Чек-лист тестирования мобильных приложений

Чек-лист для тестирования мобильных приложений состоит из восьми разделов:

1. Functional Testing - Функциональное тестирование
2. Compatibility testing - Тестирование совместимости
3. Security and Data Privacy - Тестирование безопасности
4. Localization or Internationalization testing - Тестирование локализации и глобализации
5. Usability and User Interface testing - Тестирование удобства использования
6. Stress Testing - Стрессовое тестирование
7. Cross-platform testing - Кросс-платформенное тестирование
8. Performance testing - Тестирование производительности
9. Recovery testing - Тестирование на восстановление
10. Certification testing – тестирование на соответствие сертификатам
11. Резюме

1. Functional Testing - Функциональное тестирование

В данном пункте нам важно убедиться, что наш продукт соответствует нужной функциональной спецификации, упомянутой в документации по разработке

Что проверяем?

1. Установка/удаление/накатка версий
2. Запуск приложения (отображение Splash Screen – *изображение которое появляется при запуске*)
3. Работоспособность основного функционала приложения
   3.1. Авторизация (по номеру телефона/через соц. сети/e-mail)
   3.2. Регистрация (по номеру телефона/через соц. сети/e-mail)
   3.3. Восстановление пароля
   3.4. Онбординг новых пользователей
4. Выход из системы: самостоятельный, по истечению сессии и т.д.
5. Проверки на прерывание чем-то из вне входящим звоноком, смс, push уведомлением, будильник, режим «Не беспокоить», разрядка телефона/подзарядка (звонок, а мы в игрушке, что происходит с приложением когда произошел вызов, в каком виде подгрузилась информация после возврата к игре, не вылетает ли, не сворачивается в трей, не начинает тормозить, позволяет ли это сделать количество оперативной памяти)
6. Валидация обязательных полей (Проверить корректность работы обязательных полей. Убедиться, что обязательные поля отображаются на экране не так, как необязательные)
7. Навигация между разделами приложения

8. Редактирование данных в профиле пользователя
9. Проверка оплаты (Убедиться, что приложение поддерживает платежные операции через системы оплаты Visa, Mastercard, Paypal и др.)
10. Тестирование фильтров (к примеру, не подключены ли взаимо исключаемые фильтры)
11. Корректное отображение ошибок
12. Работа с файлами (отправка/получение/просмотр)
13. Тестирование тайм-аутов (крутится спинер, если ответ не приходит в течение заданного времени 5 сек, то спинер перестает крутится, и выдает то, что прописано в документации)
14. Тестирование заглушек (не соединения с интернетом/нет, например, товаров и т.д пример – например страницы не существует вылазит страница красиво оформленная с сообщением что хэй юзер ты делаешь что-то не то)
15. Тестирование pop-up, алертов (Проверить наличие сообщений об ошибках, например, сообщения «Ошибка сети. Пожалуйста, попробуйте позже» в случае некорректной работы сети.)
16. Тестирование WebView (WebView — это компонент платформы Android, который позволяет встраивать web-страницы в Android-приложения. По сути, это встраиваемый браузер.)
17. Скролл/свайп элементов
18. Тестирование PUSH уведомлений
19. Сворачивание/разворачивание приложения.
20. Разные типы подключений (сотовая связь/Wi-Fi, что при потере связи, изменении типа соединения 3G 4G 5G Wi-Fi)
21. Ориентация экрана (альбомная/портретная)
22. Темная/светлая темы
23. Реклама в приложении

24. Шаринг контента в соц. Сети (Проверить, как функционируют необходимые опции для работы с социальными сетями — Поделиться, Публикация, Навигация.)
25. Убедиться, что установленное приложение не препятствует нормальной работе других приложений и не съедает их память.
26. Работа приложения в фоне

Пагинация страниц (проверка корректно ли листаются страницы, как между ними осуществляется переход вперед назад)

Политики конфиденциальности и прочие ссылки на документы

2. Compatibility testing - Тестирование совместимости

Тестирование совместимости используется, чтобы убедиться, что ваше приложение совместимо с другими версиями ОС, различными оболочками и сторонними сервисами, а также аппаратным обеспечением устройства.
Имеет смысл рассмотреть клиентский уровень конфигурационного тестирования. Сервер у мобильных приложений часто единственный, а клиент устанавливается на большое количество самых разнообразных устройств. Но нужно учесть, что если приложение специфичное — например, игра, для которой выделено несколько серверов, то серверный уровень также становится приоритетным.
Дальше я сосредоточена на тестировании на части клиента.
Лучше всего проводить тестирование совместимости после того как провели функциональное тестирование на одном устройстве.

На клиентском уровне можно выделить:

- Тип устройства: смартфон, планшет и т.д.
- Конфигурация устройства: количество оперативной памяти, тип процессора, разрешение экрана, емкость аккумулятора и т.д.
- Тип и версия операционной системы. iOS 6, 7; Android 4.2.2 и т.д.
- Тип сети: Wi-Fi, GSM.

Перед проведением конфигурационного тестирования рекомендуется
Создавать матрицу покрытия (матрица покрытия — это таблица, в которую заносят все возможные конфигурации).

Не нужно проверять совместимость со 100% устройств. Иначе будет ооочень долго и дорого. Определяются приоритетные девайся и тестируют на них. В плоть до того что закупают на руки нужные девайсы. Чтобы узнать какой девайс нужен используют сайты со статистикой:
Сайты со статистикой по использованию мобильных:
mixpanel.com – сколько процентов девайсов на какой оси сейчас используется.
Statcounter.com – на сколько распространены модели телефонов и производителей
Broserstack.com – какие девайсы покроют потребности в тестировании.

Шаг за шагом, в соответствии с расставленными приоритетами, проверяют каждую конфигурацию. Уже на начальном этапе становится очевидно: чем больше требований к работе приложения при различных конфигурациях рабочих станций, тем больше тестов нам необходимо будет провести. В связи с этим рекомендуется по возможности автоматизировать этот процесс.

Что проверяем?

1. Ensure that the software works properly on the devices with different
    - screen resolutions (720×1280, 1080×1920, 750×1334, 1440×2960, 640×1136, etc.)
    - manufactures (Xiaomi, Samsung, Apple, Huawei, and others)
    - operating systems (Android, IOS, Windows 10 Mobile, Blackberry, Symbian, Palm OS, etc.)
    - RAM (1GB, 2GB, 3GB or more)
    - connectivity options (2G, 3G, 4G, Wi-Fi)
2. Корректное отображение гео (к примеру, можем ли мы подключаться к API телефона и использовать GPS? Который есть там)
3. Информации об операциях (чеки и т.д.)

4. Различные способы оплаты (Google Pay, Apple Pay)
5. Проверить подключение камеры фронтальную и тыльную, широкофрматную и др функции
6. Тестирование датчиков (акселерометр, освещенности, температуры устройства, гироскоп и т.д.)
7. Подключение внешних устройств (карта памяти/наушники и т.д.)
8. Работа с жестами – поддержка удара костяшек пальцев, свайпы и т.д.
9. Тестирование всех функций, к которым открыт доступ приложению на телефоне

Most common compatibility issues:

- alignment problems
- CSS style changes
- broken frames
- distorted tables
- bad navigation
- too big or too small fonts and elements
- functionality issues
- installation and upgrade problems

Основная цель тестирования совместимости - проверить, работает ли программное обеспечение должным образом в конкретной среде. Что ж, этот котик определенно знает, что делает.


COMPATIBILITY

3. Security and Data Privacy - Тестирование безопасности

Данная проверка нацелена на поиск недостатков и пробелов с точки зрения безопасности приложения. Это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Что проверяем?

1. Тестирование разрешений (доступ к камере/микрофону/галерее/и т.д. не используем ли мы какие-то другие функции которые не заявлены в доступе)
2. Данные пользователя (пароли) не передаются в открытом виде
3. В полях, с вводом пароля и подтверждением пароля, данные скрываются астерисками

   Дальше зачитывать я не стану т.к. для понимания нужны примеры и широкая развернутая информация которую приводить очень долго и она специфична
4. Убедиться в том, что данные пользователей приложения — логины, пароли, номера банковских карт — защищены от сетевых атак автоматизированных систем и не могут быть найдены путем подбора.
5. Удостовериться в том, что приложение не дает доступ к секретному контенту или функциональности без надлежащей аутентификации.
6. Убедиться в том, что система безопасности приложения требует надежного пароля и не позволяет взломщику завладеть паролями других пользователей.

7. Убедиться в том, что время таймаута сессии адекватно для приложения.
8. Найти динамические зависимости и принять меры для защиты этих уязвимых участков от взломщиков.
9. Защитить приложение от атак типа SQL-injection.
10. Найти случаи неуправляемого кода и устранить его последствия.
11. Удостовериться в том, что срок действия сертификатов не истек, вне зависимости от того, использует приложение Certificate Pinnig или нет.
12. Защитить приложение и сеть от DoS-аттак.
13. Проанализировать требования хранения и проверки данных.
14. Обеспечить управление сеансами для защиты информации от неавторизованных пользователей.
15. Проверить все криптографические коды и, если необходимо, исправить ошибки.
16. Удостовериться в том, что бизнес-логика приложения защищена и не подвержена атакам извне.
17. Проанализировать взаимодействие файлов системы, выявить и скорректировать уязвимые места.
18. Проверить обработчики протокола (например, не пытаются ли перенастроить целевую страницу по умолчанию, используя вредоносные плавающие фреймы).
19. Защитить приложение от вредоносных атак на клиентов.
20. Защитить систему от вредоносных внедрений в момент работы программы.
21. Предотвратить возможные вредоносные последствия кэширования файлов.
22. Предотвратить ненадежное хранение данных в кэш-памяти клавиатуры устройства.
23. Предотвратить возможные вредоносные действия файлов cookie.
24. Обеспечить регулярный контроль безопасности данных.

25. Изучить пользовательские файлы и предотвратить их возможное вредоносное влияние.
26. Обезопасить систему от случаев переполнения буфера или нарушения целостности информации в памяти.
27. Сделать анализ различных потоков данных и защитить систему от их возможного вредоносного влияния.

4. Localization or Internationalization testing - Тестирование локализации и глобализации

Тестирование интернационализации/глобализации приложения включает тестирование приложения для различных местоположений, форматов дат, чисел и валют, а также замену фактических строк псевдостроками. Тестирование локализации включает тестирование приложения с локализованными строками, изображениями и рабочими процессами для определенного региона.

Что проверяем?

1. Все элементы в приложении переведены на соответствующий язык
2. Тексты зашиты внутри приложения и пользователь в настройках приложения может выставить необходимый язык
3. Тексты зависят от языка в системных настройках
4. Тексты приходят с сервера
5. Корректное отображение форматов дат (ГОД — МЕСЯЦ — ДЕНЬ или ДЕНЬ — МЕСЯЦ — ГОД.)
6. Корректное отображение времени в зависимости от часового пояса

5. Usability and User Interface testing - Тестирование удобства использования

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

- Производительность, эффективность (efficiency) — сколько времени и шагов понадобится пользователю для завершения основных задач приложения, например, размещения новости, регистрации, покупки (чем меньше времени и шагов понадобится пользователю, тем лучше).
- Правильность (accuracy) — сколько ошибок сделал пользователь во время работы с приложением?
- Активизация в памяти (recall) — как долго пользователь помнит о том, как пользоваться приложением, после приостановки работы с ним на длительный период времени? (Повторное выполнение операций после перерыва должно проходить быстрее, чем у нового пользователя).
- Эмоциональная реакция (emotional response) — Как пользователь себя чувствует после завершения задачи: растерян, испытал стресс или, наоборот, ему все понравилось? Порекомендует ли пользователь систему своим друзьям?

Для улучшения удобства использования полезно следовать двум принципам:

1. «Защита от дурака». Если поле предполагает ввод номера телефона, то стоит ограничить диапазон ввода только цифрами и соответствующим образом сформировать клавиатуру. Аналогично для e-mail и остальных элементов, которые предполагают пользовательский ввод данных.
2. Использовать цикл Демминга (планирование-действие-проверка- корректировка), то есть собирать информацию о дизайне и удобстве использования у существующих пользователей, и на основе их мнений планировать изменения в приложении.

Тестирование удобства использования помогает удостовериться в простоте и эффективности использования продукта пользователем, с целью достижения поставленных целей. Иными словами, это не что иное, как тестирование дружелюбности приложения для пользователя.

Юзабилити-тестирование проводится для создания быстрых и простых в обращении приложений. Главная цель — обеспечить удобство пользования приложением, создать интуитивный, соответствующий принятым стандартам интерфейс.
Юзабилити-тестирование обычно проводится на пользователях, поскольку только люди могут понять субъективные ощущения других людей, вызываемые тем или иным приложением.

Что проверяем?

1. Корректное отображение элементов на устройствах с различными разрешениями экранов (Сосисочное тестирование - Убедиться в том, что кнопки имеют нормальный размер и подходят для крупных пальцев.)
2. Текст прост, ясен и виден пользователю. Короткие предложения и абзацы возможно прочитать. (в случае загрузки пользователем больших объемов информации приложение предупреждает о возможных сбоях в его работе из-за этого, контекстуальные меню не перегружены, так как они предполагают быстрое использование). Все шрифты соответствуют требованиям. Все тексты правильно выровнены
3. Все сообщения об ошибках верные, без орфографических и грамматических ошибок
4. Корректные заголовки экранов
5. В поисковых строках присутствуют плейсхолдеры
6. Неактивные элементы отображаются серым
7. Ссылки на документы ведут на соответствующий раздел на сайте
8. Анимация между переходами
9. Корректный возврат на предыдущий экран

10. Поддерживаются основные жесты при работе с сенсорными экранами (swipe back и т.д.)
11. Пиксель-перфект
12. Поместить кнопки в одной области экрана, чтобы не вызвать замешательства у пользователей. Цвет кнопок, выполняющих одну и ту же функцию, совпадает.
13. Системы уменьшения и увеличения масштаба просмотра работает корректно.
14. Убедиться в наличии возможности возврата или отмены действия в случае нажатия не на ту кнопку.
15. Обеспечить пользователя руководством, которое бы помогло ему понять работу приложения и эффективно им пользоваться.
16. Все ли соответствует гайд лайнам


UX

6. Stress Testing - Стрессовое тестирование

Нагрузочное тестирование :
Нагрузочное тестирование — это тип тестирования производительности, который определяет производительность системы, программного продукта или программного приложения в реальных условиях нагрузки.

Стресс-тестирование :
Стресс-тестирование — это тип тестирования программного обеспечения, который проверяет стабильность и надежность системы. Этот тест, в частности, определяет ее устойчивость и устойчивость к ошибкам в условиях чрезвычайно высокой нагрузки.
Эти термины часто используют как синонимы, а также «тестирования производительности», но это не так, так как эти виды тестирования отвечают на разные бизнес-вопросы и используют различную методологию.

Стресс-тестирование является обязательным условием для обнаружения исключений, зависаний и взаимоблокировок, которые могут остаться незамеченными во время функционального тестирования и тестирования пользовательского интерфейса.

Что проверяем?

1. Высокая загрузка центрального процессора (Загрузите в свое приложение как можно больше данных, чтобы попытаться достичь его предела.)
2. Нехватка памяти

3. Загрузка батареи
4. Отказы
5. Низкая пропускная способность сети
6. Большое количество взаимодействий пользователя с приложением (для этого может понадобиться имитация реальных условий состояния сети)

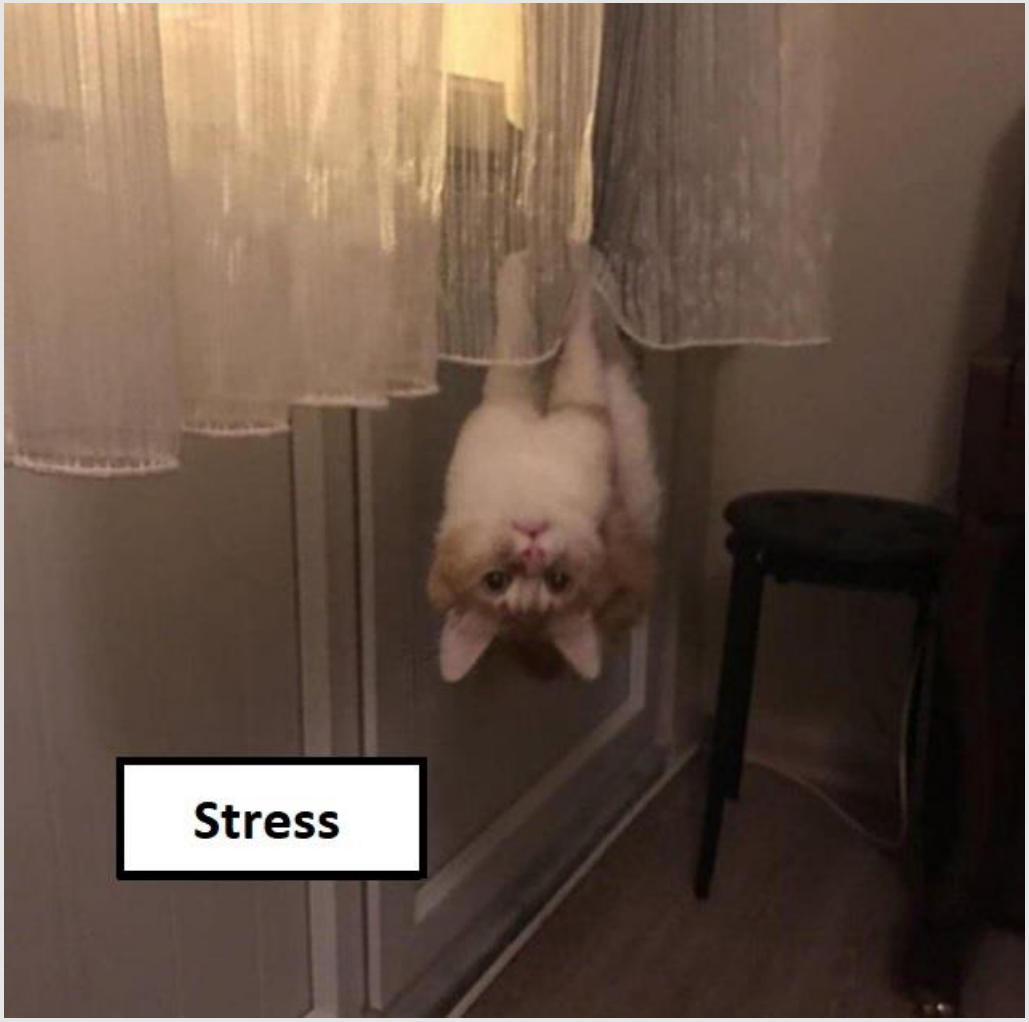
Выполняйте одни и те же операции снова и снова. ***– например обновление виджета приводило к зависанию.***

Выполняйте повторные операции на разных скоростях — очень быстро или очень медленно.

Оставьте приложение работающим в течение длительного периода времени, одновременно взаимодействуя с устройством и просто оставляя его бездействующим, или выполняя некоторую автоматическую задачу, которая занимает много времени, например, слайд-шоу.

На вашем устройстве должно быть запущено несколько приложений, чтобы вы могли часто переключаться между приложением и другими приложениями на устройстве.

Проверка нагрузки - это автоматизированный вид тестирования, моделирующий работу определенного количества пользователей на определенном ресурсе. Нельзя сказать, что здесь много пользователей, но нагрузка однозначно чрезмерная.



7. Cross-platform testing - Кросс-платформенное тестирование

Важный вид тестирования, который необходимо проводить для понимания того, будет ли должным образом отображаться тестируемый продукт на различных платформах, используемых целевой аудиторией.

Что проверяем?

— Работоспособность приложения на различных устройствах разных производителей

8. Performance testing - Тестирование производительности

Если пользователь устанавливает приложение, и оно не отображается достаточно быстро (например, в течение трех секунд), оно может быть удалено в пользу другого приложения. Аспекты потребления времени и ресурсов являются важными факторами успеха для приложения, и для измерения этих аспектов проводится тестирование производительности.

Что проверяем?

1. Время загрузки приложения
2. Обработка запросов (проверяем на сколько быстро работает приложение, отзывчивость)
3. Кэширование данных
4. Потребление ресурсов приложением (например, расход заряда батареи, оперативной памяти)
5. Запуск внутренней памяти с флеш накопителя (актуально для андроид приложений, установка, удаление и т.д. на флеш накопителе)

Вот вам очень мощный котик

9. Recovery testing - Тестирование на восстановление

Кто-то выделяет это тестирование кто-то заносит его в функциональное

Тестирование на восстановление проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи.
Применяется чаще всего в приложениях, которые должны работать 24х7, где каждая минута простоя стоит очень дорого, играх например

1. Проверка восстановления после сбоя системы и сбоя транзакций.
2. Проверка эффективного восстановления приложения после непредвиденных сценариев сбоя.
3. Проверка способности приложения обрабатывать транзакции в условиях сбоя питания (разряженная батарея / некорректное завершение работы приложения).
4. Проверка процесса восстановления данных после перерыва в соединении.

Если выделяют эту ветку тестирования, то сюда же включают и другие важные области проверки:

1. Тестирование установки (быстрая, соответствующая требованиям установка приложения).
2. Тестирование удаления (быстрое, соответствующее требованиям удаление приложения).
3. Сетевые тест-кейсы (проверка адекватной работы сети в разных условиях загрузки, а также способности сети обеспечить функционирование всех приложений, используемых в ходе тестирования).
4. Проверка наличия нефункциональных клавиш.

5. Проверка экрана загрузки приложения.
6. Проверка возможности ввода с клавиатуры во время сбоев сети.
7. Проверка методов запуска приложения.
8. Проверка наличия эффекта зарядки в случае, если приложение находится в фоновом режиме.
9. Проверка функционирования экономичного режима и режима высокой производительности.
10. Выявление последствий извлечения аккумулятора во время работы приложения.
11. Проверка уровня потребления энергии приложением.
12. Проверка побочных эффектов приложения.

10. Certification testing – тестирование на соответствие сертификатам

Примеры:

Android:

- The installation file for the application (.apk) complies with the program policies. (не проверили какие сертификаты запрашивает приложение и использует по факту, и на использование камеры не запросили сертификат, получили по шапке)
- The app meets UIG User Interface Guide requirements.
- There are no viruses in the application. The Android market semi-automatically checks the application for viruses and can block your account if it detects them.
- You must follow the order of version control when publishing an updated version of your application.

IOS:

- The application complies with the requirements of the User Interface Guide.
- The application must have a unique name.
- You need to provide a link for feedback to the developer.
- The application must be placed in a specific category.
- The App Store checks the application for compatibility.
- The application does not contain prohibited materials, unforeseen delays in work or repetition of existing functions.


MAJOR BUG

11. Резюме

Мы ознакомились с универсальной шпаргалкой по тестированию мобильных приложений. Не забывайте читать документацию и дополнять чек-лист проверками, характерными для вашего приложения.