

Задание 1 - Теоретические вопросы

Описание задачи:

Представьте, что вам необходимо протестировать мобильное приложение в условиях плохого интернет-соединения (например, 2G или 3G). Ваша задача — проверить, как приложение ведет себя в этих условиях и убедиться, что основные функции продолжают работать корректно.

Действие:

Опишите пошагово, как бы вы провели тестирование с использованием сниффера трафика (плюсом будет скрины/видео пошаговых действий работы в сниффере трафика).

Для выполнения этого задания я использовала MacOS Big Sur 11.7.10, Charles v4.6.6 и тестовое устройство OnePlus 7 Pro (GM1910), Android 12.

Провести тестирование основных функций приложения (смоук тестирование) в условиях плохого интернет соединения можно двумя способами:

Способ 1

С помощью встроенной в смартфон функциональности. Наглядно можно увидеть на [скринкасте](#).

Способ 2

С помощью сниффера трафика Charles. После установки Charles и сертификатов на устройства, подключаем устройства на одну сеть Wi-Fi, настраиваем ручную прокси-сервер в тестовом устройстве. После этого можем приступить к настройке функции троттлинг (Throttle settings) в самом Charles.

Согласно [открытым источникам](#) скорости связи:

2G — 9,4-384кБит/с;

3G — до 42 мБит/с;

4G — до 1 Гбит/с;

5G — свыше 1 Гбит/с.

Артефакты настройки:

- [Скрин настройки тестового устройства](#);
- [Скрин настройки Charles](#).

Настроив нужную скорость связи, можно приступить к смоук тестированию и контролировать скорость запросов и ответов через Charles.

Описание задачи:

Представьте, что после установки новой версии мобильного приложения на Android, на экране списка клиентов (где отображаются ФИО) пропал весь список.

Действие:

Опишите, как бы вы действовали перед тем, как завести баг.

Необходимо воспроизвести этот тест-кейс еще раз, проанализировать проблему и собрать максимум информации, которая поможет разработчикам исправить ошибку. Ошибка может быть как во фронтенде, так и в бэкенде. В зависимости от этого мы поймем какому именно разработчику передавать баг-репорт. И также причина может быть в устройстве или скорости подключения.

Для уточнения причин предлагаю выполнить несколько проверок:

1. Попробовать воспроизвести тест-кейс на предыдущей версии приложения;
2. Сменить аккаунт в приложении и воспроизвести тест-кейс;
3. Перепроверить версию приложения и убедиться что установлена новая версия;
4. Переустановить приложение и повторить тест-кейс;
5. Перезапустить приложение и повторить проверку;
6. Перезагрузить тестовое устройство и воспроизвести тест-кейс снова;
7. Проверить тест-кейс на другом тестовом устройстве;
8. Воспроизвести проверку на эмуляторе (если нет других устройств под рукой. Лучше тестировать на физическом устройстве конечно);
9. Проверить интернет-подключение, переключившись на другую сеть wi-fi или мобильный интернет и провести проверку с каждым подключением;
10. Включить/выключить режим "Самолет" и повторить проверку;
11. Запустить Charles Proxy и посмотреть, приходят ли запросы к серверу. Если запросов нет – возможно, ошибка в коде запроса. Если запросы есть, но сервер отвечает ошибкой – возможно, проблема на бэке. Если сервер отвечает корректно, но список не отображается – возможно проблема на стороне фронтенда.
12. Подключить тестовое устройство к ПК/ноутбуку и через ADB или Android Studio посмотреть логи. Если в логах есть ERROR, WARN, NullPointerException, то их можно выгрузить для баг-репорта;
13. Если есть документация к API, то дополнительно можно протестировать бэкэнд через Postman. С помощью метода GET отправить запрос с URL на клиентов. Проанализировав ответы от сервера, можно также определить проблему, например:
 - 200 OK + JSON со списком клиентов - сервер работает, проблема во фронтенд мобильного приложения (UI не отображает данные).
 - 200 OK, но пустой массив ([]) - возможно, проблема с логикой бэка (неправильные фильтры, доступы).
 - 401 Unauthorized - проверить токен..
 - 403 Forbidden - у пользователя нет прав.
 - 500 Internal Server Error - ошибка на сервере.
 - 504 Gateway Timeout - сервер не отвечает (проблемы с сетью или бэком).

Собрав все артефакты (скрины, скринкасты, логи) можно оформлять баг-репорт.

Задание 2 - Тест кейс, баг-репорт

Требуется скачать приложение доставки еды (apk файл) и провести исследовательское тестирование функционала.

1) При нахождении багов составить баг-репорт.

2) Составить тест-кейс для тестирования разделов “корзина” и “оформление заказа”.

Важно!

При оформлении заказа в комментарии писать “Тестовый заказ”.

Для выполнения этого задания я использовала тестовое устройство OnePlus 7 Pro (GM1910), Android 12. [Артефакты тестирования по ссылке.](#)

Задание 3. Автотест

Написать автотест на Selenium для авторизации в Facebook. Автотест должен в себя включать статус авторизации (Например: Успешно, не успешно).

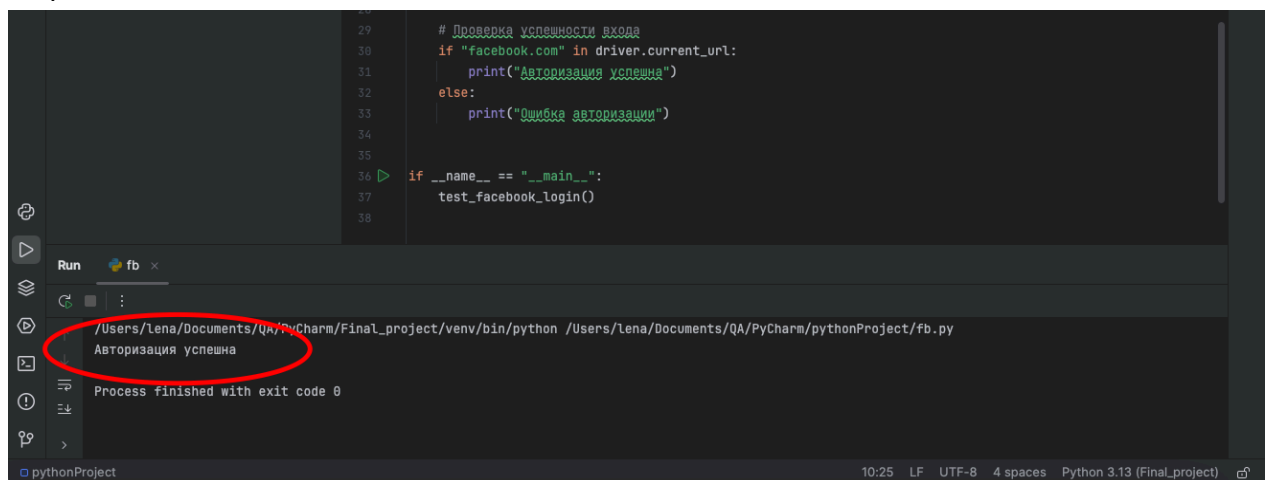
Для автотеста необходим Selenium (*pip install selenium*) и WebDriver для браузера (в моем случае ChromeDriver).

Код fb.py доступен в github.

[Скринкаст автотеста.](#)

*В целях безопасности Facebook использует CAPTCHA для защиты от ботов, поэтому ее приходится добавлять вручную в данном тесте.

Обойти CAPTCHA можно с помощью сторонних платных сервисов, либо с помощью сохраненных Cookies.



```
29 # Проверка успешности входа
30 if "facebook.com" in driver.current_url:
31     print("Авторизация успешна")
32 else:
33     print("Ошибка авторизации")
34
35
36 if __name__ == "__main__":
37     test_facebook_login()
38
```

Run fb x

/Users/Lena/Documents/QA/PyCharm/Final_project/venv/bin/python /Users/Lena/Documents/QA/PyCharm/pythonProject/fb.py

Авторизация успешна

Process finished with exit code 0

pythonProject 10:25 LF UTF-8 4 spaces Python 3.13 (Final_project)