Тестовое задание для QA Engineer

Пункт 1

Тест План

1. **Что надо тестировать?**
   1. описание объекта тестирования: системы, приложения, оборудования
2. **Что будете тестировать?**
   1. список функций и описание тестируемой системы и её компонент в отдельности
3. **Как будете тестировать?**
   1. стратегия тестирования, а именно: виды тестирования и их применение по отношению к объекту тестирования
4. **Когда будете тестировать?**
   1. последовательность проведения работ: подготовка , тестирование, анализ результатов в разрезе запланированных фаз разработки
5. **Критерии начала тестирования**:
   1. готовность тестовой платформы
   2. законченность разработки требуемого функционала
   3. наличие всей необходимой документации
6. **Критерии окончания тестирования**:
   1. результаты тестирования удовлетворяют критериям качества продукта:
      1. требования к количеству открытых багов выполнены
      2. выдержка определенного периода без изменения исходного кода приложения
      3. выдержка определенного периода без открытия новых багов

**Чек-Лист**

1. Тестирование установки и обновления приложения
2. Работа приложения с большими объемами данных
3. Тестирование при включенной функции «Управление одной рукой»
4. Тестирование iOS девайсов при включенной функции «Увеличенный режим экрана»
5. Тестирование установки приложения с нуля
6. Тестирование обновления приложения
7. Тестирования приложения с большим количеством записей в телефонной книге
8. Тестирования приложения с большим количеством фотографий в галерее

**Тест Кейсы**

**Тест-кейс № 1.** Изменение фото  
  
**Шаги**

1. Зайти в мобильное приложение
2. Авторизоваться (логин - wergork, пароль – 12323sadasdQA)
3. Нажать на кнопку " Карандашик”
4. Нажать на кнопку "Заменить фото"

**Ожидаемый результат**  
Откроется новое окно с возможностью изменить фото профиля

**Тест-кейс № 2.** Очистка поля ввода  
  
**Шаги**

1. Зайти в мобильное приложение
2. Авторизоваться (логин - wergork, пароль – 12323sadasdQA)
3. Ввести данные в поле Основная почта”
4. Нажать на кнопку " Крестик”

**Ожидаемый результат**  
Поле ввода “Основная почта” будет очищенно

**Тест-кейс № 3.** Проверка GUI  
  
**Шаги**

1. Зайти в мобильное приложение
2. Авторизоваться (логин - wergork, пароль – 12323sadasdQA)
3. Ввести данные в каждое поле ввода данных
4. Переместить полосу прокрутки несколько раз в разные стороны

**Ожидаемый результат**  
Графический интерфейс останется на своем изначальном месте, а также не потерпит изменений

Пункт 2

**Скриншот №2 “Изменение данных”**

Значок “Мусорки” присутствует только в двух полях ввода данных “Skype”и “Прочее”

что ассиметрично

Для заполнения информации о пользователе достаточно только указать Имя(т.к обозначено \* - обязательное поле), соответственно необходимо добавить \* к другим необходимым полям вроде “Основной телефон”

Увеличить поля ввода данных(в поле “Прочее” некорректно отображается введённая информация )

Поле ввода “Основной телефон” реализовать возможность изменения кода телефона с +7…. на +8….

Кнопка “Заменить фото” названа некорректно, заменить название на “Изменить фото”

**Скриншот №1 “Контакты и социальные сети”**

Не понятно как долго будет присутствовать значок “Новый”

**Bugreport**

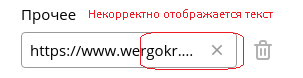
Скриншот №2 “Изменение данных” поле ввода данных“Прочее” при воде данных больше 18 символов , данные отображаются не корректно

Шаги **для воспроизведения**

1. Зайти в мобильное приложение
2. Авторизоваться (логин - wergork, пароль – 12323sadasdQA)
3. Ввести данные больше 18 символов в поле ввода данных “Прочее”

Ожидаемые результат: данные введенные в данное поле отображаются корректно

Фактический результат: данные отображаются не корректно



Пункт 3

Описать процесс тестирования frontend

1. Сравнить результат с макетом
2. Посмотреть, как все выглядит на разных экранах
3. Протестировать на кроссбраузерность
4. Проверить верстку на мобильном устройстве
5. Проверить, как выглядят длинные строки в формах ввода

iloveadaptive — уникальный сервис по разработке и тестированию адаптивной HTML-верcтки под все устройства.

BrowserStack и lambdatest — SAAS-платформа для тестирования frontend на эмуляторах реальных устройств, запущенных в виртуальной машине сервиса.

PerfectPixel — расширение для Chrome, позволяющий накладывать изображение макета поверх сайта, таким образом можно сравнить приближенность верстки к макету.

Think with google — проверяет удобство просмотра на мобильных устройствах, а также скорость работы страницы на мобильных и настольных устройствах.

Screaming Frog — SEO-инструмент, который позволяет находить некорректные и битые ссылки на сайте.

Описать процесс тестирования backend

1. тестирование [API](https://automated-testing.info/tags/api) возвращаемого бекендом  
2. проверка логики web server’a: [работа](https://automated-testing.info/c/rabota) заголовков, проверка HTTP кеширования и т.д  
3. проверка работы background процессов сервера: команды application server’а которые выпоняются периодически (например через cron)  
4. проверка работы любых backend составляющих: состояние кешей в key\val хранилищах, вычитывание очередей из RabbitMQ, и .т.д.

W3C validator — проверка соответствия HTML-кода страниц сайта и CSS-кода на корректность.

Пункт 4

* Знание целевой аудитории
* Правильная фрагментация устройств
* Проведение автоматизированных тестов

**Проблемы iOS и android?**

В двух словах, андроид использует ядро хрома и с этим ничего не сделаешь. iOS использует сафари ядро,плюс анимации переходов для ядра уникальны, если их не перерисовывали,но этим только крупные кампании занимаются.Это для всех браузеров актуально,а также передача цветов разная для мобилок IPS и Oled

Пункт 5

С автоматизацией на практике не сталкивался , соответственно предположить и описать сам процесс автоматизации тестирования страницы не могу , однако после прочтения множества статей , считаю возможным составить перечень framework'ов используемых при автоматизации тестирования. Если стоит задача протестировать небольшое приложение в сжатые сроки, в первую очередь нужно учитывать такие факторы, как тип тестируемого приложения и опыт специалистов. Если тесты пишет разработчик, лучше выбрать родной язык и инструмент для его платформы .Если тестами занимаются специалисты SDET, которые знакомы с другими языками (Java, JavaScript, Python и др.) и работали с Selenium, удобно использовать Appium. Если опытного SDET в команде нет, а тесты будут писать специалисты QA, лучше выбрать платные фреймворки, поскольку в них есть утилиты для записи тестов и более стабильная техподдержка, чем в open source фреймворках(TestComplete,Ranorex и др).