**Теоретическая часть:**

1. Отличие чеклиста от списка тест-кейсов

**Тест-кейс** описывает шаги, условия и параметры, необходимые для проверки тестируемой функции или её части. Состоит из PreConditions,  
Test Case Description, PostConditions.  
Виды по ожидаемому результату:

- Позитивный тест кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию.  
- Негативный тест кейс оперирует как корректными так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.  
**Чек-лист** описывает, что должно быть протестировано. При этом чеклист может быть абсолютно разного уровня детализации. На сколько детальным будет чеклист, зависит от требований к отчетности, уровня знания продукта сотрудниками и сложности продукта.  
Как правило, чеклист содержит только действия (шаги), без ожидаемого результата. Чеклист менее формализован, чем тестовый сценарий. Его уместно использовать тогда, когда тестовые сценарии будут избыточны. Также чеклист ассоциируется с гибкими подходами в тестировании.  
Тест-кейс (что и как тестируется) более подробный, чем чеклист (что, но не как).

1. Тестирование как процесс. Его суть

Спецификации –> тест план –> тест кейсы -> оценка необходимости использования автоматизации как [функционального тестирования](http://www.protesting.ru/automation/functional.html), так и [нагрузочного](http://www.protesting.ru/automation/performance.html) –> дымовое тестирование

if smoke test failed,

приложение дорабатывается

else –

регрессионное и санитарное тестирование

Багтрекер -> перепроверить [дефекты](http://www.protesting.ru/testing/bugreport.html), переведённые в статус Fixed, Rejected, Can't Reproduce и т.д. -> закрытие и переоткрытие дефектов -> централизованное тестирование по тест кейсам и/или) -> "исследование" приложения

if все, что было запланировано, пройдено, есть результаты прогона тест кейсов, [баг репорты](http://www.protesting.ru/testing/bugreport.html), вопросы к аналитикам и заметки -> отчет по проведенному тестированию -> отправка его на проектную группу

от версии к версии происходит примерно то же -> результаты тестирования сходятся с прописанными в плане тестирования критериями окончания тестирования -> передача приложения заказчику

*Суть*

Тест — это деятельность; смысл которой в процессе, а не в артефактах. Цель — выявить статус производимого продукта и оценить, что может угрожать его полезности, так, чтобы клиенты могли принимать обоснованные решения по поводу него; повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах, соответствовать всем описанным требованиям; предоставление актуальной информации о состоянии продукта на данный момент.

При помощи тестирования руководство проекта получает оценку рисков того, что разрабатываемое ПО поведёт себя не так, как ожидается. В итоге тестированием мы боремся не с тем, чтобы ошибок в программе не было вовсе, а оцениваем, насколько и нам, и пользователям будет хуже, если мы выпустим продукт с текущим числом недоработок и багов. тестирование совсем не про качество кода, а про оценивание вероятности получить с этим кодом ту или иную проблему. Опираясь на эту оценку, руководство проекта может более обоснованно принимать решения.

**Тестирование программного обеспечения** вообще есть проверка соответствия между реальным и ожидаемым поведением программы. В более широком смысле, тестирование — это одна из техник контроля качества, включающая в себя мероприятия по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis)

3. Этапы тестирования:

1) Анализ продукта

2) Работа с требованиями

3) Разработка стратегии тестирования и планирование процедур контроля качества

4) Создание тестовой документации

5) Тестирование прототипа

6) Основное тестирование

7) Стабилизация

8) Эксплуатация

4. Что такое ошибка?

***Error*** — ошибка пользователя, то есть он пытается использовать программу не предусмотренным способом.  
Пример — вводит буквы в поля, где требуется вводить цифры (возраст, количество товара и т.п.).  
В качественной программе предусмотрены такие ситуации и выдаются сообщение об ошибке (error message).  
***Bug (defect)*** — ошибка разработчика, в программе что-то идёт не так, как планировалось и она выходит из-под контроля. Например, когда никак не контролируется ввод пользователя, в результате неверные данные вызывают краши или иное в работе программы. Либо внутри программа построена так, что изначально не соответствует тому, что от неё ожидается.

1. Опишите жизненный цикл разработки ПО

• Пре-альфа  
• Альфа  
• Бета  
• Релиз-кандидат  
• Релиз  
• Пост-релиз

1. Какие существуют техники тест-дизайна?

- Эквивалентное Разделение (Equivalence Partitioning - EP)

- Анализ Граничных Значений (Boundary Value Analysis - BVA)

- Причина / Следствие (Cause/Effect - CE)

- Предугадывание ошибки (Error Guessing - EG)

- Исчерпывающее тестирование (Exhaustive Testing - ET). В пределах этой техники необходимо проверить все возможные комбинации входных значений, и в принципе, это должно найти все проблемы. На практике применение этого метода не представляется возможным из-за огромного количества входных значений.

1. Какие отличия тестирования веб- и мобильных приложений?

М**обильные приложения предназначены для большего числа пользователей и более разнообразных устройств, чем веб-приложения.**

В то время как мобильные приложения должны постоянно выполнять «на ходу» вычислительные функции, веб-приложения обычно сосредоточены на стационарных портативных или настольных операциях, с классическими функциональными возможностями мыши, курсора или маршрутизатора Wi-Fi, которые менее задействованы в мобильных приложениях. Когда тестирование web-приложений переходит к тестированию мобильных приложений задача последовательно расширяется. Тестирование мобильных приложений должно касаться нескольких различных функций в отношении нескольких различных сред в нескольких разных условиях.

Увеличение интереса пользователей и их привязанности к мобильным устройствам привело к тому, что QA должны проверять приложения на:

- постоянную связь

- управление уведомлениями

- синхронизацию на нескольких платформах

Таким образом, тестирование программного обеспечения представляет собой задачу координации и обеспечения уровня функциональности мобильных устройств, которая будет учитывать требования и предпочтения клиентов.

*RAM и ограничения хранения*

Многие мобильные устройства по-прежнему поставляются с 1 или 2 ГБ оперативной памяти, а также со сравнительно небольшими 16 ГБ SSD. Это создает серьезные ограничения для оперативной памяти и емкости хранилища для тестирования, особенно в отношении огромного объема памяти и хранилища, которые доступны любому современному веб-браузеру. Кроме того, такие услуги, как рекламные платформы, могут серьезно замедлить работу мобильного браузера, так что перенос веб-приложения на телефон или планшет может вызвать трудности.

*Различные взаимодействия для разных пользователей*

Взаимодействия для компьютеров и ноутбуков стабилизировано более 30 лет - мышь и клавиатура - это стандарт для всего, от работы в Facebook до флэш-игры. Это составляет особенности тестирования веб приложений.

Проблема со стороны мобильных приложений заключается в широком спектре сенсорных действий - прокрутка, вытягивание, защемление + голосовые возможности (Siri и Google Now). Специфические для конкретного устройства инновации, такие как жесты hand wave на некоторых гарнитурах Samsung или новый набор аудио iPhone, добавляют сложности на тестирование ios приложений и Android-приложений.

*Индивидуальные типы приложений*

Десктопное веб-приложение разработано на HTML, CSS и JavaScript с некоторыми вариантами в зависимости от того, какие платформы разработчик хочет использовать. Мобильные приложения не так просты. Они могут быть созданы, как нативные приложения на Java или Objective-C, или как гибридные, которые могут использовать специальные платформы для представления системных API в качестве API-интерфейсов JavaScript, адресованных веб-кодом. Очень важно, чтобы был разработан roadmap для платформы, чтобы управлять испытаниями для всех типов тестирования.

*Поведение с подключением и без него*

Веб-приложение, работающее на Mac или ПК, будет бесполезным без подключения к Интернету. То же самое для мобильного веб-приложения. Однако нативное или гибридное мобильное приложение может обладать автономными функциями. Некоторые игры могут воспроизводиться без подключения, а редакторы документов также могут сохранять функциональность в режиме полета. Необходимо проверять, как мобильные приложения ведут себя при переключении между соединениями или использовании нестабильных сетей.

Это поможет провести тестирование андроид-приложений, а также [iOS](https://artjoker.ua/ru/uslugi/application-development-for-ios/" \t " _blank)- и веб-приложений более качественно.

Ссуществует некоторое совпадение между веб-и мобильными приложениями. Многие мобильные приложения частично относятся к веб-приложениям, и тестировщики придерживаются некоторых общих схем тестирования в рамках гибкого цикла разработки. Наличие программного обеспечения для управления тестированием является ключевым строительным блоком, поскольку вы формируете общую стратегию тестирования и обращаете внимание на учет широкого круга рабочих процессов, ресурсов и возможных проблем, связанных как с мобильными, так и с веб-приложениями.

**Виды тестирований**

Ручное тестирование, как и автоматизированное, играют важную роль для QA. Машина может значительно сэкономить время и облегчить процесс, но не все тестовые процессы обойдутся без участия человека.

Автоматизированное тестирование используется для:

* тестирования модулей и библиотек
* соответствия UI/UX
* API

Тестирование безопасности веб-приложений обеспечивает комфортную работу для клиентов и избавляет от обработки багов после релиза приложений.