Geekbrains

**Исследование особенностей организации процесса ручного тестирования веб-приложений в условиях отсутствия технического задания на примере тестирования веб-приложения строительной компании “AVANT GROUP”**

Программа:

Автоматизация тестирования. Специалист.

Гулу-заде Ирада Рауф кызы

Баку

2024 г.

**Содержание**

Введение (2-3 стр)

Глава 1. Теоретические основы ручного тестирования веб-приложений

1.1. Особенности тестирования веб-приложений.

1.2. Понятие тестирования программного обеспечения.

1.3. Ручное тестирование: сущность, этапы, инструменты.

1.4. Проблемы организации тестирования при отсутствии технического задания.

**Глава 2. Подготовка к ручному тестированию веб-приложения (~15 стр)**

2.1. Составление чек-листов

2.2. Составление тест-кейсов

2.3. Разработка тест-плана

2.4. Тест-дизайн и тест-аналитика

2.5. Баг-репорты

**Глава 3. Практическое исследование разработка предложений по улучшению тестирования веб-приложения строительной компании «AVANT GROUP» в условиях отсутствия ТЗ (~20 стр)**

3.1. Тестирование функциональности

3.2. Тестирование UI и UX

3.3. Подведение итогов. Оценка эффективности предложенной методики и рекомендации по ее применению.

Заключение

Приложения

# Введение

В современном мире, где информационные технологии играют ключевую роль, веб-приложения стали неотъемлемой частью практически любого бизнеса. Обеспечение их качества и надежности является приоритетной задачей, поскольку от этого напрямую зависят удовлетворенность пользователей и успех компании. Ручное тестирование остается важнейшим этапом обеспечения качества веб-приложений, особенно на ранних стадиях разработки и при ограниченных ресурсах. Отсутствие формализованного технического задания (ТЗ) на разработку веб-приложения – распространенная ситуация, особенно в компаниях малого и среднего бизнеса. Это создает дополнительные трудности для тестировщиков, вынужденных действовать в условиях неполной информации и размытых требований к продукту.

**Актуальность данной дипломной работы** обусловлена необходимостью исследования особенностей организации процесса ручного тестирования веб-приложений в условиях отсутствия ТЗ и разработки рекомендаций по повышению эффективности данного процесса.

**Объектом исследования** выступает процесс ручного тестирования веб-приложений. **Предмет исследования** – особенности организации ручного тестирования веб-приложений в условиях отсутствия ТЗ.

**Целью дипломной работы** является исследование особенностей организации процесса ручного тестирования веб-приложений в условиях отсутствия ТЗ на примере тестирования веб-приложения, а также разработка рекомендаций по повышению эффективности данного процесса.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные виды и методы ручного тестирования веб-приложений.
3. Ознакомиться с основными принципами составления, такой тестовой документации, как чек-листы, тест-кейсы, тест-планы, баг-репорты.
4. Составить план ручного тестирования веб-приложения “AVANT GROUP” с использованием техник тест-дизайна.
5. Выполнить ручное тестирование веб-приложения “AVANT GROUP” .
6. Разработать предложения по улучшению ручного тестирования веб-приложения “AVANT GROUP”.

**Практическая значимость работы** заключается в возможности применения полученных результатов и разработанных рекомендаций для повышения эффективности процесса ручного тестирования веб-приложений в компаниях, где разработка ПО ведется без формализованного ТЗ.

Структура дипломной работы включает в себя введение, три главы, заключение, список использованных источников и приложения. В первой главе рассматриваются теоретические основы ручного тестирования веб-приложений. Вторая глава посвящена анализу особенностей организации ручного тестирования при отсутствии ТЗ. В третьей главе представлены результаты практического исследования на примере тестирования веб-приложения строительной компании и даны рекомендации по повышению эффективности процесса.

Инструменты, используемые в работе: Google Sheets, DevTools, Qase.io, GitHub, Visual Studio Code, Bandicam.

**Состав команды**: Гулу-заде Ирада Рауф кызы (Тестировщик)

# Глава 1. Теоретические основы ручного тестирования веб-приложений

# 1.1. Особенности тестирования веб-приложений

В качестве объекта в данной работе мы рассматриваем процесс ручного тестирования веб-приложений, поэтому для начала стоит рассмотреть, что же такое веб-приложение?

Веб-приложения предоставляют пользователям доступ к информации и сервисам через интернет, обеспечивая взаимодействие с различными системами и данными. Веб-приложения отличаются от традиционного программного обеспечения рядом особенностей, которые необходимо учитывать при их тестировании:

1. Распределенная архитектура: Веб-приложения обычно состоят из клиентской и серверной частей, взаимодействующих по сети.
2. Использование различных технологий: Для разработки веб-приложений используется широкий спектр технологий, включая языки программирования (JavaScript, Python, PHP), фреймворки (React, Angular, Vue.js), базы данных (MySQL, PostgreSQL) и другие.
3. Кроссплатформенность: Веб-приложения должны корректно работать в различных браузерах (Chrome, Firefox, Safari, Edge) и на разных операционных системах (Windows, macOS, Linux, Android, iOS).
4. Высокие требования к производительности и масштабируемости: Веб-приложения должны быстро загружаться, обрабатывать большое количество запросов и пользователей, а также легко масштабироваться при необходимости.
5. Постоянное развитие: Веб-технологии стремительно развиваются, поэтому веб-приложения нуждаются в постоянном обновлении и доработке.

Веб-приложения можно разделить на несколько типов, в зависимости от разных сочетаний его основных составляющих:

1. Backend (бэкенд или серверная часть приложения) работает на удаленном компьютере, который может находиться где угодно. Она может быть написана на разных языках программирования: PHP, Python, Ruby, C# и других. Если создавать приложение используя только серверную часть, то в результате любых переходов между разделами, отправок форм, обновления данных, сервером будет генерироваться новый HTML-файл и страница в браузере будет перезагружаться.
2. Frontend (фронтенд или клиентская часть приложения) выполняется в браузере пользователя. Эта часть написана на языке программирования Javascript. Приложение может состоять только из клиентской части, если не требуется хранить данные пользователя дольше одной сессии. Это могут быть, например, фоторедакторы или простые игрушки.
3. Single page application (SPA или одностраничное приложение). Более интересный вариант, когда используются и бэкенд и фронтенд. С помощью их взаимодействия можно создать приложение, которое будет работать совсем без перезагрузок страницы в браузере. Или в упрощенном варианте, когда переходы между разделами вызывают перезагрузки, но любые действия в разделе обходятся без них.

Учитывая особенности веб-приложений, для обеспечения их качества необходимо применять различные виды тестирования:

1. Функциональное тестирование: Проверка соответствия приложения функциональным требованиям и корректности выполнения всех его функций.
2. Тестирование пользовательского интерфейса (UI) и удобства использования (UX): Оценка удобства использования приложения, интуитивности интерфейса, доступности для пользователей с ограниченными возможностями.
3. Тестирование безопасности: Проверка защищенности приложения от несанкционированного доступа, утечки данных и других угроз безопасности.
4. Нагрузочное тестирование: Оценка производительности приложения при высокой нагрузке (большое количество пользователей, запросов, данных).
5. Стресс-тестирование: Проверка устойчивости приложения к экстремальным нагрузкам и определение его предельных возможностей.
6. Кроссбраузерное тестирование: Проверка корректности отображения и работы приложения в различных браузерах и их версиях.
7. Тестирование совместимости: Проверка работы приложения на различных операционных системах, устройствах, разрешениях экрана.

Тестирование веб-приложений — сложный и многогранный процесс, связанный с рядом трудностей:

1. Многообразие платформ и браузеров: Необходимо обеспечить корректную работу приложения на всех популярных платформах и в браузерах.
2. Сложность воспроизведения ошибок: Ошибки, связанные с сетевыми задержками, работой сервера и другими факторами, могут быть трудно воспроизводимыми.
3. Необходимость постоянного обновления тестов: В связи с частыми изменениями веб-приложений тесты нуждаются в регулярном обновлении.
4. Сложность автоматизации тестирования: Автоматизировать тестирование UI и UX может быть сложно, особенно при использовании динамических элементов.

# 1.2. Понятие тестирования программного обеспечения

# В современном мире, где программное обеспечение (ПО) играет критически важную роль во всех сферах жизни, его качество и надежность выходят на первый план. Ошибки в работе ПО могут привести к серьезным последствиям: финансовым потерям, утечке конфиденциальной информации, нарушению бизнес-процессов и даже угрозе безопасности. Именно поэтому тестирование программного обеспечения является неотъемлемым этапом жизненного цикла разработки ПО.

# Тестирование программного обеспечения - это процесс исследования программного продукта с целью обнаружения дефектов и проверки соответствия требованиям.

# Основные цели тестирования:

# Обнаружение дефектов: Выявление ошибок, несоответствий в работе ПО, некорректной реализации функционала.

# Повышение качества ПО: Обеспечение соответствия ПО установленным требованиям, повышение его надежности, производительности, безопасности и удобства использования.

# Снижение рисков: Минимизация вероятности возникновения ошибок в процессе эксплуатации ПО.

# Повышение доверия к ПО: Подтверждение работоспособности ПО, формирование уверенности заказчика в качестве продукта.

# Существует множество классификаций видов тестирования ПО в зависимости от различных критериев. Рассмотрим основные из них:

# По цели проведения:

# Функциональное тестирование: Проверка соответствия ПО функциональным требованиям, корректности выполнения всех заявленных функций.

# Нефункциональное тестирование: Оценка характеристик ПО, не связанных напрямую с его функционалом: производительность, безопасность, удобство использования, надежность, масштабируемость и др.

# По знанию исходного кода:

# Тестирование «черного ящика» (Black Box Testing): Тестирование, проводимое без знания внутренней структуры кода. Основано на анализе входных данных и ожидаемых результатов.

# Тестирование «белого ящика» (White Box Testing): Тестирование, проводимое с учетом внутренней структуры кода. Включает в себя анализ логики работы приложения, покрытия кода тестами и т.д.

# Тестирование «серого ящика» (Gray Box Testing): Комбинация подходов «черного» и «белого» ящиков, когда тестировщик имеет частичное знание о внутренней структуре ПО.

# По степени автоматизации:

# Ручное тестирование: Тестирование, выполняемое человеком вручную, без использования специальных инструментов автоматизации.

# Автоматизированное тестирование: Тестирование, выполняемое с помощью программных средств, позволяющих автоматизировать запуск тестов, анализ результатов и формирование отчетов.

# По времени проведения:

# Модульное тестирование (Unit Testing): Тестирование отдельных модулей или компонентов ПО. Проводится на ранних этапах разработки.

# Интеграционное тестирование (Integration Testing): Тестирование взаимодействия между отдельными модулями и компонентами ПО.

# Системное тестирование (System Testing): Тестирование ПО в целом, как единой системы.

# Приемочное тестирование (Acceptance Testing): Тестирование, проводимое заказчиком или конечными пользователями с целью проверки готовности ПО к эксплуатации.

# Процесс тестирования ПО не ограничивается простым поиском ошибок. Это итеративный процесс, включающий в себя ряд этапов:

# Планирование тестирования: Определение целей, объектов, стратегии тестирования, необходимых ресурсов, сроков и критериев приемки.

# Разработка тестовой документации: Создание тестовых сценариев (test cases), чек-листов (checklists), планов тестирования (test plans) и других документов.

# Проведение тестирования: Выполнение тестовых сценариев, фиксация результатов и найденных дефектов.

# Анализ результатов тестирования: Обработка полученных данных, оценка качества ПО, принятие решения о необходимости доработки.

Теперь разберемся, чем отличаются две основные профессии в тестировании ПО: тестировщик и QA-инженер.

Тестировщик - член команды разработки, главной целью которого является забота о качестве в виде обнаружения багов до того, как их найдут пользователи.

QA-инженер — тот, кто заботится о качестве в виде превентирования появления багов, то есть принимает меры для предотвращения появления багов в будущем.

Важно помнить, что Качество (как и его отсутствие) — это результат 1) работы всех участников процесса разработки ПО, а также 2) отлаженности и настроек самого процесса.

# 1.3. Ручное тестирование: сущность, этапы, инструменты.

Существует два основных подхода к тестированию: ручное и автоматизированное.

Ручное тестирование представляет собой метод проверки программного обеспечения, при котором тестировщик выступает в роли конечного пользователя, взаимодействуя с приложением вручную, чтобы выявить ошибки, дефекты и несоответствия требованиям.

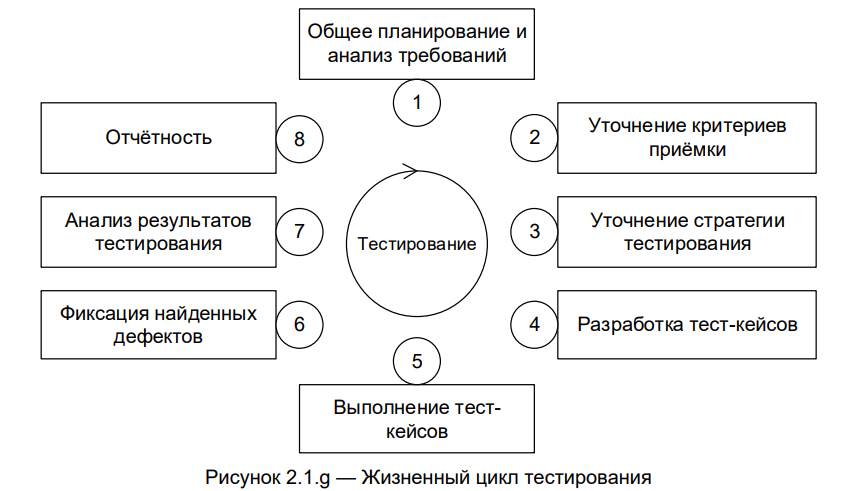
Основные задачи ручного тестирования:

1. Обнаружение дефектов: выявление ошибок в функциональности, интерфейсе, логике работы приложения.
2. Оценка удобства использования: анализ эргономичности, интуитивности и понятности интерфейса для конечного пользователя.
3. Проверка соответствия требованиям: убедиться, что приложение работает в соответствии с заданной спецификацией и требованиями заказчика.

Процесс ручного тестирования обычно включает в себя следующие этапы:

1. Анализ требований: Тщательное изучение документации по проекту, функциональных и нефункциональных требований к приложению.
2. Планирование тестирования: Определение стратегии тестирования, выбор методов и техник, составление тест-плана, определение сроков и необходимых ресурсов.
3. Разработка тестовой документации: Создание тест-кейсов (test cases) — пошаговых инструкций по тестированию конкретного функционала, чек-листов (checklists) — списков проверок, а также другой документации.
4. Проведение тестирования: Выполнение тестовых сценариев, фиксация результатов в баг-трекинговой системе, создание отчетов о дефектах (bug reports) с подробным описанием проблемы и шагов для её воспроизведения.
5. Анализ результатов тестирования: Оценка общего качества приложения, определение степени его готовности к выпуску, формулирование рекомендаций по улучшению продукта.

Более подробно можно рассмотреть в таблице, представленной ниже:



Несмотря на кажущуюся "простоту" ручного тестирования, для его эффективного проведения используется ряд инструментов:

1. Системы управления тестированием (Test Management Systems, TMS): Платформы для создания, хранения, управления и отслеживания тестовой документации (например, TestRail, Zephyr, PractiTest).
2. Баг-трекинговые системы: Системы для регистрации, отслеживания и управления ошибками и дефектами, найденными в процессе тестирования (Jira, Redmine, Bugzilla).
3. Инструменты для создания скриншотов и видеозаписей: Для документирования ошибок и наглядного их представления разработчикам (Lightshot, Snagit, Camtasia).
4. Браузерные инструменты разработчика: Встроенные в браузеры инструменты для анализа HTML, CSS, JavaScript кода, сетевых запросов, производительности и других параметров веб-страниц.

Рассмотрим преимущества и недостатки ручного тестирования.

Преимущества:

1. Обнаружение сложных ошибок: Ручное тестирование, выполняемое человеком, позволяет выявить неочевидные ошибки в логике работы приложения и юзабилити интерфейса, которые могут быть пропущены при автоматизированном тестировании.
2. Гибкость и адаптивность: Ручное тестирование позволяет легко адаптироваться к изменениям в требованиях и быстро вносить коррективы в тестовые сценарии.
3. Низкий порог входа: Для начала работы с ручным тестированием не требуется глубоких технических знаний и навыков программирования.

Недостатки:

1. Высокая стоимость: Ручное тестирование является трудоемким процессом, требующим значительных временных затрат специалистов.
2. Человеческий фактор: Вероятность ошибок со стороны тестировщика, пропуска дефектов в результате усталости или невнимательности.
3. Сложность повторения тестов: Повторное прохождение большого количества тестовых сценариев вручную может быть утомительным и занимать много времени.

В рамках данного исследования будет рассмотрена роль и значение ручного тестирования в условиях отсутствия технического задания, когда требования к продукту могут быть сформулированы не полностью или изменяться в процессе разработки.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Глава 2. Подготовка к ручному тестированию веб-приложения

# 2.1. Составление чек-листов

Чек-лист (checklist) – это документ, содержащий перечень проверок, которые необходимо выполнить для обеспечения качества тестируемого объекта. В контексте тестирования веб-приложения, чек-лист представляет собой список контрольных пунктов, охватывающих различные аспекты его работы: функциональность, удобство использования, производительность, безопасность и другие.

Роль чек-листов в тестировании:

1. Систематизация процесса тестирования: Чек-листы помогают структурировать тестирование, гарантируя, что все важные аспекты будут проверены.
2. Уменьшение риска пропуска дефектов: Четкий перечень проверок снижает вероятность того, что тестировщик забудет проверить какой-либо элемент или функцию.
3. Повышение эффективности тестирования: Чек-листы позволяют сконцентрироваться на тестировании, а не на воспоминании, что именно нужно проверить.
4. Документирование процесса тестирования: Чек-листы служат доказательством того, что тестирование было проведено, и позволяют отслеживать прогресс работ.

Отсутствие технического задания усложняет составление чек-листов, так как отсутствует исчерпывающий источник информации о требованиях к продукту.

В таких условиях рекомендуется следующий подход:

1. Изучение аналогов: Анализ сайтов конкурентов позволяет определить типовые функции и элементы для сайтов данной отрасли, что поможет составить более полный чек-лист.
2. Коммуникация с заказчиком: Необходимо максимально детально уточнить у заказчика его видение сайта, целевую аудиторию, желаемый функционал.
3. Использование прототипов и макетов: Если у заказчика есть прототип или макет сайта, это существенно облегчит составление чек-листа, так как появится наглядное представление о структуре и элементах сайта.
4. Итеративный подход: Чек-лист не является статичным документом, он может и должен дополняться, и корректироваться по мере изучения сайта и получения обратной связи от заказчика.

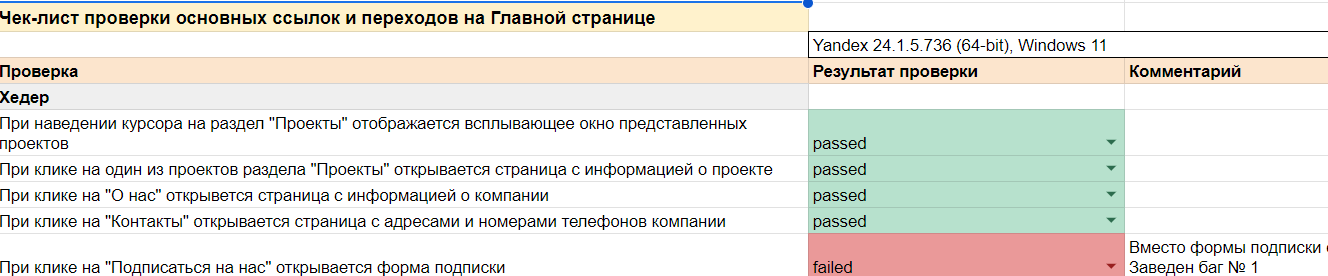
Чек-лист для тестирования веб-приложения обычно включает следующие разделы:

1. Общая информация: Название проекта, версия сайта, дата создания чек-листа, ФИО тестировщика.
2. Функциональное тестирование: Проверка работы всех элементов и функций сайта: формы, кнопки, меню, поиск, фильтры, корзина, оплата и т.д.
3. Тестирование юзабилити: Проверка удобства навигации, понятности интерфейса, логичности расположения элементов, доступности информации.
4. Кроссбраузерное и кроссплатформенное тестирование: Проверка корректности отображения и работы сайта в разных браузерах и на разных устройствах.
5. Тестирование производительности: Проверка времени загрузки страниц, работы сайта при высокой нагрузке и т.д.
6. Тестирование безопасности: Проверка защищенности сайта от взлома, утечки данных и других угроз.

Каждый пункт чек-листа должен содержать:

1. Краткое описание проверки: Что именно нужно проверить.
2. Ожидаемый результат: Как должен работать сайт в идеале.
3. Фактический результат: Что произошло в реальности.
4. Статус: Пройден/Не пройден/Требует уточнения.
5. Комментарий: Дополнительная информация о проблеме или замечание тестировщика.

Ниже приведен пример фрагмента чек-листа для тестирования главной страницы веб-приложения строительной компании:



# 2.2. Составление тест-кейсов

Тест-кейс (test case) — это набор входных значений, предусловий выполнения, ожидаемых результатов и пост-условий выполнения, разработанный для проверки соответствия тестируемого объекта определенному требованию.

Проще говоря, тест-кейс описывает, как именно нужно проверить ту или иную функцию или элемент веб-приложения.

Преимущества использования тест-кейсов:

1. Детализация процесса тестирования: Тест-кейсы описывают шаги тестирования с высокой степенью детализации, что позволяет избежать неоднозначности и субъективности при их выполнении.
2. Повторное использование: Созданные тест-кейсы могут быть использованы повторно при регрессионном тестировании, что повышает эффективность и снижает временные затраты.
3. Оценка охвата тестирования: Количество и содержание тест-кейсов позволяют оценить полноту охвата тестируемого функционала.
4. Систематизация накопления знаний о тестировании продукта: Тест-кейсы служат хранилищем информации о проведенных проверках и обнаруженных ошибках.

Отсутствие технического задания значительно осложняет составление тест-кейсов, так как отсутствует единый источник информации о требованиях к сайту. Однако, существуют подходы, позволяющие минимизировать риски и составить достаточно полный набор тест-кейсов: анализ конкурентов, тесное взаимодействие с заказчиком, использование прототипов и макетов, приоритизация тестирования.

Пример тест-кейса

1. ID: TC\_001

2. Название: Проверка работы формы обратной связи

Предусловия:

1. Открыта главная страница сайта.

Шаги тестирования:

1. Прокрутить страницу вниз до раздела "Контакты".

2. Заполнить все поля формы обратной связи: Имя, Email, Сообщение.

3. Нажать кнопку "Отправить".

Ожидаемый результат:

1. На экран выводится сообщение "Ваше сообщение успешно отправлено".

2. На указанный в форме email адрес приходит письмо с копией сообщения.

Фактический результат:

Статус:

Комментарии:

С. Куликов [3] предлагает достаточно простой алгоритм, позволяющий нам создавать эффективные проверки даже в таких условиях. Он говорит, что, приступая к обдумыванию чек-листа, тест-кейса или набора тест-кейсов, нужно задать себе следующие вопросы и получить чёткие ответы:

* Что перед вами? Если вы не понимаете, что вам предстоит тестировать, вы не уйдете дальше бездумных формальных проверок.
* Кому и зачем оно нужно (и насколько это важно)? Ответ на этот вопрос позволит вам быстро придумать несколько характерных сценариев использования того, что вы собираетесь тестировать.
* Как оно обычно используется? Это уже детализация сценариев и источник идей для позитивного тестирования (их удобно оформить в виде чек-листа).
* Как оно может сломаться, т.е. начать работать неверно? Это также детализация сценариев использования, но уже в контексте негативного тестирования (их тоже удобно оформить в виде чек-листа).

Обращаясь ко всем перечисленным методам составления тест-кейсов в условиях отсутствия ТЗ можно значительно повысить эффективность проведенного тестирования.

# 2.3. Разработка тест-плана

Тест-план является важнейшим документом, регламентирующим процесс тестирования. Он определяет стратегию, подходы, ресурсы и сроки, необходимые для проведения эффективной проверки качества веб-приложения. В условиях отсутствия технического задания (ТЗ) разработка тест-плана приобретает особую важность, поскольку приходится компенсировать недостаток формализованных требований.

Отсутствие ТЗ накладывает ряд ограничений на процесс разработки тест-плана:

1. Неполнота информации о функциональных требованиях: Отсутствие четко сформулированных требований к функциям веб-приложения затрудняет определение полного объема тестирования.
2. Неопределенность нефункциональных требований: Сложно определить критерии производительности, безопасности, удобства использования без четких задач от заказчика.
3. Риск неверной интерпретации требований: Отсутствие формализованного документа увеличивает вероятность разночтений и неверной реализации функционала.

Для преодоления сложностей, связанных с отсутствием ТЗ, необходимо адаптировать процесс разработки тест-плана:

1. Активное взаимодействие с заказчиком: Регулярное общение с заказчиком для выявления его потребностей, ожиданий и приоритетов относительно веб-приложения.
2. Анализ аналогов: Изучение конкурентов и похожих веб-приложений для определения стандартного функционала и выявления лучших практик.
3. Использование прототипов и макетов: Если доступны прототипы или макеты, они служат визуальным представлением ожидаемого функционала и дизайна.
4. Итеративный подход: Разработка тест-плана поэтапно, с возможностью корректировки по мере получения новой информации и уточнения требований

Несмотря на отсутствие ТЗ, тест-план должен содержать следующие разделы:

* Введение: цель и задачи тестирования, краткое описание тестируемого веб-приложения , список документов, используемых при разработке тест-плана (прототипы, макеты, аналоги)
* Объекты тестирования: список функций и модулей веб-приложения, подлежащих тестированию, браузеры и платформы, на которых будет проводиться тестирование
* Стратегия тестирования: описание типов тестирования, которые будут проводиться (функциональное, юзабилити, кроссбраузерное и т.д.), описание подходов к тестированию (ручное, автоматизированное), описание критериев приоритезации тестов
* Ресурсы и сроки: список специалистов, участвующих в тестировании, необходимое оборудование и программное обеспечение, график проведения тестирования
* Критерии завершения тестирования: описание условий, при которых тестирование считается завершенным (например, покрытие тестами всех критически важных функций, исправление всех блокирующих ошибок)
* Риски и проблемы: описание потенциальных рисков, которые могут возникнуть в процессе тестирования (например, изменение требований, нехватка ресурсов), описание плана действий по минимизации рисков
* Приложение: шаблоны тест-кейсов, шаблоны отчетов об ошибках.

Разработка тест-плана при отсутствии ТЗ — сложная, но решаемая задача. Ключевыми факторами успеха являются тесное взаимодействие с заказчиком, тщательный анализ аналогов и гибкость в планировании. Четко прописанный и согласованный тест-план позволит провести эффективное тестирование веб-приложения и добиться высокого уровня его качества, даже при отсутствии полной формализованной документации.

# 2.4. Тест-дизайн и тест-аналитика

Прежде, чем приступить к работе, тестировщик изучает полученные требования на новую фичу (тест-анализ), затем переходит к написанию тестовой документации (тест-дизайну) и только потом, когда разработчик реализует фичу, тестировщик начинает тестировать.

Тест-дизайн — это этап тестирования ПО, на котором проектируются и создаются тестовые случаи (тест-кейсы) в соответствии с определенными ранее критериями качества и целями тестирования.

Условно: требования = критерии качества.

Цели тест-дизайна:

1. максимально покрыть функциональность тестами;
2. обнаружить самые серьезные баги;
3. сократить количество тестов за счет исключения непродуктивных тест-кейсов;
4. не пропустить важные тесты.

Тест-дизайн позволяет провести тестирование наиболее эффективно, чтобы завершить его в установленные сроки. При этом система должна быть покрыта так, чтобы не оставалось функциональностей, не покрытых проверками.

Мы помним, что Исчерпывающее тестирование недостижимо. Вместе с тем благодаря техникам тест-дизайна становится возможным сократить количество тестов без потери качества тестирования.

Автор книги "[A Practitioner’s Guide to Software Test Design](http://testingbooks.ru/a-practitioners-guide-to-software-test-design/)", Lee Copeland, выделяет следующие техники тест-дизайна:

1. Тестирование на основе классов эквивалентности (equivalence partitioning) — это техника, основанная на методе чёрного ящика, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений.

Смысл этого подхода заключается в выборе значений, представляющих различные классы тестовых данных, чтобы мы могли проверить требования к продукту.

Где использовать?

Везде! При тестировании пользовательского интерфейса (UI) - это поля, даты, конкретные кнопки. При тестировании API нам нужно проверить все возможные параметры в теле запроса (body), заголовках (headers), пути (path) или параметрах запроса (query parameters).

1. Техника анализа граничных значений (boundary value testing) — это техника проверки поведения продукта на крайних (граничных) значениях входных данных.

Смысл этого подхода заключается в выборе значений на границах эквивалентных классов с минимальным шагом. И так же ее можно использовать везде.

1. Попарное тестирование (pairwise testing) — это техника формирования наборов тестовых данных из полного набора входных данных в системе, которая позволяет существенно сократить количество тест-кейсов.

Этот подход основан на большом количестве входных параметров. Чем больше параметров, тем больше вероятность ошибки. Наша цель как специалиста по тестированию — сократить количество тест-кейсов до оптимального. Подход парного тестирования поможет нам в этом.

Суть его заключается в том, чтобы рассмотреть все возможные комбинации каждой пары входных параметров.

Ее стоит использовать в том случае, когда входные данные связаны друг с другом. Точнее результат выполнения теста напрямую зависит от того, какие комбинации данных будут подаваться на входе.

1. Тестирование на основе состояний и переходов (State-Transition Testing) — применяется для фиксирования требований и описания дизайна приложения.

Тестирование приложения связано с последовательностью экранов (страниц), созданием/чтением/обновлением/удалением разных типов объектов. Диаграммы состояний могут помочь нам охватить все ветви для таких объектов и экранов.

Смысл данного подхода заключается в создании карты переходов для каждого типа объекта и создании набора тестов, охватывающих все переходы между состояниями.

Его используют прежде всего, при тестировании моделей объектов домена. Этот метод можно применять и к части пользовательского интерфейса, как уже упоминалось ранее. Мы можем охватить все переходы между экранами (страницами) пользовательского интерфейса и создать тестовые случаи, проверяющие переключение между ними.

1. Таблицы принятия решений (Decision Table Testing) — техника тестирования, основанная на методе чёрного ящика, которая применяется для систем со сложной логикой.

Эта техника помогает наглядно изобразить комбинаторику условий из требований. Это помогает нам сократить количество ненужных тестов и предоставить наиболее эффективный набор тестов.

Этот подход помогает нам увидеть зависимости между тестовыми данными и конечным результатом и увидеть, действительно ли нужны ли нам некоторые сценарии, потому что они могут быть покрыты другими.

Таблица решений может описывать сложные правила/требования. Условия — это входные данные, действия — это ожидаемый результат, а столбцы — тестовые примеры.

1. Доменный анализ (Domain Analysis Testing) — это техника основана на разбиении диапазона возможных значений переменной на поддиапазоны, с последующим выбором одного или нескольких значений из каждого домена для тестирования.

Этот подход заключается в систематизации карты влияния на приложение. Знание того, какое условие должно привести к какому-либо отклику, может помочь нам оптимизировать количество тест-кейсов и обеспечить соответствующее покрытие.

Эта техника может быть использована в случаях, когда у нас есть неочевидные зависимости и сложные условия для принятия решения. Она также может быть применена, когда наши действия влияют на хранение данных или другие внешние сервисы. Стоит отметить, что эта техника хорошо сочетается с диаграммами состояний и последовательности.

1. Сценарий использования (Use Case Testing) — Use Case описывает сценарий взаимодействия двух и более участников (как правило — пользователя и системы).

Этот подход основан на вашем предыдущем опыте использования других аналогичных приложений / платформ. Предполагается, что вы знаете некоторые ситуации, которые могут вызвать ошибки и запутать пользователя с неожиданными результатами.

Везде, где пользователь может предоставлять данные, там и используют этот подход. Для UI это могут быть формы, флажки, кнопки и т. д. Для API-сервисов - параметры запроса, параметры пути, параметры тела.

# 2.5. Баг-репорты

В условиях отсутствия исчерпывающего технического задания (ТЗ) на веб-приложение строительной компании, баг-репорты становятся особенно важным инструментом коммуникации между тестировщиком и командой разработчиков. Четкое и информативное описание найденных ошибок позволяет:

* Однозначно интерпретировать проблему: Отсутствие ТЗ может привести к различному пониманию функционала приложения. Четкий баг-репорт исключает двусмысленность в описании ошибки.
* Эффективно воспроизвести и исправить ошибку: Подробное описание шагов воспроизведения бага экономит время разработчиков на поиск и локализацию проблемы.
* Отслеживать жизненный цикл бага: Баг-репорты позволяют отслеживать статус ошибки (новая, в работе, исправлена, проверена) и обеспечивать обратную связь между тестировщиком и разработчиком.
* Собирать информацию о качестве продукта: Анализ баг-репортов позволяет выявить проблемные места в приложении, оценить риски и спланировать дальнейшую разработку и тестирование.

Несмотря на то, что универсального шаблона баг-репорта не существует, эти ключевые элементы должны присутствовать в нем обязательно:

1. Тема (краткое описание): краткое описание проблемы, позволяющее сразу понять суть ошибки.
2. Окружение: информация о тестовом окружении (операционная система, браузер, версия приложения и т.д.).
3. Шаги воспроизведения: детальное пошаговое описание действий, которые приводят к возникновению ошибки.
4. Ожидаемый результат: Описание корректного поведения приложения, основываясь на анализе аналогов, прототипов или общей логике работы.
5. Фактический результат: Описание того, что происходит в действительности при возникновении ошибки.
6. Серьезность (Severity): Оценка критичности ошибки (блокирующая, критическая, значительная, незначительная).
7. Приоритет (Priority): Очередность исправления ошибки (высокий, средний, низкий).
8. Скриншоты/видео: Наглядная иллюстрация ошибки (скриншот, видеозапись).
9. Логи: Файлы логов приложения, содержащие информацию об ошибке.

В условиях отсутствия ТЗ важно обратить внимание на следующие моменты:

* Описывайте ожидаемый результат максимально подробно: Поскольку формальных требований нет, важно четко аргументировать ожидаемое поведение системы.
* Прикладывайте дополнительные материалы: скриншоты, видео и логи приобретают особое значение, так как помогают компенсировать недостаток документации.
* Активно взаимодействуйте с разработчиками: задавайте уточняющие вопросы, участвуйте в обсуждении задач и предлагайте варианты решения проблем.

Грамотно составленные баг-репорты являются незаменимым инструментом в процессе ручного тестирования веб-приложений, особенно в условиях неполной или отсутствующей документации. Четкое и информативное описание ошибок позволяет сократить время на их исправление, повысить качество продукта и обеспечить эффективное взаимодействие внутри команды.

# Глава 3. Практическое исследование разработка предложений по улучшению тестирования веб-приложения строительной компании «AVANT GROUP» в условиях отсутствия ТЗ

# 3.1. Тестирование функциональности

После обширной теории по тестированию веб-приложений мы наконец переходим к практике - тестированию сайта «AVANT GROUP».

Для начала мы ознакомимся с тестируемым продуктом.

«AVANT GROUP» - официальный сайт одноименной строительной компании в Азербайджане. Веб-приложение «AVANT GROUP» довольно объемное, содержит большое количество страниц, наполненных различным контентом.

В условиях отсутствия ТЗ мы понимаем, что полное, исчерпывающее тестирование невозможно. Но на примере данного сайта мы можем показать, что даже в условиях отсутствия ТЗ тестирование осуществимо и довольно эффективно.

Основным функционалом нашего веб-приложения является получение полной точной информации о продукте с целью приобретения недвижимости на выгодных условиях.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить, соответствует ли фактическое поведение веб-приложения ожидаемому результату. То есть проведение функционального тестирования позволяет проверить способность системы в определенных условиях решать задачи, нужные пользователям.

Поскольку у нас нет доступа к требованиям, то мы будем проводить тестирование *по методу черного ящика* (без доступа к коду). Попробуем осуществить тестирование как если бы мы были реальным пользователем, но с элементами исследовательского тестирования (то есть будем одновременно изучать продукт, проектировать тесты и выполнять их).

Приступая к тестированию вспомним основной порядок процесса тестирования:

1. Подготовка к тестированию: изучение и анализ предмета тестирования;
2. Подготовка к тестированию: планирование тестирования;
3. Исполнение тестирования.

В ходе тестирования функциональности сайта **«AVANT GROUP»** мы проверим:

* Основные ссылки и переходы
* Работу форм
* Работу ключевых функций сайта
* Файлы cookie

В данный список стоило бы включить проверку базы данных, однако в условиях отсутствия требований и доступа к базе, мы исключим этот пункт.

На данном этапе подготовки к тестированию должен быть произведен тщательный анализ требований. Но мы работаем в условиях отсутствия требований. Поэтому самостоятельно определяем и формулируем требования к продукту:

* Жизненный опыт
* Здравый смысл
* Общение
* Устоявшиеся стандарты
* Статистические данные
* Авторитетное мнение

Наиболее эффективные техники сбора и выявления требований:

* Семинары и мозговой штурм
* Наблюдение
* Прототипирование
* Анализ документов
* Моделирование процессов и взаимодействий

В условиях, что у нас нет доступа к макетам и коду и возможности уточнить информацию у команды разработки (программиста, аналитика, дизайнера и др.) и у руководителя проекта, мы можем применить, здравый смысл, жизненный опыт и устоявшиеся стандарты. Самостоятельно предположим, каким требованиям должна соответствовать та или иная функциональность сайта строительной компании. Более того, мы проводим тестирование с точки зрения обычного пользователя. Стандарты работы любого веб-приложения предполагают интуитивно понятный интерфейс. Чтобы провести тестирование в ознакомительных целях, нам не обязательно иметь за плечами богатый профессиональный опыт тестировщика - достаточно представить себя на месте пользователя, который решил посетить сайт строительной компании и, желающий получить точную и полную информацию.

Вторым этапом идет подготовка к тестированию: планирование тестирования.

Для составления тестовой документации тестировщики пользуются такими инструментами как Google DOCs для чек-листов, баг-репортов, тест-плана, а так же TMS Qase.io для тест-кейсов.

В нашем плане тестирования будут следующие чек-листы:

1. Чек-лист проверки основных ссылок и переходов на Главной странице
2. Чек-лист для раздела "Проекты""
3. Чек-лист для раздела "О нас"
4. Чек-лист для раздела "Контакты"
5. Чек-лист для раздела "Подписаться на нас"
6. Чек-лист для раздела "360°"
7. Чек-лист “Файлы cookie”

Несомненно, перечисленные чек-листы не обладают достаточным покрытием для качественного тестирования функциональности всего сайта. Мы выбрали только основные проверки для наглядного примера того, что можно протестировать, не имея доступа к требованиям, и это далеко не весь перечень. Помним, что наша задача - практика тестирования без доступа к требованиям, поэтому мы вынуждены ограничить тестирование наиболее показательными на наш взгляд проверками.

**Тест-план.**

Поскольку мы проводим тестирование с учебной целью, то в нашем тест плане некоторые пункты будут опущены. Созданный тест план доступен по ссылке: <https://docs.google.com/document/d/12hwPW2VKGgdVh-vAb3CbBur3XexbVhCgcoASFDIT7j8/edit?usp=sharing>

**Идентификатор тест плана:** 00001.

**Введение:** необходимо провести ручное тестирование веб-приложения без доступа к требованиям с целью выявления дефектов. На основе полученных результатов будут сформулированы рекомендации по улучшению тестирования веб-приложения.

**Объект тестирования:** веб-приложение AVANT GROUP https://avantgroup.az/ .

**Что будет протестировано:**

Функциональное тестирование:

1. Проверка основных ссылок и переходов на Главной странице
2. Проверка раздела "Проекты""
3. Проверка раздела "О нас"
4. Проверка раздела "Контакты"
5. Проверка раздела"Подписаться на нас"
6. Проверка раздела "360°"
7. Файлы cookie

* Тестирование UI и UX
* Тестирование совместимости

**Что не будет протестировано:** нагрузочное тестирование, тестирование безопасности.

**Подходы:** тестирование методом черного ящика. Виды тестирования: функциональное тестирование, тестирование удобства использования, тестирование интерфейса и тестирование совместимости.

**Тестовая документация:** чек-листы.

**Критерии прохождения тестирования:** наличие и серьезность багов; наличие тестовых данных.

**Критерии начала тестирования:** не позднее 01.06.2024.

**Критерии окончания тестирования**: все проверки по созданным чек-листам пройдены, баг-репорты заведены.

**Критерии приостановления и возобновления тестирования:** обнаружение багов, препятствующих дальнейшему тестированию.

**Результаты тестирования:** количество найденных багов, количество багов по степени серьезности, процентное соотношение пройденных и проваленных проверок в каждом чек-листе.

**Требования среды:** компьютер/ноутбук с доступом к сети Интернет, Google Sheets, DevTools, Qase.io, GitHub, Visual Studio Code, Bandicam.

**Обязанности:** Гулу-заде И.Р. - тестировщик, ответственный за проведение функционального, UI, UX, тестирования, тестирования совместимости и составление тестовой документации.

**Расписание:**

| Дата | Вид работ |
| --- | --- |
| 01.06.2024 - 20.06.2024 | Функциональное тестирование: чек-листы, баг-репорты |
| 22.06.2024 - 26.06.2024 | UI и UX: чек-листы, баг-репорты |
| 26.06.2024 | Отчет о тестировании |

**Оценка рисков:** высокая вероятность пропустить баги. В условиях отсутствия требований мы опираемся на логику и здравый смысл, т.е. наше исследование субъективно. В данной ситуации необходимо помнить, что есть риск принять корректное поведение системы за некорректное.

**Составил:** Гулу-заде И.Р.

**Утвердил:** Гулу-заде И.Р.

На этом этап планирования тестирования завершен.

Как мы уже говорили ранее, третий этап “Исполнение тестирования” мы объединим с процессом создания документации - будем одновременно писать тесты и проходить их.

После того, как мы создали план тестирования, можем приступать к написанию тестовой документации и собственно к тестированию.

**Чек-листы**

Для составления чек-листов создаем файл в Google Таблицы (ссылка на документ: <https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-_ug/edit?usp=sharing>

Составили чек-листы исходя из правил:

* Один пункт – одна проверка
* Пункты всегда начинаются с существительного или глагола неопределенной формы
* Соблюдать структуру
* Опираться на требования
* Давать пунктам чек-листа названия по форме, общей для всех членов команды
* Формулировать пункт чек-листа следует в виде утверждения, ответом на которое будет “Да”/”Нет”

Пункты 4 и 5 нам не пригодятся, а вот остальные правила мы обязательно учтем при создании чек-листов.

Каждый чек-лист будет состоять из трех столбцов:

1. Проверка
2. Результат (untested, passed, failed, blocked, skipped)
3. Комментарий

Чек-листы составляются таким образом, чтоб они были универсальны и легко поддерживаемы.

Все проверки пройдены успешно. Ознакомиться более подробно с результатами можно по ссылке: https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-\_ug/edit?usp=sharing

На этом тестирование функциональности веб-приложения AVANT GROUP завершено. Переходим к тестированию UX UI.

# 3.2. Тестирование UI и UX

Тестирование пользовательского опыта (UX) и интерфейса (UI) играет критически важную роль в успехе любого веб-приложения. В случае с сайтом строительной компании, UX/UI тестирование выходит на первый план, так как именно удобство и простота использования сайта будут являться ключевыми факторами привлечения и удержания клиентов.

Отсутствие ТЗ в данном контексте создает как сложности, так и дополнительные возможности:

Сложности:

* Отсутствие четких критериев оценки: Необходимо самостоятельно определить, что считать "хорошим" UX/UI, опираясь на лучшие практики, анализ конкурентов и здравый смысл.
* Риск субъективности: Оценка UX/UI может быть субъективной. Важно привлекать к тестированию не только специалистов, но и представителей целевой аудитории.

Возможности:

* Гибкость и адаптивность: Отсутствие жестких рамок позволяет оперативно вносить изменения в интерфейс и функциональность на основе обратной связи от пользователей.
* Фокус на потребностях пользователя: UX/UI тестирование становится основным инструментом для понимания и удовлетворения потребностей целевой аудитории.

В рамках UX/UI тестирования веб-сайта строительной компании, особое внимание стоит уделить следующим аспектам:

* Навигация: Интуитивность, логичность расположения элементов, простота перемещения между разделами сайта.
* Контент: Понятность, структурированность, релевантность информации для целевой аудитории (например, описание объектов строительства, условия ипотеки).
* Визуальное оформление: Соответствие дизайна сайта тематике строительства, использование качественной графики и фотографий, гармоничное цветовое решение.
* Формы взаимодействия: Удобство и понятность форм обратной связи, заказа звонка, подписки на новости.
* Адаптивность: Корректность отображения и работы сайта на разных устройствах (компьютеры, планшеты, смартфоны).
* Доступность: Соответствие сайта требованиям доступности для людей с ограниченными возможностями.

В рамках данной задачи оценивается:

* Сколько шагов нужно сделать для выполнения задачи?
* Сколько времени требуется на выполнение задачи?
* Сколько ошибок делает пользователь-новичок при выполнении задачи?
* Какое впечатление осталось у пользователя от работы с программой?
* Эмоции пользователя во время выполнения задачи.

При проведении UX/UI тестирования можно использовать следующие методы:

* Эвристическая оценка: анализ сайта экспертом на предмет соответствия принципам юзабилити.
* Тестирование на пользователях: проведение тестирования с привлечением представителей целевой аудитории для получения обратной связи о удобстве использования сайта.
* A/B тестирование: сравнение эффективности разных вариантов дизайна или функционала на основе показателей поведения пользователей.
* Сбор обратной связи: анализ отзывов пользователей, комментариев в социальных сетях, обращений в службу поддержки.

Пример: Как мы проводили тестирование раздела "Проекты"

Раздел: "Проекты" (информация о завершенных, строящихся и планируемых к строительству объектах компании).

Аспекты для тестирования:

* Структура представления информации: логичность группировки объектов, наличие фильтров и сортировки.
* Карточка объекта: наличие всей необходимой информации (фотографии, описание, планировки, цены), удобство просмотра.
* Интерактивные элементы: Возможность заказать звонок, скачать буклет, построить маршрут до объекта.

Методы тестирования:

* Эвристическая оценка: проверка сайта на соответствие лучшим практикам представления информации об объектах недвижимости.
* Тестирование на пользователях: наблюдение за тем, как пользователи ищут нужную им информацию об объектах, какие сложности у них возникают.
* Сбор обратной связи: анализ отзывов пользователей о разделе "Проекты", выявление потенциальных проблем и направлений для улучшения.

После произведенного нами тестирования UX/UI сайта AVANT GROUP мы можем прийти к выводу, что приложение нуждается в доработке, но не критично, поскольку выявленные баги незначительно сказываются на работе сайта.

Итоговый чек-лист в нашем документе идет под названием "3.2 Тестирование UI и UХ":

<https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-_ug/edit?usp=sharing>

# 3.3. Подведение итогов. Оценка эффективности предложенной методики и рекомендации по ее применению.

После проведения полного тестирования мы переходим к самой ответственной части - это написание отчета о тестировании. Отчет о результатах тестирования - это документ, содержащий информацию о выполненных действиях в ходе тестирования, результатах проведённой работы, статистику по дефектам, оценку качества объекта тестирования и рекомендации. Отчет обычно включает в себя таблицы и диаграммы.

Мы проделали большую работу и теперь нам остается подвести итоги.

С отчетом о тестировании можно ознакомиться по ссылке:

<https://docs.google.com/document/d/1AgmC7fkPBqVKmE9pzfY7VBqoDB6wlyWdItIKQwtLfDg/edit?usp=sharing>

Отчет имеет следующую структуру:

* Информация о проекте
* Команда тестировщиков
* Описание процесса тестирования
* Статистика по дефектам
* Качество объекта тестирования
* Рекомендации
* Приложения

### Информация о проекте тестирования

В информации о проекте мы обозначили объект и цель тестирования: Необходимо протестировать функционал веб-приложения AVANT GROUP, а также провести тестирование UI, UX без доступа к требованиям.

### Команда тестировщиков

Описание команды тестировщиков представлено в виде таблицы с указанием списка специалистов, задействованных в тестировании проекта, и закрепленного за каждым участником тестируемого функционала.

### Описание процесса тестирования

Описание процесса тестирования включает в себя:

* Тестовое окружение, на котором проводилось тестирование: Windows 11, Yandex 24.1.5.736 (64-bit).
* Веб-приложение, на котором проводилось тестирование: https://avantgroup.az/ru/ .
* Инструменты, которые использовались во время тестирования: Google Sheets, DevTools, Qase.io, GitHub, Visual Studio Code, Bandicam.
* Указание на техническую документацию: "Техническая документация отсутствует".
* Указание на тестовую документацию, используемую на проекте, и обоснование выбора: "Выбор был сделан в пользу чек-листов, так как в условиях отсутствия технической документации, тест-кейсы с детализированными проверками могут исказить результаты тестирования. Нам не известна логика работы каждой функциональности, задуманная командой разработки веб-приложения AVANT GROUP, поэтому, ориентируясь на жизненный опыт, логику и здравый смысл мы создали чек-листы с общими проверками".
* Особенности проведения тестирования, техники тест-дизайна: "Тестирование проведено с точки зрения реального пользователя с элементами исследовательского тестирования".
* Ссылку на тестовую документацию: "Тестовая документация находится по адресу: <https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-_ug/edit?usp=sharing> ".

### Статистика по дефектам

В статистике по дефектам указана ссылка на баг-репорты, общее количество найденных багов и количество багов по степени серьезности:

"Заведенные баг-репорты можно найти по следующему адресу:

<https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-_ug/edit?usp=sharing>

Из 15 заведенных багов:

* 6 багов имеют серьезность “Major” (незначительный).
* 9 багов имеют серьезность “Trivial” (тривиальный)".

Также указана статистика багов по статусам: "Все баги находятся в статусе “Open”".

Далее приведена сводная таблица по багам, содержащая следующую информацию:

1. ID бага (его уникальный номер)
2. Название баг-репорта
3. Серьезность
4. Статус

Для наглядности одинаковые статусы и серьезность выделены одним цветом.

### Качество объекта тестирования

В данном разделе мы указали сводную таблицу по статусам проверок (в процентном соотношении) по каждой протестированной функциональности. Также добавлены результаты проверок тестирования UI/UX. На основе полученных результатов по обнаруженным дефектам и количеству успешно пройденных проверок, дана общая оценка работы объекта тестирования:

"Исходя из полученных результатов тестирования, можно сделать вывод, что веб-приложение выполняет основные функции успешно. Блокирующих, критических и значительных дефектов не обнаружено. Все заведенные дефекты имеют незначительную или тривиальную серьезность и относятся к UI/UX.

Поскольку тестирование проведено без доступа к требованиям и макетам, необходимо учитывать, что полученные результаты не в полной мере отражают качество объекта тестирования".

### Рекомендации

* Следует актуализировать информацию в разделе “Подписаться на нас”.
* Развить либо создать аккаунты в других социальных сетях и восстановить на них ссылки.
* Выровнять надписи в едином стиле
* При нажатии на “Узнать больше” удобнее было бы читать более подробную информацию о проекте,нежели вписывать свои данные для связи с менеджером
* В русскоязычной версии сайта все надписи и скачанные документы рекомендовано перевести на русский язык.

# 

# Заключение

В современной сфере информационных технологий потребности в качестве программного обеспечения могут меняться практически ежедневно, что может привести к ситуации, когда проект по информационным технологиям не будет сопровождаться подробными техническими спецификациями и явными требованиями к продукту. Многие компании сегодня переходят от традиционных моделей разработки программного продукта в сторону использования гибких подходов. Поэтому для специалистов по тестированию имеет решающее значение умение адаптироваться к постоянным изменениям и постоянное совершенствование навыков в области тестирования в условиях, когда функциональные требования к тестируемому продукту отсутствуют.

В результате проведенного тестирования веб-приложения AVANT GROUP была составлена следующая тестовая документация:

* Тест-план: <https://docs.google.com/document/d/12hwPW2VKGgdVh-vAb3CbBur3XexbVhCgcoASFDIT7j8/edit?usp=sharing>
* Чек-листы и баг-репорты: <https://docs.google.com/spreadsheets/d/1sjGELUxcFxY-BXN0ycxm8HhTf44Ffdtu4jkrzXg-_ug/edit?usp=sharing>
* Отчет о тестировании: <https://docs.google.com/document/d/1AgmC7fkPBqVKmE9pzfY7VBqoDB6wlyWdItIKQwtLfDg/edit?usp=sharing>

Напомним, что целью исследования было изучить особенности ручного тестирования веб-приложений в условиях отсутствия спецификации к программному продукту и разработать предложения по улучшению веб-приложения AVANT GROUP.

Задачи:

1. изучить литературу, касающуюся темы исследования (что такое веб-приложение, его особенности, тестирование веб-приложений).
2. рассмотреть основные виды и методы ручного тестирования веб-приложений
3. ознакомиться с основными принципами составления такой тестовой документации как чек-листы, тест-кейсы, тест планы, баг-репорты
4. составить план ручного тестирования веб-приложения AVANT GROUP
5. выполнить ручное тестирование веб-приложения AVANT GROUP
6. разработать предложения по улучшению веб-приложения AVANT GROUP

Из проделанной работы можно заключить, что цель исследования достигнута, поставленные задачи выполнены. Следует отметить, что в процессе текущего тестирования отсутствовала команда разработчиков. При работе над проектом, где отсутствует детальная спецификация функционала продукта, источниками ключевых требований могут выступать заказчик, разработчики, аналитики и другие участники команды проекта. Также полезную информацию можно получить через обратную связь от конечных пользователей продукта, если он уже выпущен. К сожалению, данным аспектам не было уделено достаточного внимания, однако в исследовании содержится ценная информация о методах источниках тестирования программного продукта в условиях отсутствия доступа к требованиям. Эта информация может помочь начинающему тестировщику развивать навыки работы с продуктом без спецификации проекта и выявлять дефекты.

# Список используемой литературы

* Р. Савин. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах. — М.: Дело, 2007.
* С. Куликов. Тестирование программного обеспечения. Базовый курс. 3-е издание. Версия книги 3.0.11 от 16.07.2021.
* <https://habr.com/ru/post/450282/?ysclid=l8k8gdlt3z992425699>
* <https://habr.com/ru/post/549054/>
* <http://testingbooks.ru/a-practitioners-guide-to-software-test-design/>
* <https://habr.com/ru/articles/740026/>
* Тестирование веб-приложений и сайтов: полное руководство: <https://www.internet-technologies.ru/articles/rukovodstvo-po-testirovaniyu-saytov.html?ysclid=la2r0r7jsi432180726#header-13292-15>
* Лекция 7, ч.1. Тестирование UI и верстки · Курс лекций "Тестирование програмного обеспечения": <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-7-ch1-testirovanie-ui-i-verstki.html>