Собственно вопросы:

1. **Объясните термин «жизненный цикл программного обеспечения».**
   Жизненным циклом программного обеспечения (SLC) является период времени, начинающийся с момента появления концепции ПО и заканчивающийся тогда, когда использование ПО более невозможно. Жизненный цикл программного обеспечения обычно включает в себя следующие этапы: концепт, описание требований, дизайн, реализация, тестирование, инсталляция и наладка, эксплуатация и поддержка и, иногда, этап вывода из эксплуатации. Данные фазы могут накладываться друг на друга или проводиться итерационно.

2. **Объясните термин «жизненный цикл разработки программного обеспечения».**
   Жизненным циклом разработки программного обеспечения (SDLC) является концепция, которая описывает комплекс мероприятий, выполняемых на каждом этапе (фазе) разработки программного обеспечения.

3. **Объясните преимущество использования модели жизненного цикла разработки ПО (SDLC).**
   - обеспечение основы проекта (методологии, активность...);
   - обеспечение визуализации хода реализации проекта;
   - помощь компании в эффективности и успешного завершения проекта (сокращение затрат, уменьшение сроков разработки и тестирования, повышение качества конечного продукта);
   - уменьшение рисков, связанных с процессом разработки ПО;
   - обеспечение специальным механизмом отслеживания прогресса проекта.

4. **Каковы основные фазы модели жизненного цикла разработки ПО?**
   - Принятия решения (идея) о необходимости создания ПО;
   - Сбор и анализ требований;
   - Дизайн (Системы и ПО) на основе требований;
   - Кодирование на основе дизайна системы;
   - Тестирование;
   - Внедрение в пользовательскую среду;
   - Сопровождение (в том числе фиксация найденных в пользовательской среде ошибок);
   - Изъятие из эксплуатации (редко);

5. **Объясните, что такое Обеспечение качества (Quality Assurance)?**
   Обеспечение качества (QA) — это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации ПО и предпринимаемых на разных стадиях жизненного цикла ПО, для обеспечения требуемого уровня качества выпускаемого продукта.

   Обеспечение качества определено в стандарте ISO 9000:2005 «Системы менеджмента

качества. Основные положения и словарь» как «часть менеджмента качества, направленная на создание уверенности в том, что требования к качеству будут выполнены».

Менеджмент качества в этом же стандарте представлен как «скоординированная деятельность по руководству и управлению организацией применительно к качеству», а в примечании сказано, что он «обычно включает разработку политики и целей в области качества, планирование качества, управление качеством, обеспечение качества и улучшение качества».

6. **Объясните, что такое Контроль качества (Quality Control)?**
   Контроль качества (QC) — это совокупность действий, проводимых над ПО в процессе разработки, для получения информации о его актуальном состоянии в аспектах готовности к выпуску, соответствия зафиксированным требованиям и соответствия заявленному уровню качества этого ПО.

7. **Объясните, что такое тестирование ПО?**
   Тестирование ПО — это процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, и показать, что они подходят для заявленных целей и для определения дефектов.

   Из этого определения становится понятно, что тестирование ПО включает в себя два различных процесса:
   Валидация (validation): доказанное объективными результатами исследования подтверждение того, что требования для конкретного определенного использования приложения были выполнены. [ISO 9000]
   Верификация (verification): доказанное объективными результатами исследования подтверждение того, что определенные требования были выполнены. [ISO 9000]

8. **Какие основные цели тестирования ПО?**
   Цель тестирования (test objective, test target) — это причина или цель разработки и выполнения теста.

   Основные цели:
   - обеспечить очищения ПО от ошибок (Вы не можете предоставить 100% покрытие, но Вы должны сделать все возможное, и гарантировать, что очевидные ошибки исправлены);
   - убедить, что ПО отвечает оригинальным требованиям и спецификации;
   - обеспечить уверенность в ПО (пользователям, заказчикам и т.д.).

9. **Когда следует начинать тестировать ПО?**
   Простой ответ — как только это возможно! Более детально:

- когда тестирование ПО проводится на ранней стадии, вы можете с легкостью повлиять на дизайн, так как его изменение на этой стадии не столь дорогостоящее чем на более поздних стадиях;
- в свою очередь, чем раньше обнаруживается ошибка, тем дешевле она стоит компании;
- также тестирование может начинаться до фактического получения ПО (статическое тестирование), что действительно немаловажно, так как снижает сложность провождения динамической стадии тестирования. Бытует мнение, что многие ошибки, которые найдены в стадии динамического тестирования, могли и должны были зафиксированные в стадии статического тестирования;
- тестирование на ранних стадиях (изучение требований, спецификаций, бизнес случаев и т.д.) обеспечит тестировщику больше знаний о ПО, поможет обнаружить логические и технические ошибки, которые бы влияли на ПО, его конечный дизайн и стоимость.

10.Когда следует заканчивать тестирование ПО?

Простой ответ — управленческое решение, которое вероятней всего будет принято на основе:

- тестового покрытия;
- анализа рисков;
- ухудшения тестирования.

Более детальному изучению данного вопроса поможет крутейшая статья Майкла Болтона в его блоге с разными там эвристиками «пиньяты» и «мертвой лошади».

11.Какие основные уровни тестирования ПО?

- **Компонентное** (component)/ **модульное тестирование** (module, unit testing) — это тестирование отдельных компонентов ПО [IEEE 610];
- **Интеграционное тестирование** (integration testing) — это тестирование, выполняемое для обнаружения дефектов в интерфейсах и во взаимодействии между интегрированными компонентами или системами.

  Следует также понимать что такое big-bang тестирование, тестирование «сверху вниз», восходящие и инкрементное тестирование;

- **Системное тестирование** (system testing) — это процесс тестирования системы в целом с целью проверки того, что она соответствует установленным требованиям;
- **Приёмочное тестирование** (acceptance testing) — это формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приёмки (критерии выхода, которым должны соответствовать компонент или система, для того, чтобы быть принятыми пользователем, заказчиком или другим уполномоченным лицом) [IEEE 610], с чего следуют такие **виды**, о которых желательно не забывать:

- пользовательское приёмочное тестирование (user acceptance testing);
- производственное приемочное тестирование (factory acceptance testing) — это приемочное тестирование, проводимое на стороне разработчика программного продукта и проводящееся сотрудниками компании-поставщика с целью определить, соответствует или нет компонент или система как программным, так и аппаратным требованиям;
- стороннее приемочное тестирование (site acceptance testing) — это приёмочное тестирование пользователями или заказчиком на своей стороне. Проводится с целью определить как соответствие бизнес-процессу, так и удостовериться, что данная система или компонент удовлетворяет потребностям пользователей или заказчика. Обычно включает в себя проверку как программного обеспечения, так и технической базы;
- эксплуатационное приемочное тестирование (operational acceptance testing) — это эксплуатационное тестирование в фазе приемочного тестирования, обычно выполняемое пользователем и/или сотрудниками с администраторским доступом, в рабочей среде (возможно, симулированой), фокусируясь на функциональных аспектах (восстанавливаемость, поведение ресурсов, устанавливаемость и техническое соответствие).

- **Альфа-тестирование** (alpha testing) — это моделируемое или действительное эксплуатационное тестирование потенциальными пользователями/заказчиками или независимой командой тестирования на стороне разработчиков, но вне разрабатывающей организации. Альфа-тестирование часто применяется к коробочному ПО в качестве внутреннего приёмочного тестирования;
- **Бета-тестирование** (beta testing) — это эксплуатационное тестирование потенциальными и/или существующими клиентами/заказчиками на внешней стороне никак не связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. Бета-тестирование часто проводится как форма внешнего приёмочного тестирования готового программного обеспечения для того чтобы получить отзывы рынка;

12.Что такое критерии входа?

Критерии входа (entry criteria) — это набор общих и специфичных условий для продолжения процесса с определенной задачей, например, фаза тестирования. Цель критериев входа — предотвращение начала задачи, которое может потребовать больше (бесполезных) усилий, чем на устранение не пройденных критериев входа.

Проще говоря для Вас, как будущего тестировщика, критерии входа следует понимать как основные условия, которые должны быть выполнены до того, как Вы и Ваша команда могут начать тестирование.

13.Приведите несколько примеров, которые объясняют критерии входа для тестирования ПО.

- все дефекты, которые относятся к ранним стадиям (проектирования) закрыты и проверены;
- код проверенный с помощью осуществления «Unit» тестов;
- основные функциональные возможности ПО готовы для тестирования;
- имеется документация, которая определяет требования;
- все тестировщики ознакомлены с архитектурой ПО;
- все тестировщики ознакомлены с целями проекта;
- готова среда тестирования;
- доступные для использования билды;
- утверждены план тестирования и/или тестовые случаи.

14.Что такое критерии выхода?

Критерии выхода (exit criteria) — это набор общих и специфичных условий, согласованных заранее с заинтересованными сторонами, для того, чтобы процесс мог официально считаться завершенным. Цель критериев выхода — предотвращение возможности, когда задание считается завершенным, однако еще существуют отдельные незавершенные части задания. Критерии выхода используются для отчетности, а также планирования того, когда остановить тестирование.

Проще говоря, как критерии входа определяют начало тестирования, так и критерии выхода определяют его окончание и ПО готово к следующему этапу жизненного цикла (внедрение и т.д.).

15.Приведите несколько примеров, которые объясняют критерии выхода для тестирования ПО.

- все заранее предопределенные области ПО как «рисковые» протестированы и такой статус понижен/удален;
- все ошибки тщательно задокументированы и доведены менеджменту/акционерам/заказчикам;
- все тесты с высоким приоритетом пройдены и соответственно помечены как «Pass»;
- все требования документации SRS (Спецификация требований ПО);
- STR утверждено собственником проекта;
- протестирована архитектура ПО;
- ни одна серьезная или критическая ошибки не остаются открытыми;
- 90-95% всех тестов сделано.

16.Приведите несколько инструментов, которые могут использоваться для автоматизации тестирования.

- внутренние инструменты автоматизации;
- серия программных продуктов Selenium;
- MS VS;

- JUnit;
- Test Complete;
- LoadRunner;
- Testing Anywhere;
- WinRunner.

В дополнение — результаты крутого опроса по автоматизированному тестированию.

17.Как вы объясните Bug/Defect/Error в ПО?

Любая проблема/ошибка в ПО, что вызвана следующим поведением:

- поломка ПО или отображение недействительного уведомления;
- ПО предоставляет недействительные результаты;
- ПО не удалось выполнить, как ожидалось (любое отклонение от ожидаемых результатов).

18.Объясните процесс верификации.

Реальный вопрос, на который мы ищем ответ: **строим ли мы продукт правильно?**

Процесс верификации (verification) выполняется, чтобы убедиться, что каждый этап жизненного цикла разработки ПО (разработка, тестирование и т.д.) строится на основе предопределенных требований (requirements) и спецификаций (specifications) и без каких-либо отклонений от них. (см. № 7)

19.Опишите различные мероприятия процесса верификации.

- анализ различных аспектов тестирования (сроки, ресурсы, персонал, стоимость, инструменты тестирования и т.д.);
- **покрытие операторов** (statement coverage) — процентное отношение операторов, исполняемых набором тестов, к их общему количеству; **покрытие условий** (condition coverage) — процент исходов условий, которые были проверены набором тестов. 100% покрытие условий требует, чтобы каждое отдельное условие в каждом выражении решения было проверено как Истина и Ложь; **покрытие альтернатив** (decision coverage) — процент результатов альтернативы, который был проверен набором тестов. Стопроцентное покрытие решений подразумевает стопроцентное покрытие ветвей и стопроцентное покрытие операторов;
- **рецензирование** (review) — это оценка состояния продукта или проекта с целью установления расхождений с запланированными результатами и для выдвижения предложений по улучшению. Примерами рецензирования могут служить: управленческое рецензирование, неформальное рецензирование, технический анализ, инспекция и разбор;
- **разбор** (walkthrough) — это пошаговый разбор, проводимый автором документа для сбора информации и обеспечения одинакового понимания содержания документа;
- **инспекция** (inspection) — это тип равноправного анализа, основанный на визуальной проверке документов для поиска ошибок. Например, нарушение

стандартов разработки и несоответствие документации более высокого уровня. Наиболее формальная методика рецензирования и поэтому всегда основывается на документированной процедуре.

20.Приведите примеры верификации в зависимости от уровней тестирования. (см. № 11)

- Модульное тестирование (unit testing):
  -проверка осуществления проектирования программного обеспечения.
- Интеграционное тестирование (integration testing):
  -тестирование на интеграцию между всеми соответствующими компонентами до того как ПО перейдет к следующему уровню (system).
- Системное тестирование (system testing):
  -обеспечение соответствия системы предопределенным требованиям и спецификации.
- Приёмочное тестирование (acceptance testing):
  -убедитесь, что система отвечает требованиям клиента.

21.Объясните процесс валидации.

Реальный вопрос, на который мы ищем ответ: **строим ли мы правильный продукт?**

Процесс, позволяющий тестировщику оценить ПО после стадии разработки до передачи его заказчику. В этом процессе мы должны убедиться, что ПО разработано на основе потребностей пользователей.

Помните, валидация охватывает динамическую сторону тестирования, где определенное ПО тестируется и оценивается вопреки заданной SRS документации.

22.Приведите несколько причин, которые приводят к багам в ПО.

- человеческие ошибки (процесс проектирования и процесс реализации);
- изменение требований в то время как ПО под испытанием;
- непонимание требований и спецификаций;
- отсутствие времени;
- плохая приоритизация тестирования;
- плохая ориентация в версиях ПО;
- сложность самого ПО.

23.Что такое процедура тестирования (Test Procedure)?

Документ, описывающий последовательность действий при выполнении теста. Также известен как ручной сценарий тестирования.

24.Что такое программный компонент?

В основном, программные компоненты — это мелкие айтемы ПО, которые построены из еще мелких единиц, которые в свою очередь интегрированы вместе (классы, методы, хранимые процедуры, бинарные деревья и т.д.).

25.Объясните Покрытие кода.

Покрытие кода (code coverage) — это метод анализа, определяющий, какие части ПО были проверены (покрыты) набором тестов, а какие нет, например, покрытие операторов, покрытие альтернатив или покрытие условий.

26.Объясните Инспекцию кода.

Инспекция кода (code inspection) или просмотр кода (code review) — это систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. Целью просмотра является улучшение качества программного продукта и совершенствование навыков разработчика.

В процессе инспекции могут быть найдены и устранены такие проблемы, как ошибки в форматировании строк, состояние гонки (race condition), утечка памяти (memory leak) и переполнение буфера (buffer overflow), что улучшает безопасность программного продукта. Системы контроля версий дают возможность проведения совместной инспекции кода. Кроме того, существуют специальные инструментальные средства для совместной инспекции кода.

Программное обеспечение для автоматизированной инспекции кода упрощает задачу просмотра больших кусков кода, систематически сканируя его на предмет обнаружения наиболее известных уязвимостей.

27.Что значит фраза Код завершен?

Простой термин, имеющий отношение к конкретному этапу SDLC. Говоря «код завершен», мы на самом деле имеем ввиду, что его реализация завершена (вся функциональность ПО успешно реализована). Хотя если даже код будет полностью реализован, всегда есть новые ошибки обнаруженные во время тестирования.

28.Что такое разбор (walkthrough) кода?

Разбор (walkthrough) — это методика тестирования, используемая для обзора хода осуществления кода программистом и командой тестирования, во время разбора код выполняется с помощью нескольких простых тестов, чтобы определить его качество и логику.

29.Что такое отладка?

Отладка (debugging) — это процесс поиска, анализа, и устранения причин отказов и ошибок в ПО.

Чтобы понять, где возникла ошибка, приходится:

- узнавать текущие значения переменных;
- выяснять, по какому пути выполнялась программа.

Существуют две взаимодополняющие технологии отладки.

- Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.

30.Что такое эмулятор и симулятор?

Эмуляция — это воспроизведение работы программы или системы (а не какой-то её мизерной части) с сохранением ключевых её свойств и принципов работы. Эмуляция выполняет программный код в привычной для этого кода среде, состоящей из тех же компонентов, что и эмулируемый объект.

Симуляция — это воспроизведение работы программы-оригинала сугубо виртуально, на движке специальной программы (средство разработки курсов, к примеру). Симуляция лишь имитирует выполнение кода, а не копирует его, всё виртуально на 100%, всё «понарошку».

Следовательно:

Эмулятор ПО — это полнофункциональный аналог оригинального ПО, либо его версия, в которой может быть предусмотрен ряд ограничений по функционалу, возможностям и поведению ПО.

Симулятор ПО — это модель оригинального ПО, в которой реализуется логика работы этого ПО (частично или полностью), имитируется поведение ПО, копируется его интерфейс.

Симулятор по полноте функций/учитываемых параметров уже, чем эмулятор. Эмулируется объект, а симулируются его свойства, функции или поведение.

31.Что такое спецификация ПО?

Спецификация (specifications) — это текстовый файл с описанием того, что нужно протестировать в тестовых данных. В ней указывается какие результаты должна получить программа. Тестовый код находит реальные, вычисленные на живом коде результаты. А тестовый движок производит сверку спецификации и вычисленных результатов.

Спецификацию получаем от заказчика проанализировав, исследовав его требования и переведя их на качественно новый, более детализированный уровень, на котором ими и будет пользоваться команда разработчика.

32.Что такое кодирование?

Кодирование (coding) — это процесс написания программного кода, скриптов, с целью реализации определённого алгоритма на определённом языке программирования.

Некоторые путают такие понятия, как программирование и непосредственно кодирование. Кодирование является лишь частью программирования, наряду с анализом, проектированием, компиляцией, тестированием и отладкой, сопровождением (В узких кругах кодирование также может называться «кодинг». Однако, если верить Вики, в литературе этот термин используется редко.).

33.Что такое требование?

Требование (requirement) — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований.

Требования могут выражаться в виде текстовых утверждений и графических моделей.

В классическом техническом подходе совокупность требований используется на стадии проектирования программного обеспечения (ПО). Требования также используются в процессе тестирования ПО, так как тесты основываются на определённых требованиях.

Этапу разработки требований, возможно, предшествовало технико-экономическое обоснование, или концептуальная фаза анализа проекта. Фаза разработки требований может быть разбита на выявление требований (сбор, понимание, рассмотрение и выяснение потребностей заинтересованных лиц), анализ (проверка целостности и законченности), спецификация (документирование требований) и проверка правильности.

34.Что такое тестирование стабильности?

Задачей тестирования **стабильности** (stability) / **надежности** (reliability) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты влияющие именно на стабильность работы.

35.Расскажите про критичность (серьезность) бага и общепринятые уровни такой критичности.

Критичность (severity) — это важность воздействия конкретного дефекта на разработку или функционирование компонента или системы.

Критичность ошибки определяется тестировщиком, который обнаружил баг, но перед этим он должен ответить себе на такие вопросы:

- Как эта ошибка будет влиять на процесс тестирования?

- Как эта ошибка будет влиять на клиента?
- Как эта ошибка влияет на систему?
- Как эта ошибка влияет на сроки тестирования?
- Блокирует ли эта ошибка другие тесты?
- И т.д.

Каждая компания может определить свою собственную шкалу для степени критичности (серьезности), но есть несколько уровней, которые используются почти всеми командами:

- **Blocker/show-stopper** (блокирование) — ПО или конкретный компонент не подходит для использования/тестирования (полный отказ, краш системы и т.д.) и нет обхода.
  Примеры: система рухнет, когда пользователь нажимает кнопку «Пуск»; система не запускается после повреждения инсталлятора; отключение ПО, вызванное аппаратными сбоями.
- **Critical** (критический) — главная функциональность не работает как надо, есть обходной путь, который может повлиять на целостность испытаний.
  Пример: ПО может рандомно крашиться, используя различные функциональные возможности; ПО вырабатывает противоречивые результаты, основные требования не удается подтвердить.
- **Major** (важный) — поражение незначительных функциональностей, нет влияния на другие компоненты, и есть быстрый и действующий обход.
  Пример: Пользователь не может использовать определенную функциональность напрямую, но может использовать ее же, воспользовавшись доступом к ней из разных модулей.
- **Minor** (второстепенный, незначительный) — незначительное воздействие на определенном месте, нет необходимости создавать обходной путь, целостность ПО не затронута.
  Примеры: ошибки орфографии, улучшения, запрос на изменение

36.Расскажите про приоритет бага.

Приоритет (priority) — это степень важности, присваиваемая багу. Другими словами определяется, насколько срочно это ошибка должна быть исправлена.

Приоритет — инструмент менеджмента, и перед его определением последний должен ответить минимум на следующие вопросы:

- Как баг влияет на сроки?
- Как баг влияет на процесс тестирования?
- Как баг влияет на работу остальных тестировщиков?
- Каковы затраты необходимы на устранение бага?
- Должны ли мы изменить требования к ПО?

37.Что такое Сборка?

Сборка (build) — подготовленный для использования информационный продукт. Чаще всего это исполняемый файл (двоичный файл, содержащий исполняемый код программы).

Предположим, что номер версии сборки выглядит так: 1.35.6.2

- Первый идентификатор — основной номер версии.
- Второй идентификатор — дополнительный номер версии.
- Третий идентификатор — номер сборки.
- Четвёртый идентификатор — номер редакции.

38.Можно ли начинать тестирование без рабочей сборки?

Безусловно — да! Ведь существует два типа методологии тестирования (статическое и динамическое), которые позволяют тестировщику начинать работать без рабочей сборки статическим методом, тем более такой метод более рентабельный чем «динамический».

39.Что такое статический анализ?

Статический анализ (static analysis) — это анализ программных артефактов, таких как требования или программный код, проводимый без исполнения этих программных артефактов.

40.Что такое тестовый драйвер и тестовая обвязка?

Драйвер (driver) — это компонент ПО или средство тестирования, которое заменяет компонент, обеспечивающий управление и/или вызов компонента или системы.

Обвязка (harness) — это тестовое окружение, включающее в себя заглушки и драйверы, необходимые для проведения теста.

41.Что такое матрицы трассировки?

Для измерения покрытия требований, необходимо проанализировать требования к продукту и разбить их на пункты. Опционально каждый пункт связывается с тест кейсами, проверяющими его. Совокупность этих связей — и является **матрицей трассировки** (traceability matrix). Проследив связи, можно понять какие именно требования проверяет тестовый случай.

Тесты не связанные с требованиями не имеют смысла. Требования, не связанные с тестами — это «белые пятна», т.е. выполнив все созданные тест кейсы, нельзя дать ответ реализовано данное требование в продукте или нет.

42.Что такое тестирование End-to-End?

End-to-end тестирование — это тип тестирования где тестировщик использует ПО (сценарии, которые исследуют весь поток выполнения) в условиях которыми вероятней всего обладает пользователь.

В дополнение, тесты будут работать сочетая в себе несколько сценариев уместных в реальном мире:

- запуск ПО в среде с задержками связи сети;
- запуск ПО в среде с низкими ресурсами;
- Запуск ПО на другом серверном оборудовании;
- Запустите ПО в той же среде, что имеет некоторые другие приложения, которые потребляют ресурсы сервера.

43. Что такое функциональное тестирование? Какие основные типы функционального тестирования? Какие виды функциональных тестов вы знаете?

Функциональное тестирование (functional testing) — это тестирование, основанное на анализе спецификации функциональности компонента или системы.

Функциональные тесты базируются на функциях и особенностях, а также взаимодействии с другими системами, и могут быть представлены на всех уровнях тестирования: компонентном или модульном (Component/Unit testing), интеграционном (Integration testing), системном (System testing) иприемочном (Acceptance testing). Функциональные виды тестирования рассматривают внешнее поведение системы.

Далее перечислены одни из самых распространенных **видов функциональных тестов**:

- **Функциональные тесты** основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (use cases). Тестирование функциональности может проводиться в двух аспектах:
    - тестирование **в перспективе «требования»** (requirement-based testing) использует спецификацию функциональных требований к системе как основу для дизайна тестовых случаев (Test Cases). В этом случае необходимо сделать список того, что будет тестироваться, а что нет; приоритезировать требования на основе рисков (если это не сделано в документе с требованиями), а на основе этого приоритезировать тестовые сценарии (test cases). Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.
    - тестирование в перспективе **«бизнес-процессы»** (business-process-based testing) использует знание этих самых бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этой перспективе тестовые сценарии (test scripts), как правило, основываются на случаях использования системы (use cases).

- **Тестирование защищенности** (security testing) — это тестирование с целью оценить защищенность ПО. Общая стратегия безопасности основывается на трех основных принципах:
    - конфиденциальность;
    - целостность;
    - доступность;

  В настоящее время наиболее распространенными видами уязвимости в безопасности программного обеспечения являются:
    - ***xss (cross-site scripting)*** *— это вид уязвимости программного обеспечения (Web приложений), при которой, на генерированной сервером странице, выполняются вредоносные скрипты, с целью атаки клиента;*
    - ***xsrf / csrf (request forgery)*** *— это вид уязвимости, позволяющий использовать недостатки HTTP протокола, при этом злоумышленники работают по следующей схеме: ссылка на вредоносный сайт установливается на странице, пользующейся доверием у пользователя, при переходе по вредоносной ссылке выполняется скрипт, сохраняющий личные данные пользователя (пароли, платежные данные и т.д.), либо отправляющий СПАМ сообщения от лица пользователя, либо изменяет доступ к учетной записи пользователя, для получения полного контроля над ней;*
    - ***code injections (sql, php, asp и т.д.)*** *— это вид уязвимости, при котором становится возможно осуществить запуск исполняемого кода с целью получения доступа к системным ресурсам, несанкционированного доступа к данным либо выведения системы из строя;*
    - ***server-side includes (ssi) injection*** *— это вид уязвимости, использующий вставку серверных команд в HTML код или запуск их напрямую с сервера;*
    - ***authorization bypass*** *— это вид уязвимости, при котором возможно получить несанкционированный доступ к учетной записи или документам другого пользователя.*

- **Тестирование возможности взаимодействия** (interoperability testing) — это процесс тестирования для определения возможности взаимодействия программного продукта.

Функциональное тестирование разделяются по ожидаемому результату на позитивные и негативные тесты:

- **Позитивный тест** использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию.
- **Негативный тест** 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.

44.Что такое нефункциональное тестирование?

Нефункциональное тестирование (non-functional testing) — это тестирование атрибутов компонента или системы, не относящихся к функциональности, то есть надежность, эффективность, практичность, сопровождаемость, переносимость и т.д. (тесты, сделанные по всем аспектам, которые непосредственно не связанны с конкретным действием пользователя).

45.Несколько примеров тестов, которые включает в себе нефункциональное тестирование?

- **Тестирование переносимости** (portability testing) это процесс тестирования с целью определить переносимость программного продукта.
- **Тестирование возможности взаимодействия** (interoperability testing) — это Процесс тестирования для определения возможности взаимодействия программного продукта.
- **Тестирование производительности** (performance testing) — это процесс тестирования с целью определить производительность программного продукта.
- **Тестирование надежности** (reliability testing) — это процесс тестирования, исследующий надежность программного продукта.
- **Тестирование практичности** (usability testing) — это тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации.
- **Тестирование безопасности** (safety testing) — это тестирование программного продукта с целью определить его безопасность.
- **Стрессовое тестирование** (stress testing) — это вид тестирования производительности, оценивающий систему или компонент на граничных значениях рабочих нагрузок или за их пределами, или же в состоянии ограниченных ресурсов, таких как память или доступ к серверу.
- **Нагрузочное тестирование** (load testing) — это вид тестирования производительности, проводимый с целью оценить поведение компонента или системы под увеличивающейся нагрузкой (число одновременно работающих пользователей и/или число транзакций) для определения максимально допустимого уровня нагрузки для исследуемого компонента или системы.

46.Что такое тестирование Белого ящика?

Белый ящик — это тестирование кода на предмет логики работы программы и корректности её работы с точки зрения компилятора того языка, на котором она писалась.

Техника тестирования по принципу Белого ящика, также называемая техникой тестирования, управляемой логикой программы, позволяет проверить внутреннюю структуру программы. Исходя из этой стратегии тестировщик получает тестовые данные путем анализа логики работы программы.

Техника Белого ящика включает в себя следующие методы тестирования:

- покрытие решений;
- покрытие условий;
- покрытие решений и условий;
- комбинаторное покрытие условий;

47.Что такое тестирование Чёрного ящика?

Тестирование чёрного ящика или поведенческое тестирование — это стратегия (метод) тестирования функционального поведения ПО с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из технических требований и их спецификаций

Некоторые приёмы тестирования чёрного ящика:

- эквивалентное разбиение;
- анализ граничных значений;
- анализ причинно-следственных связей;
- предположение об ошибке;

48.Что такое Конверсионное тестирование?

Конверсионное тестирование (conversion testing) — это методика тестирования, что используется для проверки того, как имеющие в системе А данные будут преобразовываться и становиться доступными для использования в системе Б.

49.Что такое Конформационное тестирование?

Конформационное тестирование — это тестирование с целью убедиться, что ПО отвечает определенным отраслевым стандартам (IEEE, W3C и т.д.) для разработки ПО.