Виды тестирования (https://guru.qahacking.ru/ - Собаседник)

Классификация по доступу к коду

- Черный ящик
  - Определение можно сформулировать так: тестировщик оказывает на приложение воздействия (и проверяет реакцию) тем же способом, каким при реальной эксплуатации приложения на него воздействовали бы пользователи или другие приложения. В рамках тестирования по методу черного ящика основной информацией для создания (тест-кейсов выступает документация (особенно - требования) и общий здравый смысл (для случаев, когда поведение приложения в некоторой ситуации не регламентировано явно; иногда это называют" тестированием на основе неявных требований", но определения у этого подхода нет)
- Серый ящик

По запуску кода на исполнение

- Статическое (Static testing) - ответ системы без запуска программы

По степени автоматизации

- Ручное (Manual testing)
- Автоматизированное (Automated testing)

По степени важности тестируемых функций

- Дымовое тестирование (Smoke test, Intake test, Build verification test)
- Тестирование критического пути (Critical path test)
- Расширенное тестирование (Extended test)

По принципам работы

- Позитивное (Positive testing) - Делаем всё правильно, согласно функционалу/по правилам
- Негативное (Negative testing, Invalid testing) - Проводится, когда нужна неадекватная реакция системы. Деструктивные действия, вызывающие сбой в системе. Пример: мы вводим неверные символы в поле, ждем отсутствие ответа на ошибку, т.е. неожиданый ответ системы.

По природе происхождения (веб-сайт можно открыть на компьютере, телефоне, ноутбуке)

- Веб - сайты (Web-applications testing)
- Мобилка - телефон, планшеты (Mobile applications testing)
- Настольное (Desktop applications testing) - то, что устанавливается на компьютер

По фокусировке на уровне архитектуры приложения

- Тестирование уровня представления (Presentation tier testing)
- Тестирование уровня бизнес логики (Business logic tier testing)

По привлечению конечных пользователей (уточнить степень готовности сайта). Думаю, что он не готов для использования конечными пользователями.

По степени формализации

- Тестирование на основе тест-кейсов (Scripted testing, Test case based testing)
- Исследовательское (может включать интуитивное) - идём по сценарию (Exploratory testing). На основе тест-кейсов - пишутся определенные задачи. Пример: есть 3 тестировщика, а ты новичок. Чтобы не тратить много времени на знакомство с продуктом - дают чек-кейсы.
- Свободное (интуитивное) тестирование (Ad hoc testing) - Интуитивное (ad hoc) – пишется чек-лист без документации - проверка: работает система или нет, работа с элементами системы - Функциональное тестирование - например: кнопка возвращения назад, кнопка "Звонок" или "добавить файл" и "отправка данных".Нефункциональное (форма с заполнениями данными - поставить галочки (чек-бокс) - что мы можем ставить галочки или кружочки, ввод данных в поле, можем ли нажать на кнопку, работает ли ссылка (пример: переход по социальных сетям). Табы ("О нас", "Новости", "Категория товаров" и т.п.) - переход по ссылке. Функциональное/нефункциональное (Например: ввести информацию в поле - нефункциональное, отправить эту информацию - функциональное)

По назначению функций или по целям и задачам

- Функциональное (направлено на проверку корректности работы приложения. Пример: авторизация)
- Нефункциональное (Пример: проверяем дизайн, удобно ли расположение кнопок для пользователя, можем проверить валидацию (если проверяем без входа - без проверки валидации) - вводятся неверные символы)
- Приёмочное (проверка приложения с точки зрения пользователя, что соответствует бизнес-задачам и требованиям)
- Операционное тестирование (operational testing) — тестирование, проводимое в реальной или приближенной к реальной операционной среде (operational environment), включающей операционную систему, системы управления базами данных, серверы приложений, веб-серверы, аппаратное обеспечение и т.д.
- UAT (юзер ассептенс) - принимается по пользовательским сценариям (например: как пользователь взаимодействует с кнопкой - моделирование поведения пользователя без к.-л. негативных сценариев, пользователь может ввести символы - он может их ввести, пользователь может нажать на кнопку - кнопка нажимается и т.п.)
- Тестирование интерфейса - UI (проверка дизайна, картинки, размер кнопок и их расположение, шрифт - тип/размер)/ удобство использования UX (проверка на сколько удобно пользоваться приложением)
- Безопасность (проверка системы противостоять хакерским атакам, проверка на утечку данных, проверка пароля (чтобы он скрывался "звёздочками" и "точечками" - астериски)
- Глобализация, Интернационализация. Локализация - можно предложить доработать этот компонент на митапе
- Тестирование совместимости (compatibility testing, interoperability testing) — тестирование, направленное на проверку способности приложения работать в указанном окружении.
- Конфигурационное тестирование - Совместимость с браузерами и их версиями (кросс-браузерное тестирование, cross-browser testing).
- Кроссбраузерное/кроссплатформенное - проверка приложения на различных платформах и браузерах.
- Тестирование использования ресурсов (resource utilization testing, efficiency testing, storage testing) — совокупность видов тестирования, проверяющих эффективность использования приложением доступных ему ресурсов и зависимость результатов работы приложения от количества доступных ему ресурсов. Часто эти виды тестирования прямо или косвенно примыкают к техникам тестирования производительности.
- Сравнительное тестирование (comparison testing) — тестирование, направленное на сравнительный анализ преимуществ и недостатков разрабатываемого продукта по отношению к его основным конкурентам/
- Тестирование надёжности (reliability testing) — тестирование способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.
- Нагрузочное (способность системы работать по планируемой нагрузке - например с помощью JMeter и др. инструментов - не на боевой системе)
- Тестирование масштабируемости (scalability testing) — исследование способности приложения увеличивать показатели производительности в соответствии с увеличением количества доступных приложению ресурсов.
- Объёмное тестирование (volume testing) — исследование производительности приложения при обработке различных (как правило, больших) объёмов данных.
- Стресс-тест (проверка системы на превышающую нагрузку - сайты интернет-магазины "Чёрная пятница" - нужна ли дополнительная мощность; либо в часы максимальной нагрузки)
- Конкурентное тестирование (concurrency testing) — исследование поведения приложения в ситуации, когда ему приходится обрабатывать большое количество одновременно поступающих запросов, что вызывает конкуренцию между запросами за ресурсы (базу данных, память, канал передачи данных, дисковую подсистему и т.д.). Иногда под конкурентным тестированием понимают также исследование работы многопоточных приложений и корректность синхронизации действий, производимых в разных потоках.