

Как тестировать мобильные приложения. Руководство

01.12.2023

Мобильные технологии и смарт-устройства не стоят на месте. Мы все можем быть уверены в этом, не так ли? Это факт, что мы не можем представить свою жизнь без них. Сегодня мы используем мобильные устройства практически для всего – от покупки продуктов до просмотра фильмов. Тестирование мобильных приложений стало насущной необходимостью. Давайте посмотрим, как это делается.

Введение в тестирование мобильных приложений

Было время, когда компьютеры были в почете. Они изменили то, как мы, люди, думали и учились. Однако сейчас рынок захватили мобильные решения. Люди не хотят включать свои ноутбуки/ПК для решения любых задач, они хотят, чтобы их портативные устройства выполняли все быстро.

Поэтому мобильные решения, которые мы поставляем нашим клиентам, должны быть очень хорошо протестированы. Это руководство¹ предназначено для тех, кто уже занимается мобильным тестированием или тех, кто перешел на него в последнее время.

Этот мануал будет одновременно введением и руководством по мобильному тестированию. Итак, вперед!

Виды мобильного тестирования

В целом существует 2 типа тестирования, которые проводятся на мобильных устройствах:

1. Аппаратное тестирование:

Устройство включает в себя внутренние процессоры, внутреннее оборудование, размеры экрана, разрешение, объем и память, камеру, радио, Bluetooth, WIFI и т.д. Иногда это называют просто “мобильным тестированием”.

2. Тестирование программного обеспечения или приложений:

Приложения, работающие на мобильных устройствах, и их функциональность. Это называется “тестирование мобильных приложений”, чтобы отличить его от предыдущего метода.

В мобильных приложениях есть несколько основных различий, которые важно понимать:

1) Нативные приложения: Нативное приложение создается для использования на таких платформах, как мобильные телефоны и планшеты.

2) Мобильные веб-приложения – это серверные приложения для доступа к сайту/сайтам на мобильных устройствах с помощью различных браузеров, таких как Chrome, Firefox, подключаясь к мобильной сети или беспроводной сети WIFI.

3) Гибридные приложения – это комбинации нативных и веб-приложений. Они работают на устройствах или в автономном режиме и написаны с использованием веб-технологий, таких как HTML5 и CSS.

Есть несколько основных различий, которые отличают их друг от друга:

- Нативные приложения привязаны к одной платформе, в то время как мобильные веб-приложения – к кросс-платформе.
- Нативные приложения пишутся на платформах типа SDK, в то время как мобильные веб-приложения пишутся с использованием веб-технологий, таких как HTML, CSS, asp.net, Java и PHP.
- Для нативного приложения требуется установка, а для мобильных веб-приложений установка не требуется.
- Нативные приложения можно обновлять из Play Store или

App Store, в то время как мобильные веб-приложения обновляются централизованно.

- Многие нативные приложения не требуют подключения к Интернету, но для мобильных веб-приложений оно обязательно.
- Нативные приложения работают быстрее по сравнению с мобильными веб-приложениями.
- Нативные приложения устанавливаются из магазинов приложений, таких как Google Play Store или App Store, в то время как мобильные веб-приложения являются веб-сайтами и доступны только через Интернет.

Остальная часть статьи посвящена тестированию мобильных приложений.

Значение тестирования мобильных приложений

Тестирование приложений на мобильных устройствах является более сложной задачей, чем тестирование веб-приложений на настольных компьютерах, поскольку

- **Различный ассортимент мобильных устройств** с разными размерами экрана и аппаратными конфигурациями, такими как жесткая клавиатура, виртуальная клавиатура (сенсорный экран), трекбол и т. д.
- **Широкий выбор мобильных устройств**, таких как HTC, Samsung, Apple и Nokia.
- **Различные мобильные операционные системы**, такие как Android, Symbian, Windows, Blackberry и IOS.
- **Различные версии операционных систем**, такие как iOS 5.x, iOS 6.x, BB5.x, BB6.x и т. д.
- **Различные операторы мобильных сетей**, например GSM и CDMA.
- Частые обновления – (например, Android- 4.2, 4.3, 4.4, iOS-5.x, 6.x) – с каждым обновлением рекомендуется новый

цикл тестирования, чтобы убедиться, что функциональность приложения не нарушена.

Как и в случае с любым другим приложением, тестирование мобильных приложений также очень важно. Клиентура на определенный продукт обычно исчисляется миллионами, и никому не нужен продукт с ошибками. Это может привести к денежным потерям, юридическим проблемам и непоправимому ущербу для имиджа бренда.

Основные различия между тестированием мобильных и настольных приложений:

Несколько очевидных аспектов, которые отличают тестирование мобильных приложений от тестирования настольных

- На настольном компьютере приложение тестируется на центральном процессоре. На мобильном устройстве приложение тестируется на таких телефонах, как Samsung, Nokia, Apple и HTC.
- Размер экрана мобильного устройства меньше, чем у настольного.
- Мобильные устройства имеют меньший объем памяти, чем настольные компьютеры.
- Мобильные устройства используют такие сетевые подключения, как 2G, 3G, 4G или WIFI, в то время как настольные компьютеры используют широкополосные или коммутируемые соединения.
- Средства автоматизации, используемые для тестирования настольных приложений, могут не работать с мобильными приложениями.

Типы тестирования мобильных приложений:

Чтобы учесть все вышеперечисленные технические аспекты, для мобильных приложений проводятся следующие виды тестирования.

- **Тестирование юзабилити:** Убедиться, что мобильное приложение просто в использовании и обеспечивает удовлетворительный пользовательский опыт для клиентов
- **Тестирование совместимости:** Тестирование приложения на различных мобильных устройствах, браузерах, размерах экрана и версиях ОС в соответствии с требованиями.
- **Тестирование интерфейса:** Тестирование пунктов меню, кнопок, закладок, истории, настроек и навигационного потока приложения.
- **Тестирование сервисов:** Тестирование сервисов приложения в режиме онлайн и офлайн.
- **Низкоуровневое тестирование ресурсов:** Тестирование использования памяти, автоматического удаления временных файлов и роста локальной базы данных, известное как низкоуровневое тестирование ресурсов.
- **Тестирование производительности:** Тестирование производительности приложения путем изменения соединения с 2G, 3G на WIFI, обмена документами, расхода батареи и т.д.
- **Эксплуатационное тестирование:** Тестирование резервного копирования и планов восстановления в случае разряда батареи или потери данных при обновлении приложения из магазина.
- **Тестирование установки:** Проверка работоспособности приложения путем его установки/удаления на устройстве.
- **Тестирование безопасности:** Тестирование приложения для проверки того, защищает ли информационная система данные или нет.

Стратегия тестирования мобильных приложений

Стратегия тестирования должна обеспечивать соблюдение всех требований к качеству и производительности. Несколько советов в этой области:

1) Выбор устройств: Проанализируйте рынок и выберите устройства, которые широко используются. (Это решение в основном зависит от клиента. Заказчик или создатели приложений учитывают фактор популярности определенных устройств, а также маркетинговые потребности приложения, чтобы решить, какие телефоны использовать для тестирования)

2) Эмуляторы: Их использование чрезвычайно полезно на начальных этапах разработки, так как они позволяют быстро и эффективно проверить приложение. Эмулятор – это система, которая переносит программное обеспечение из одной среды в другую без изменения самого программного обеспечения. Она дублирует функции и работу реальной системы.

Типы мобильных эмуляторов

- Эмулятор устройства – предоставляется производителями устройств
- Эмулятор браузера – имитирует среду мобильного браузера.
- Эмулятор операционных систем – Apple предоставляет эмуляторы для iPhone, Microsoft – для телефонов на базе Windows, а Google – для телефонов на базе Android

Рекомендуемый инструмент

Kobiton

Kobiton – это доступная и очень гибкая облачная платформа для мобильного опыта, которая ускоряет тестирование и доставку нативных, веб- и гибридных приложений на Android и iOS с использованием реальных устройств. Новая технология автоматизации тестирования без сценариев помогает командам, не имеющим опыта кодирования, легко генерировать сценарии Appium на основе открытых стандартов.



Список нескольких бесплатных и простых в использовании эмуляторов мобильных устройств

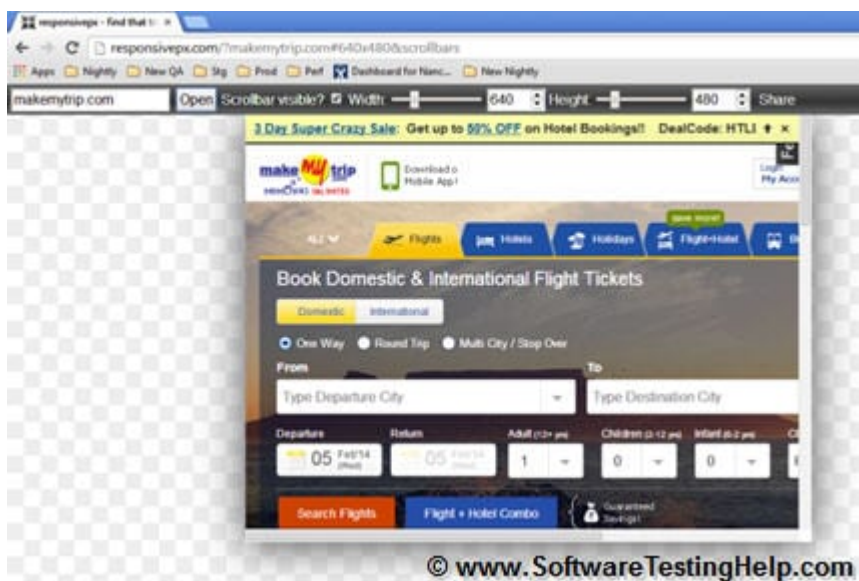
Mobile Phone Emulator: Используется для тестирования таких телефонов, как iPhone, Blackberry, HTC, Samsung и т. д.



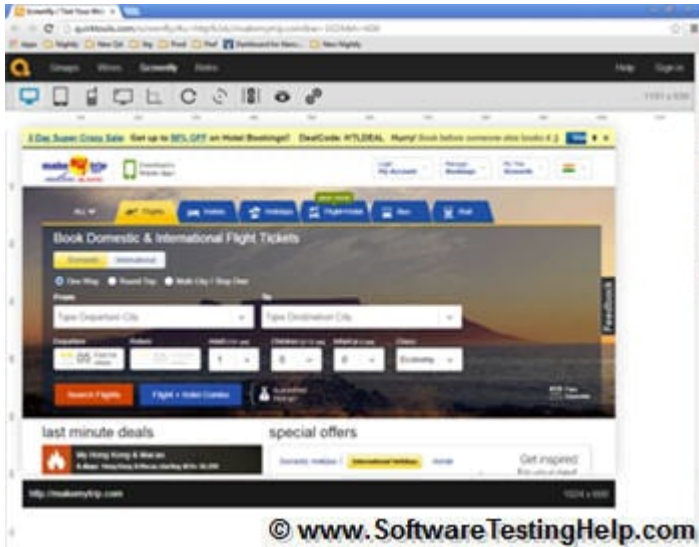
MobiReady: Мы можем не только протестировать веб-приложение, но и проверить код.



Responsiverx: Проверяет отклик веб-страниц, внешний вид и функциональность сайтов.



Screenfly: Это настраиваемый инструмент, используемый для тестирования веб-сайтов по различным категориям.



3) После достижения удовлетворительного уровня разработки мобильного приложения можно перейти к тестированию на **физических устройствах** для более реальных сценариев.

4) **Рассмотрите возможность тестирования на основе облачных вычислений:** Облачные вычисления – это, по сути, работа устройств в нескольких системах или сетях через Интернет, где приложения можно тестировать, обновлять и управлять ими. Для целей тестирования создается веб-среда на симуляторе для доступа к мобильному приложению.



Плюсы:

- Резервное копирование и восстановление – облачные вычисления автоматически создают резервные копии данных

в удаленном месте, что упрощает их восстановление и реставрацию. Кроме того, емкость хранилища неограниченна.

- Доступ к облакам можно получить с разных устройств и в любом месте.
- Облачные вычисления экономически эффективны, просты в использовании, обслуживании и обновлении.
- Быстрое и оперативное развертывание.
- Веб-интерфейс.
- Возможность параллельного запуска одного и того же сценария на нескольких устройствах.

Минусы

- **Меньше контроля:** Поскольку приложение работает в удаленной или сторонней среде, пользователь имеет ограниченный контроль и доступ к функциям.
- **Проблемы с подключением к Интернету:** настройка осуществляется через Интернет. Проблемы с сетью влияют на доступность и функционирование
- **Вопросы безопасности и конфиденциальности:** Облачные вычисления – это вычисления через Интернет, а в Интернете ничто не может быть полностью безопасным, поэтому вероятность взлома данных выше.

5) Автоматизация против ручного тестирования

- Если приложение содержит новую функциональность, тестируйте ее вручную.
- Если приложение требует тестирования один или два раза, делайте это вручную.
- Автоматизируйте сценарии для регрессионных тестов. Если регрессионные тесты повторяются, автоматизированное тестирование идеально подходит для этого.
- Автоматизируйте сценарии для сложных сценариев, которые требуют много времени при ручном выполнении.

Существует два вида инструментов автоматизации для тестирования мобильных приложений:

Объектно-ориентированные инструменты мобильного тестирования – автоматизация путем отображения элементов на экране устройства в объекты. Этот подход не зависит от размера экрана и в основном используется для Android-устройств.

- **Примеры:** Ranorex, Jamo Solutions

Инструменты мобильного тестирования на основе изображений – создание сценариев автоматизации на основе экранных координат элементов.

- **Примеры:** Sikuli, Egg Plant, RoutineBot

6) Настройка сети также является необходимой частью мобильного тестирования. Важно проверить работу приложения в различных сетях, таких как 2G, 3G, 4G или WIFI.

Тест-кейсы для тестирования мобильного приложения

В дополнение к тестовым случаям, основанным на функциональности, тестирование мобильных приложений требует специальных тестовых случаев, которые должны охватывать следующие сценарии.

- **Использование батареи:** Важно следить за расходом батареи при работе приложений на мобильных устройствах.
- **Скорость работы приложения:** время отклика на разных устройствах, при разных параметрах памяти, при разных типах сети и т. д.
- **Требования к данным:** Для установки, а также для того, чтобы проверить, сможет ли пользователь с ограниченным тарифным планом загрузить приложение.

- **Требования к памяти:** опять же, для загрузки, установки и запуска
- **Функциональность приложения:** убедитесь, что приложение не падает из-за сбоев в сети или чего-то еще.

Скачать примеры тестовых примеров для тестирования мобильных приложений:

=> [Скачать примеры тестовых ситуаций для тестирования мобильных приложений](#)

Типичные действия и процедуры при тестировании мобильных приложений

Объем тестирования зависит от количества требований, которые необходимо проверить, или от степени изменений, внесенных в приложение. Если изменений немного, то достаточно будет провести проверку на **вменяемость**. В случае крупных и/или сложных изменений рекомендуется провести **полную регрессию**.

Пример проекта тестирования приложения: ILL (International Learn Lab) – это приложение, разработанное для того, чтобы помочь администраторам и издателям совместно создавать веб-сайты. Используя веб-браузер, преподаватели выбирают из набора функций, чтобы создать класс, отвечающий их требованиям.

Процесс мобильного тестирования:

1) Определите типы тестирования : Поскольку приложение ILL применяется для браузеров, необходимо протестировать его на всех поддерживаемых браузерах с использованием различных мобильных устройств. Нам необходимо провести тестирование **юзабилити, функциональности и совместимости** на разных браузерах с помощью **комбинации ручного и автоматизированного** тестирования.

2) Ручное и автоматизированное тестирование: Методология, используемая в этом проекте, – Agile с итерацией в две недели.

Каждые две недели команда dev. выпускает новую сборку для команды тестирования, и команда тестирования запускает свои тестовые случаи в среде QA. Команда автоматизации создает сценарии для набора базовых функций и запускает сценарии, которые помогают определить, достаточно ли стабильна новая сборка для тестирования. Команда ручного тестирования будет тестировать новую функциональность.

JIRA используется для написания критериев приемки, сопровождения тестовых случаев и регистрации/перепроверки дефектов. По окончании итерации проводится собрание по **планированию итерации**, на котором присутствуют dev. Команда, владелец продукта, бизнес-аналитик и команда QA обсуждают , **что прошло хорошо и что нужно улучшить**.

3) Бета-тестирование: После того как команда QA завершит регрессионное тестирование, сборка переходит в UAT. Приемочное тестирование проводится клиентом. Они повторно проверяют все ошибки, чтобы убедиться, что все ошибки исправлены и приложение работает так, как ожидалось, в каждом утвержденном браузере.

4) Тестирование производительности: Команда тестирования производительности тестирует производительность веб-приложения с помощью скриптов JMeter и при различных нагрузках на приложение.

5) Браузерное тестирование: Веб-приложение тестируется в нескольких браузерах – как с помощью различных инструментов моделирования, так и физически, на реальных мобильных устройствах.

6) План запуска: После каждой 4-й недели тестирование переходит в стадию staging, где проводится финальный раунд сквозного тестирования на этих устройствах, чтобы убедиться, что продукт готов к производству. Затем он запускается в продажу!

Как тестировать мобильные приложения на платформах Android и iOS



Тестировщикам, которые тестируют свои приложения на платформах iOS и Android, очень важно знать разницу между ними. iOS и Android имеют много различий, касающихся внешнего вида, представления приложений, стандартов кодирования, производительности и т. д.

Основные различия между тестированием на Android и iOS

Возможно, вы уже прошли через все учебники, я же привел здесь несколько основных различий, которые, в свою очередь, помогут вам в тестировании:

1) Поскольку на рынке представлено множество Android-устройств, и все они имеют разные разрешения и размеры экрана, это одно из основных отличий.

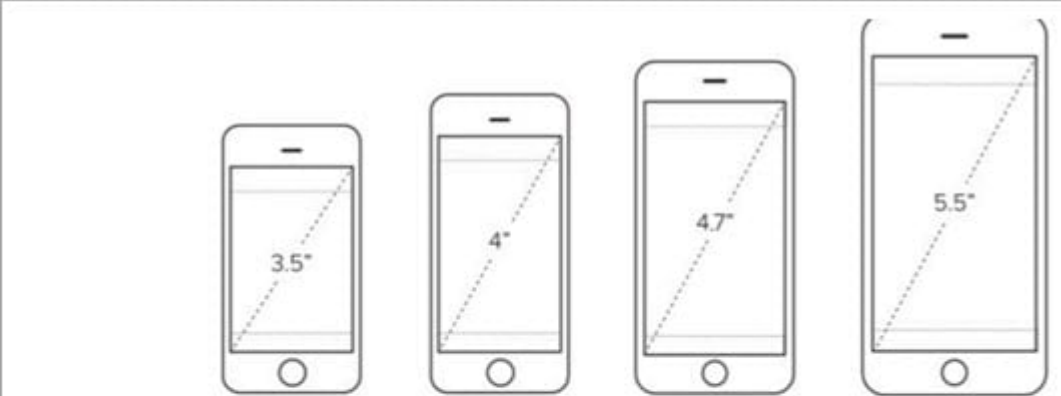
Например, размер экрана Samsung S2 слишком мал по сравнению с Nexus 6. Существует большая вероятность того, что макет и дизайн вашего приложения будет искажен на одном из устройств. Вероятность этого в iOS невелика, так как на рынке доступно

всего несколько устройств, и многие из них имеют схожее разрешение.

Например, до появления iPhone 6 и выше все старые версии имели схожий размер.

2) В качестве примера можно привести тот факт, что в Android разработчики должны использовать изображения 1x, 2x, 3x, 4x и 5x, чтобы поддерживать разрешения изображений для всех устройств, в то время как в iOS используются только 1x, 2x и 3x. Однако ответственность за то, чтобы изображения и другие элементы пользовательского интерфейса корректно отображались на всех устройствах, лежит на тестировщике.

Чтобы понять концепцию разрешения изображений, обратитесь к диаграмме ниже:



	iPhone 4/s	iPhone 5/s/c	iPhone 6	iPhone 6 Plus
Canvas Size (pts)	320 x 480	320 x 568	375 x 667	414 x 736
Screen Size (px)	640 x 960	640 x 1136	750 x 1334	1080 x 1920
Design Canvas (px)	640 x 960	640 x 1136	750 x 1334	1242 x 2208
Screen Ratio	3:2	16:9	16:9	16:9
Asset Scale	@2x	@2x	@2x	@3x

3) Поскольку наш рынок наводнен устройствами Android, код должен быть написан таким образом, чтобы производительность оставалась стабильной. Поэтому вполне вероятно, что ваше приложение будет медленно работать на устройствах с более низкой ценой.

4) Еще одна проблема Android заключается в том, что обновления

программного обеспечения доступны не для всех устройств сразу. Производители устройств сами решают, когда обновлять свои устройства. Это становится очень сложной задачей – протестировать все как на новой, так и на старой ОС.

Кроме того, для разработчиков становится обременительной задачей модифицировать свой код для поддержки обеих версий.

Например, с выходом Android 6.0 произошли значительные изменения: эта ОС начала поддерживать разрешения на уровне приложений. Чтобы уточнить, пользователь также может **изменять разрешения (местоположение, контакты) на уровне приложений.**

Команда тестирования обязана убедиться, что она показывает экран разрешений в приложении, запущенном на Android 6.0 и выше, и не показывает его в более низких версиях.

5) С точки зрения тестирования, тестирование предпроизводственной сборки (т. е. бета-версии) отличается на обеих платформах. В Android, если пользователь добавлен в список бета-пользователей, он может увидеть обновленную бета-версию в Play Store только в том случае, если он вошел в Play Store с тем же идентификатором электронной почты, который был добавлен в качестве бета-пользователя.

Ключевые факторы в мобильном тестировании

Определите свою собственную сферу тестирования

У каждого свой стиль тестирования. Некоторые тестировщики сосредотачиваются только на том, что видят перед собой, в то время как другие увлечены всем, что происходит за кулисами любого мобильного приложения.

Если вы тестируете iOS/Android, я бы посоветовал вам ознакомиться с некоторыми общими ограничениями/основными функциональными возможностями Android или iOS, так как это всегда повышает ценность нашего стиля тестирования. Я знаю, что все это сложно понять без примеров.

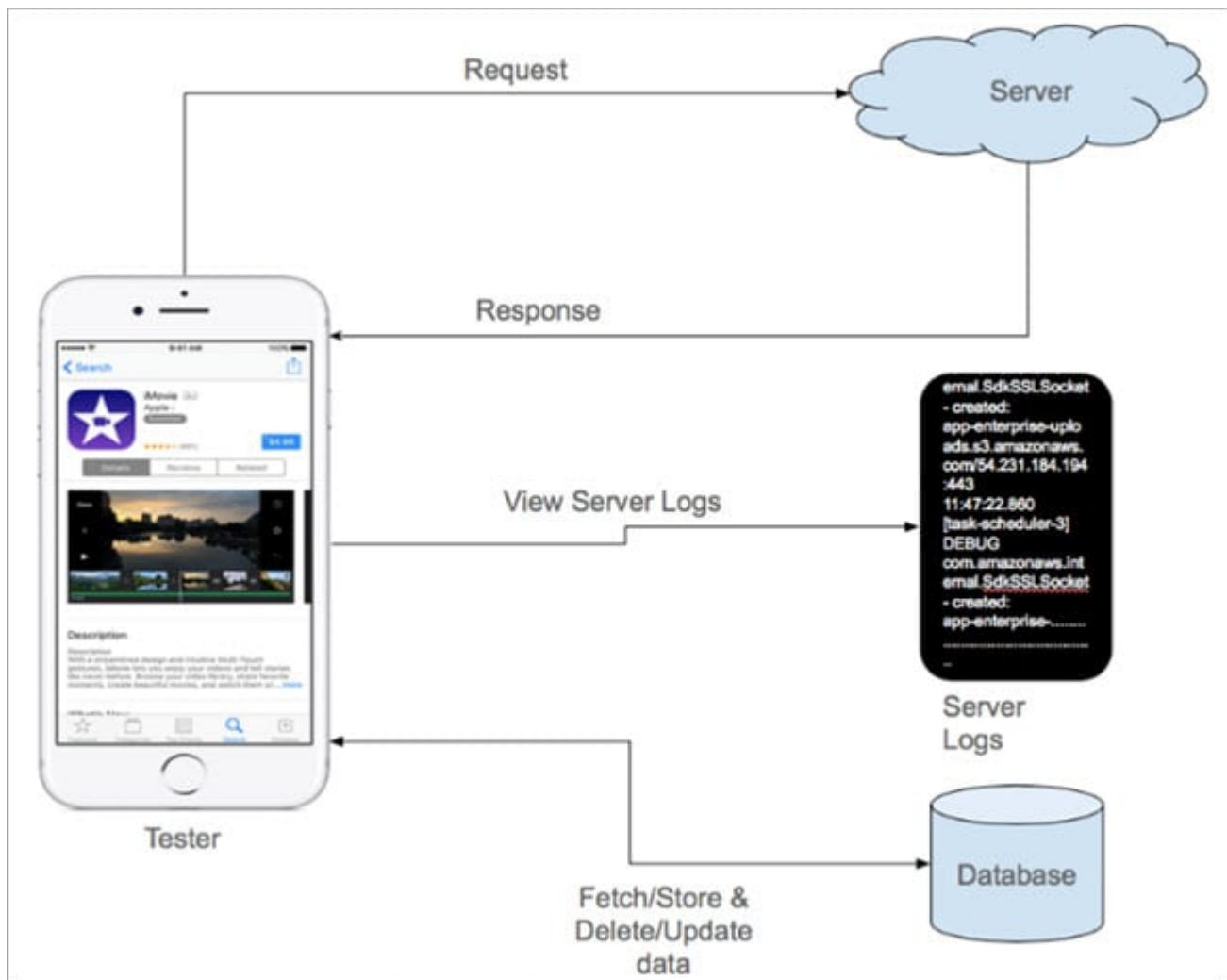
Ниже приведено несколько примеров для вашего удобства:

- Мы не можем изменить такие разрешения, как камера, хранилище и т. д. на уровне приложения на устройствах Android, которые ниже версии 6.0.1.
- Для iOS ниже версии 10.0 набор вызовов не предусмотрен. Проще говоря, набор вызовов используется приложениями для звонков и отображает полноэкранный вид, когда пользователь получает звонок от приложений для звонков, таких как WhatsApp, Skype и т. д. В то время как в версиях iOS ниже 10.0 эти звонки отображаются в виде баннера уведомления.
- Многие из вас могли столкнуться с проблемой в Paytm, когда приложение не перенаправляет вас на платежную страницу банка, если вы хотите пополнить свой кошелек. Мы думаем, что это проблема с нашим банком или сервером Paytm, но просто наш Android System WebView не обновлен. Немного знаний о программировании всегда пригодится вам, чтобы поделиться ими со своей командой.
- Проще говоря, каждый раз, когда приложение открывает какую-либо веб-страницу, Android System WebView должен обновляться.

Не ограничивайте тестирование

Тестирование не должно сводиться только к изучению мобильного приложения и регистрации ошибок. Мы, как QA, должны быть в курсе всех запросов, которые мы отправляем на сервер, и ответов, которые мы получаем от него.

Настройте Putty для просмотра логов или проверьте логи sumo на наличие логов в зависимости от того, что используется в вашем проекте. Это не только поможет вам понять конечный поток приложения, но и сделает вас лучшим тестировщиком, поскольку теперь у вас будет больше идей и сценариев.



Причина: Любое утверждение должно иметь под собой вескую причину. Причина анализа журналов заключается в том, что в журналах наблюдается множество исключений, но они не оказывают никакого влияния на пользовательский интерфейс, поэтому мы их не замечаем.

Так стоит ли игнорировать их?

Нет, не стоит. Это не оказывает никакого влияния на пользовательский интерфейс, но может вызывать опасения в будущем. Если подобные исключения будут возникать постоянно, мы можем столкнуться с аварийным завершением работы нашего приложения. Как мы уже упоминали о App Crash в предыдущем предложении, это приводит к тому, что QA должен иметь доступ к crashlytics проекта.

Crashlytics – это инструмент, в котором регистрируются сбои с указанием времени и модели устройства.

Возникает вопрос: если тестировщик видел, как приложение падает, то зачем ему нужно возиться с crashlytics?

Ответ на этот вопрос довольно интересен. Есть некоторые сбои, которые могут быть не видны в пользовательском интерфейсе, но они регистрируются в crashlytics. Это может быть сбой из-за нехватки памяти или фатальные исключения, которые впоследствии могут повлиять на производительность.

Кросс-платформенное тестирование

Кросс-платформенное тестирование взаимодействия очень важно.

Приведем простой пример: допустим, вы работаете над чат-приложением типа WhatsApp, которое поддерживает отправку изображений и видео, и приложение создано на платформах iOS и Android (разработка может идти синхронно, а может и не идти)

Обязательно проверьте связь Android и iOS. Причина в том, что iOS использует "Objective C", в то время как Android программирует на Java, и из-за того, что они оба построены на разных платформах, иногда необходимо внести дополнительные исправления на стороне приложения, чтобы распознать строки, поступающие с разных языковых платформ.

Следите за размером вашего мобильного приложения

Еще один важный совет для мобильных тестировщиков – проверяйте **размер вашего приложения** после каждого релиза.

Мы должны убедиться, что размер приложения не достигнет той точки, когда даже мы, конечные пользователи, не захотим скачивать это приложение из-за его большого размера.

Тестирование сценариев обновления приложений

Для мобильных тестировщиков **тестирование обновления приложений** очень важно. Убедитесь, что ваше приложение не падает при обновлении, поскольку команда разработчиков могла неправильно указать номер версии.

Сохранение данных также не менее важно: все настройки, которые пользователь сохранил в предыдущей версии, должны сохраниться при обновлении приложения.

Например, пользователь мог сохранить данные своей банковской карты в таких приложениях, как PayPal и т. д.

ОС устройства может не поддерживать приложение

Звучит интересно?

Да, многие устройства могут не поддерживать ваше приложение. Многие из вас наверняка знают, что производители пишут свои собственные обертки поверх iOS, и может оказаться, что любой SQL-запрос вашего приложения несовместим с устройством, поэтому оно выбрасывает исключение, что может привести к невозможности запустить приложение на этом телефоне.

Суть в следующем – попробуйте использовать свое приложение на собственных устройствах, кроме тех, которые вы используете в офисе. Вполне возможно, что вы видите какие-то проблемы с вашим приложением.

Тестирование разрешений приложения

Следующим в списке идет тестирование **разрешений мобильных приложений**. Почти каждое второе приложение запрашивает у пользователей доступ к контактам, камере, галерее, местоположению и т. д. Я видел нескольких тестировщиков, которые совершали ошибку, не проверяя правильные комбинации этих разрешений.

Я могу вспомнить пример, когда мы тестировали приложение для чата, в котором были все возможности для обмена изображениями и аудиофайлами. Разрешение на хранение было установлено на НЕТ.

Теперь, когда пользователь нажимал на опцию “Камера”, она не открывалась, пока разрешение на хранение не было установлено на YES. Этот сценарий был проигнорирован, поскольку в Android

Marshmallow есть такая функция, что если разрешение на хранение установлено на NO, камера не может быть использована для этого приложения.

Сфера применения выходит за рамки того, что мы рассмотрели в предыдущем пункте. Нам нужно убедиться, что приложение не запрашивает разрешения, которые не используются.

Любой конечный пользователь, знакомый с индустрией программного обеспечения, может не скачать приложение, в котором запрашивается слишком много разрешений. Если вы удалили какую-то функцию из своего приложения, убедитесь, что вы удалили экран разрешений для нее.

Сравните похожие и популярные приложения на рынке

Мораль сей истории такова: если вы сомневаетесь, не доводите дело до конца. Сравнение с другими похожими приложениями на той же платформе может укрепить ваши аргументы в пользу того, что тестируемая функциональность будет работать или нет.

Ознакомьтесь с критерием отклонения сборки Apple

Наконец, многие из вас могли получить отказ от Apple на свои сборки. Позвольте мне ознакомить вас с политикой Apple в отношении отклонений.

Будучи тестировщиками, нам сложно уделять внимание техническим аспектам, но все же есть некоторые критерии отклонения, о которых тестировщики могут позаботиться.

Для получения дополнительной информации об этом, пожалуйста, нажмите [здесь](#).

Всегда быть на передовой

Будучи тестировщиком, не позволяйте команде разработчиков/менеджерам перекладывать все на вас. Если вы увлечены тестированием, то **“всегда будьте на переднем крае”**. Старайтесь вовлекать себя в деятельность, которая происходит

задолго до того, как код попадает к вам в ведро для тестирования.

Самое главное – постоянно заглядывайте в JIRA, QC, MTM, или что там используется в вашем проекте, чтобы узнать все последние обновления по тикетам от клиентов и бизнес-аналитиков. Кроме того, будьте готовы поделиться своими соображениями, если вам потребуются изменения. Это относится ко всем тестировщикам, работающим на различных доменах и платформах.

Пока мы не почувствуем, что продукт принадлежит нам, мы никогда не должны давать предложения по улучшению или изменению существующего функционала.

Держите свое приложение в фоновом режиме в течение 12-24 часов

Я знаю, это звучит странно, но за кулисами происходит много всего, чего мы не понимаем.

Я рассказываю об этом, потому что видел, как приложение падало после запуска, скажем, около 14 часов из фонового состояния. Причина может быть любой, в зависимости от того, как разработчики его закодировали.

Вот реальный пример:

В моем случае причиной стало истечение срока действия токена. Одно из чат-приложений, запущенное через 12-14 часов, застряло на баннере подключения и не подключалось до тех пор, пока его не убили и не запустили заново. Такие вещи очень сложно поймать, и в некотором смысле это делает мобильное тестирование более сложным и творческим.

Тестирование производительности вашего приложения

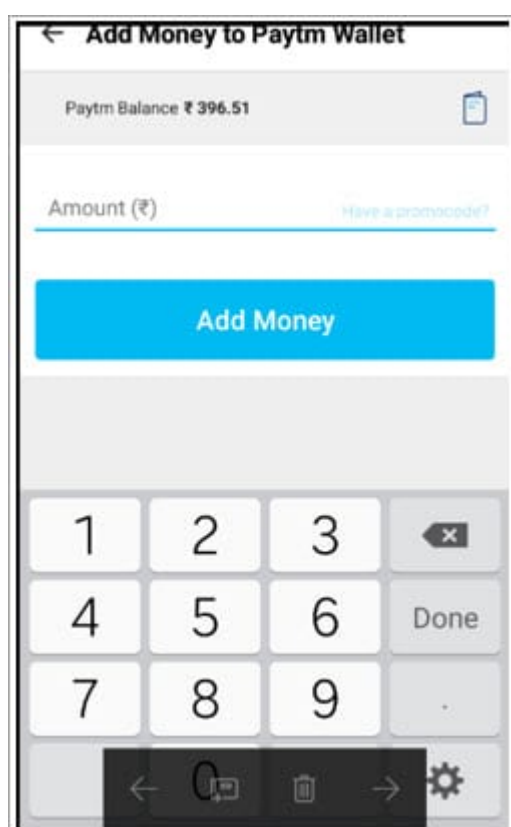
В мобильном мире производительность вашего приложения влияет на степень признания вашего приложения во всем мире. Как команде тестирования, очень важно проверить реакцию вашего

приложения и, что более важно, как оно работает, когда большое количество пользователей используют его в целом.

Пример:

Давайте поговорим о PayTm.

Все вы, наверное, нажимали на опцию ADD MONEY в приложении PayTm, которая затем отображает баланс вашего кошелька. Если рассматривать то, что происходит за кулисами, то это запрос, который отправляется на сервер с идентификатором пользователя PayTm, а сервер отправляет ответ с балансом на вашем счету.



Вышеописанный случай возможен только в том случае, если на сервер обратился один пользователь. Мы должны быть уверены, что даже если на сервер обратится 1000 пользователей, они должны получить ответ вовремя, потому что удобство использования — наша главная цель.

Заключение

В завершение этого урока я бы еще раз подчеркнул, что мобильное тестирование кажется очень простым в начале, но

когда вы вникнете в детали, то поймете, что не так-то просто гарантировать, что все, что было разработано, будет работать гладко на тысячах устройств по всему миру.

В основном вы увидите приложения, которые поддерживаются только на последних и последних версиях ОС. Однако тестировщики должны убедиться, что они не упустят ни одного сценария. Есть много других моментов, которые необходимо принимать во внимание, но они уже описаны в других руководствах.

Такие сценарии, как расход заряда батареи, тестирование на прерывание, тестирование в разных сетях (3G, Wi-Fi), тестирование при переключении сетей, тестирование мобильных приложений на обезьянах и т. д., полезны, когда речь идет о мобильном тестировании.

Отношение тестировщиков имеет большое значение, когда дело доходит до реальной среды тестирования. Если вы не любите то, чем занимаетесь, вы не станете делать то, о чем говорится в учебнике.

Разработка правильной стратегии тестирования, выбор правильных мобильных симуляторов, устройств и инструментов для мобильного тестирования могут обеспечить 100% покрытие тестами и помочь нам включить тесты безопасности, юзабилити, производительности, функциональности и совместимости в наши тестовые наборы.

Таким образом, мы постарались выполнить многочисленные просьбы наших читателей и написать руководство по тестированию мобильных приложений.

Авторы: Спасибо Свапне, Хаснету и многим другим экспертам по мобильному тестированию за помощь в составлении этой серии!