\*观察歌词\*\*：在播放过程中，通过循环定时获取当前歌词和播放进度，确保歌词与音乐播放进度的同步性。  4. \*\*异常处理\*\*：在关键操作中添加 `try-catch` 处理，确保脚本在遇到错误时能够输出清晰的错误信息并继续执行后续步骤。  5. \*\*时间控制\*\*：在必要的地方使用 `sleep` 来给应用加载和响应时间，确保元素能正确出现。  请根据您的实际环境调整元素 ID 和其他细节。如果有其他问题或需求，请随时告诉我！ |