План тестирования

Интернет-магазин

Подготовлено Мариной А.А.

12.04.2025

Оглавление

ВВЕДЕНИЕ

1.1 Содержание документа

Настоящий документ описывает процесс, стратегию и подходы к тестированию программного продукта «Интернет-магазин», предназначенного для онлайн-продажи товаров. В нем изложены ключевые аспекты организации тестирования, включая объем, методы и ресурсы, необходимые для проверки системы.

1.2 Цель документа

- Спланировать управление тестированием и техническую поддержку тестирования в ходе жизненного цикла разработки Интернет-магазина;
- Организовать контроль процесса тестирования, определить методы регистрации дефектов, вид предоставления отчетов;
- Определить график работ, описать используемую стратегию тестирования;
- Обозначить ресурсы, необходимые для реализации проекта;

1.3 Целевая аудитория

Документ предназначен для последующего использования сотрудниками группы тестирования проекта и служит для понимания целей и содержания работ по тестированию программного продукта, определяет и описывает перечень работ и этапы тестирования приложения.

1.4 Обзор программного продукта

Интернет-магазин представляет собой платформу, предназначенную для продажи товаров онлайн. Продукт предоставляет пользователям интерфейс для просмотра каталога, добавления товаров в корзину, оформления заказов, выбора способа доставки и оплаты заказа онлайн через сторонние сервисы.

1.5 Цели тестирования

Основными целями тестирования являются:

- Выявление проблем, связанных с несоответствием требований к разрабатываемому программному продукту;
- Отслеживание статуса проблем;
- Снижение рисков проекта, связанных с качеством разрабатываемого продукта.

2 ОБЪЕМ ТЕСТИРОВАНИЯ

2.1 Перечень тестируемых компонентов

Описание компанентов системы, подлежащих тестированию представлено в таблице:

Наименование, приложения/компонента	**Наименование группы функций (модуля)**	**Примеры / Комментарии**
Интернет-магазин	Просмотр каталога товаров	Проверка отображения списка товаров, фильтров, сортировки, пагинации.
	Добавление заинтересовавших товаров в корзину	Проверка добавления товаров в корзину, изменения количества, удаления товаров из корзины.
	Оформление заказа	Проверка ввода данных пользователя, выбора адреса доставки, подтверждения заказа.
	Выбор способа доставки	Проверка доступных способов доставки, расчета стоимости, сроков доставки.
	Оплата заказа онлайн через сторонний сервис	Проверка интеграции с платежной системой, сценариев успешной/неуспешной оплаты, возврата средств.

Тестированию на первой итерации подлежат все функции системы как изолировано - тестирование работы функций системы внутри модулей, так и в целом - тестирование взаимодействия модулей системы, выполнение бизнес-процессов в системе.

В процессе проведения работ по тестированию на второй итерации предполагается проведение регрессионного тестирования – проверка исправления дефектов, зарегистрированных на первой итерации, тест новой функциональности, выполнение бизнес-процессов в системе.

2.2 Нетестируемые компоненты

На начальном этапе запуска программного продукта наиболее приоритетным является проверка функциональных требований, в объеме, указанном выше.

3 РЕСУРСЫ

Предполагается, что предоставлен доступ к следующим компонентам

1. Альфа-версии интернет-магазина (frontend и backend) для проведения функционального тестирования, тестирования UI/UX , API (если доступно) и нагрузочного

2. Техническая документация **(**спецификации и требования к системе , API-документация, описание бизнес-логики) для составления чек-листов и тест-кейсов, проверки соответствия функционала документации, выявления несоответствий между документацией и реальным поведением системы.

3. Локальной инфраструктуре для кроссбраузерного тестирования, тестирования адаптивности через эмуляцию устройств в DevTools, логирования ошибок с помощью консоли браузера и сетевых запросов.

А также к сторонним ресурсам, таким как

1. Инструменты для тестирования API (Postman, Insomnia) для проверки API-методов, тестирования обработки ошибок, проверки валидации данных

2. Инструментам для автоматизации тестирования (например, Selenium) для автоматизации повторяющихся сценариев, проверки регрессии после обновлений функционала, генерации отчетов о тестировании.

3. Инструментам для нефункционального тестирования (например, JMeter) для проверки производительности сайта при большом количестве пользователей, тестирования поведения системы при пиковых нагрузках и анализа времени отклика страниц и API.

4. Сервисам для генерации тестовых данных (Mockaroo, Generatedata.com) для создания больших наборов данных, генерации тестовых адресов доставки для проверки всех сценариев.

5. Сторонним платежным сервисам для тестирования сценариев оплаты: успешная оплата, отклонение платежа, возврат средств, проверки обработки ошибок и уведомлений пользователю после транзакции.

6. Сервисам для управления тестированием (Jira, Trello) для ведения тест-кейсов и чек-листов, логирования багов с подробным описанием и отслеживания прогресса тестирования.

4 СТРАТЕГИЯ ТЕСТИРОВАНИЯ

Основным методом проверки данного программного продукта будет ручное тестирование с использованием метода «черного ящика», который базируется на использовании требований и спецификаций, и не предполагает наличия каких-либо специальных знаний о конфигурации и внутренней структуре объекта испытаний.

В процессе тестирования Интернет-магазина будут применяться следующие его типы:

4.1 Функциональное тестирование

Функциональное тестирование – это процесс тестирования программного продукта на предмет соответствия требованиям к программному продукту и правильности реализации всех характеристик, заложенных в продукт. Будет осуществляться вручную. Данный вид тестирования будет включать в себя два этапа:

4.1.1 Минимальный приемочный тест (Smoke test).

Пусть он будет включать в себя следующие проверки:

Просмотр каталога товаров, чтобы убедиться, что он открывается, а список товаров отображается с корректными названиями, ценами и изображениями.

Для этого откроем страницу каталога в браузере, проверим, что список товаров загружается, названия, цены и изображения отображаются без ошибок (например, без "битых" изображений или некорректных цен). И будем использовать для этого тестовую среду интернет-магазина, тестовый набор товаров с заранее заданными названиями, ценами и изображениями для сверки корректности отображения. Все это будем проверять с использованием различных браузеров для проверки отображения страницы. И после проведения теста сверимся с технической документацией.

Добавление товаров в корзину, чтобы убедиться, что он отображается в корзине с правильным названием, количеством и ценой.

Для этого выберем товар из каталога, нажмем кнопку "Добавить в корзину", перейдем в корзину и убедимся, что товар отображается с правильными данными (название, количество 1, цена). Среди использованных ресурсов: тестовая среда интернет-магазина, тестовые данные товара (название, цена) для добавления и проверки, различные браузеры для проверки отображения корзины, а также техническая документация.

Оформление заказа, чтобы убедиться, что форма заказа доступна, и пользователь может ввести базовые данные (например, имя и адрес).

Для этого перейдем к оформлению заказа из корзины, проверим, что форма заказа открывается, поля для ввода имени и адреса доступны, и данные можно ввести без ошибок.

Из ресурсов: доступ к форме заказа на frontend, тестовые данные, состоящие из тестового пользователя с личными данными для заполнения формы, браузер для открытия формы и ввода данных, техническая документация для проверки, что обязательные поля соответствуют требованиям.

Выбор способа доставки, можно выбрать один из них? стоимость доставки отображается корректно?

В процессе оформления заказа проверим, что список способов доставки отображается, выберем один вариант и убедимся, что стоимость доставки обновляется и отображается корректно.
Нам понадобится тестовая среда интернет-магазина (доступ к разделу выбора доставки в процессе оформления заказа, набор способов доставки с указанием стоимости для проверки, браузер для выбора способа доставки и проверки отображения стоимости, техническая документация для сверки доступных способов доставки и правил расчета стоимости.

Оплата заказа онлайн через сторонний сервис, чтобы удостовериться, что при выборе оплаты открывается страница стороннего сервиса (тестового), и система корректно перенаправляет пользователя.
На этапе оплаты выберем способ оплаты через сторонний сервис, проверим, что система перенаправляет на тестовую страницу сервиса, и страница загружается без ошибок.
Будем использовать доступ к модулю оплаты на frontend, сторонний платежный сервис для проверки перенаправления и отображения страницы оплаты, браузер для выполнения оплаты и проверки перенаправления, техническую документацию для подтверждения корректности интеграции с платежным сервисом и сервисы управления тестированием для логирования результатов теста и регистрации дефектов, если перенаправление не работает.

Критерии прохождения МПТ (Минимальный приемочный тест): Все перечисленные шаги выполняются без критических ошибок (например, падения системы, недоступности ключевых страниц или функций). Если хотя бы одна из функций не работает, версия не допускается к дальнейшему тестированию до устранения дефектов.

4.1.2 Регрессионное тестирование.

Для обеспечения надлежащего качества продукта на каждом последующем этапе будет производиться регрессионное тестирование функциональности, разработанной на предыдущей стадии и ее совместимости с новой функциональностью.

Однако в нашем случае количество проверяемых модулей небольшое и поэтому оно будет дублировать минимальный приемочный тест

Поэтому тут Ctrl+V оттуда

4.1.3 Функциональный тест.

Название	Для чего?	Что используем?
Просмотр каталога товаров	Проверка корректности отображения списка товаров, фильтров, сортировки и пагинации при стандартных сценариях; тестирование нестандартных	Доступ к frontend для проверки отображения каталога и работы фильтров/сортировки. Тестовые данные, такие как наборы товаров (стандартные и с большим объемом, например, 1000+ товаров) для проверки пагинации и нагрузки; некорректные значения (например, отрицательные числа для фильтров).

	сценариев (некорректные значения в фильтрах, поведение при большом объеме данных).	Несколько разных браузеров для проверки отображения; инструменты для анализа скорости загрузки при большом объеме данных. Сервисы для генерации тестовых данных, например, Generatedata.com, для создания больших наборов данных
Добавление товаров в корзину	Проверка добавления, изменения количества и удаления товаров в стандартных условиях; тестирование граничных значений (максимальное количество товаров) и некорректных данных (специальные символы).	Доступ к frontend для взаимодействия с каталогом и корзиной. Тестовые данные в виде товаров с разными характеристиками; данные для граничных значений и специальные символы (SQL-инъекции и др.). Разные браузеры для выполнения действий и проверки обновления корзины. Техническая документация для сверки ожидаемого поведения
Оформление заказа	Проверка ввода данных пользователя и подтверждения заказа в стандартных условиях; тестирование с некорректными данными (специальные символы) и граничными значениями (максимальная длина полей).	Доступ к форме заказа на frontend. Данные пользователей (корректные и некорректные). Различные браузер для заполнения формы и проверки поведения. Инструменты для проверки уязвимостей в полях ввода
Выбор способа доставки	Проверка отображения доступных способов доставки и расчета стоимости; тестирование поведения при отсутствии способов доставки или выборе для неподдерживаемого региона.	Доступ к разделу выбора доставки в процессе оформления заказа. Набор способов доставки (включая сценарий с пустым списком) и адреса для неподдерживаемых регионов. Различные браузеры для выбора способа доставки и проверки расчета стоимости. Техническая документация для сверки доступных способов доставки и правил расчета стоимости.
Оплата заказа онлайн через сторонний	Проверка успешной оплаты и	Доступ к модулю оплаты на frontend.

сервис	уведомлений в стандартных условиях; тестирование сценариев с некорректными данными (истекшие карты) и сбоями соединения	Сторонний платежный сервис для проверки сценариев оплаты (успешная, неуспешная, сбои). Тестовые данные карт (корректные и некорректные, например, истекший срок действия). Различные браузеры для выполнения оплаты и проверки уведомлений. Сервисы мониторинга и логирования для анализа логов при сбоях соединения с платежным сервисом.
Бизнес-процессы и взаимодействие	Проверка полного цикла (от просмотра каталога до оплаты) с акцентом на корректное обновление данных (итоговая сумма с учетом доставки) и завершение операций; тестирование логических цепочек (например, изменение способа доставки).	Доступ ко всем модулям для проверки полного цикла. Комплексные сценарии (например, заказ с несколькими товарами, разными способами доставки и оплатой). Различные браузеры для выполнения полного цикла и проверки обновления данных. Сервисы управления тестированием для логирования результатов и регистрации дефектов в бизнес-процессах.

Общий процесс: Тестирование проводится вручную с использованием указанных ресурсов, чтобы обеспечить полное покрытие функционала, включая стандартные и нестандартные сценарии, а также бизнес-процессы. Результаты фиксируются в Trello или Jira для дальнейшего анализа.

4.2 Автоматизированное тестирование.

Для UI тестов Selenium (также для кроссбраузерного тестирования)

Для проверки запросов между frontend и backend Postman**.**

Для генерации тестовых данных Mockaroo или Faker (библиотека)

4.3 Техническое тестирование

Техническое тестирование для проверки нефункциональных требований, преъявляемых к программному продукту, будет осуществляться с использованием прикладного программного обеспечения Mercury Load Runner и производится техническим тестировщиком.

Данный вид тестирования включает в себя несколько типов тестов:

- Нагрузочное тестирование (используя Micro Focus LoadRunner для эмуляции большого числа пользователей и ,например, JMeter для нагрузочного тестирования)
- Тест производительности (Micro Focus LoadRunner+ JMeter+ Gatling)
- Тестирование работоспособности системы при больших объемах данных (Micro Focus LoadRunner+ JMeter+ Postman+Mockaroo)
- Стресс тесты - проверка работоспособности системы в критических условиях (Micro Focus LoadRunner+ JMeter+ k6+Stress-ng)
- Измерение временных параметров работы системы(Micro Focus LoadRunner+ JMeter+ Gatling+Chrome DevTools)
- Восстановление работоспособности системы при сбоях(Micro Focus LoadRunner+ JMeter+ Chaos Monkey+ Docker)

Подробное описание этапов проведения технического тестирования приведено в документе [4].

5 МЕТОДИКА ТЕСТИРОВАНИЯ

5.1 Обзор процесса

Процесс тестирования включает в себя следующие этапы:

1. Получение требований к программному продукту (предоставляется заказчиком);
2. Анализ требований, определение целей и постановка задач по тестированию (определяется в данном документе)
3. Разработка проектной документации:
    a. план тестирования:
        i. определение тестовой стратегии;
        ii. выявление необходимых ресурсов для выполнения проекта;
        iii. определение тестовой конфигурации (необходимое оборудование и прикладное программное обеспечение);
        iv. расчет трудоемкости выполнения задач;
        v. составление предварительного графика тестирования;
    b. написание Test Cases для проведения функциональных тестов, разработка тестовых сценариев и наборов тестовых данных для осуществления технического и автоматизации функционального тестирования;
4. Проведение минимального приемочного теста (МПТ) и принятие решения о приемке версии продукта для тестирования или отказе (в случае принятия решения об отказе от тестирования ожидается сборка очередной версии приложения с исправленными критическими дефектами в виду выпуска патчей либо очередной версии продукта).
5. Регрессионное тестирование - проверка и валидация исправленных дефектов. Основной целью регрессионного тестирования является проверка того, что ошибки, найденные в предыдущих версиях программного продукта исправлены, и, что при их исправлении не внесены новые.
6. Проведение полного цикла функциональных тестов; регистрация дефектов.
7. Получение группой тестирования информации об изменениях в новой версии продукта, анализ новой функциональности.
8. Повторный цикл тестирования.
9. При выпуске кандидат-релиза, выполнение полного приемо-сдаточного тестирования.

5.2 Управление дефектами

Средством регистрации и отслеживанием текущего состояния дефектов для данного проекта выбран инструмент Jira, Trello и\или MS Word.

Определение степени важности дефектов

Важность дефекта требуется для правильной оценки степени влияния дефекта на корректность выполнения операций конечным пользователем.

Различают следующуе степени важности дефектов:

- «**Критический**» (Critical) – дефект, блокирующий большую часть функциональности, полностью нерабочая функция, ветвь функциональности или дефект приводящий к потере информации;
- «**Серьезный**» (Major) – дефект приводящий к некоректной работе части функциональности, недоступности опций отдельной функции, но не блокирующий тестирование в целом . Также это может быть блокирующая ошибка для отдельной функции, не оказывающей влияния на работу всей системы или ветви её функционала;
- «**Средний**» (Average) – ошибка характеризуется неправильно работающей неосновной части функциональности, не препятствующая проведению тестирования (т.е. присутствуют методы альтернативного доступа к функциям и их опциям);
- «**Незначительный**» (Minor) – приложение может успешно функционировать, не нарушена бизнес логика, исправление дефекта откладывается на последнюю очередь.

5.2.1 Регистрация дефектов

Перечень необходимых для заполнения полей при регистрации дефекта приведен в следующей таблице:

Название поля	Описание	Перевод
ID	Идентификационный номер дефекта, заполняется автоматически системой	ID
Статус	Текущее состояние дефекта	State
Заголовок	Название дефекта (краткое описание – привязка дефекта к функции)	Headline
Ответственное лицо	Член команды, принимающий решение о дальнейшей разработке ошибки, определяет (или предпринимает) комплекс мер по устранению дефекта	Owner
Важность	Серьезность дефекта, определяется для правильной оценки степени влияния дефекта на корректность выполнения операций конечным пользователем	Severity
Приоритет	Определяется для указания разработчикам на сколько срочно требуется исправление дефекта	Priority
Модуль	Наименование модуля, в котором обнаружен дефект	Module Name
Функция	Функция, в которой допущен дефект	Function
Версия тестируемого	Номер версии программного продукта	Build №

Название поля	Описание	Перевод
программного продукта		
Компонент исправления	Компонент, в котором можно будет протестировать исправление дефекта (заполняется разработчиком, осуществившим исправление дефекта)	Fixed In
Описание	Полное описание дефекта, методы его воспроизведения, влияние на другие поля и функции	Description
Новая заметка	Поле служит для внесения дополнительных сведений по дефекту, облегчения взаимодействия между командами разработчиков и группой тестирования	New note
История заметок	Поле служит для воспроизведения истории внесения заметок	Notes log
История изменений	Данное поле описывает все действия произведенные с дефектом, кто производил, даты изменений	History
Резолюция	Резолюция, выставляемая при закрытии дефектов из промежуточных состояний, неравных Resolved.	Resolution
Присоединенные файлы	Присоединенные файлы для более подробного описания дефекта (скрин-шоты, логи ошибок)	Attachments

5.2.2 Критерии качества продукта

- Программный продукт должен работать в соответствии со спецификациями и требованиями;
- Функциональность должна полностью покрывать все требования, изложенные в руководствах пользователей (в случае разбиения развития программного продукта на итерации, должна быть полностью реализована функциональность, запланированная на итерацию);
- Программный продукт не должен иметь известных дефектов со статусом «Критический» и «Высокий» к моменту поставки программного продукта заказчику.

6 ГРАФИК ТЕСТИРОВАНИЯ

6.1 Краткое изложение временной шкалы

Этапы и сроки тестирования

1. Подготовка к проверке основных функций альфа-версии **(1–2 недели)**

В первые дни будем проводить анализ предоставленных требований и технической документации, определять поддающиеся тестированию компоненты и критически важные функции. В следующие - разработаем начальные контрольные списки для smoke-тестирования, подготавливать тестовые данные для сценариев каталога, корзины и оформления заказа.

Smoke-тестирование:

- проверка наличия в списке товаров корректных названий, цен и изображений.
- проверка добавление/удаление товаров, проверка количества и итоговой суммы.
- проверка отображения информации о пользователе (Заполните форму с основными данными (имя, адрес)).
- проверка обновления стоимости при доставке.
- проверка перенаправление на сторонний сервис при оплате.

2. **Тестирование разработки, Итерация 1 (3–8 недели)**

В течение 1 недели будем создавать тестовые случаи для каталога, корзины, заказа, доставки и оплаты, включим стандартные, пограничные и недопустимые сценарии с использованием Jira для управления тест-кейсами.

В дальнейшем в течение 3 недель тестировать взаимодействие модулей по возможности с использованием Postman для тестирования API

По возможности разработка скриптов для повторяющихся задач, настройка API-тестов с помощью Postman для регрессии.

3. **Регрессионное тестирование, проверка новые функции, Итерация 2 (9-12 недели)**

В течение следующих 2 недель будем проводить повторный запуск smoke-тестов и функционального теста, постараемся использовать автоматизированные скрипты, проверим и в случае чего исправим дефекты, зарегистрированные в Итерации 1.

На следующей неделе протестируем любые добавленные дополнительные функции (на основе обновлений от разработчиков) и при необходимости обновим тестовые случаи.

В оставшееся время (неделю) проверим не встречается ли на полном пути пользователя ошибки и согласованность данных

И еще 3 дня понадобится для тестирования производительности

4. **Тестирование релиз-кандидата (13–14 недели)**

Проверка релиз-кандидата на готовность к развертыванию.

Для этого проведем полное приемочное тестирование в течение 1 недели, выполним все тестовые случаи и проверим на соответствие требованиям и пользовательским сценариям.

Потом еще 2 дня для окончательной проверки дефектов, в случае если найдем зарегистрируем чтобы потом после релиза исправить

5. **Поддержка после релиза (15–18 недели)**

В оставшиеся недели будем собирать обратную связь от пользователей, регистрировать новые дефекты и решать их в зависимости от приоритета. Повторно запустим тесты производительности, чтобы убедиться в стабильности работы при реальной пользовательской нагрузке.

7 ОЦЕНКА РИСКОВ ПРОЕКТА ТЕСТИРОВАНИЯ

Риск	Вероятность наступления	Влияние на проект	Предварительные меры
Изменение функциональных требований на завершающем этапе проекта	Низкая	Высокое	Ответственный за риск: Заказчик. Предварительные меры: тестовая документация должна иметь адаптивный характер, предусмотреть возможность быстрого внесения изменений (новых требований по проекту) и минимизирования затрат на проведение дополнительных тестов.
Программный продукт имеет известные дефекты со статусом «Критический» и «Высокий» к моменту поставки программного продукта заказчику	Низкая	Высокое	Ответственный за риск: Команда разработчиков. Предварительные меры: своевременная нотификация группой тестирования команды разработчиков обо всех новых дефектах, идентифицированных при проведении функциональных и технических тестов.
Отставание от графика работ	Низкая	Среднее	Ответственный за риск: группа тестирования / команда разработчиков. Предварительные меры: чёткое планирование временных затрат на выполнение поставленных задач, своевременная нотификация заказчика обо всех проблемах, возникающих в процессе тестирования и имплементации исправлений по дефектам и внедрения новой функциональности.
Поставленный для проведения тестирования	Низкая	Высокое	Ответственный за риск: Команда разработчиков. Требуется оперативная

программный продукт имеет дефекты блокирующие основную функциональность			реакция со стороны команды разработки для предоставления исправлений.
Не налажен процесс взаимодействия команд разработчиков и тестирования	Низкая	Среднее	Ответственный за риск: Менеджерский состав команд разработчиков и тестирования. Требуется: предварительное согласование постановки процесса взаимодействия между командами, обсуждение условий поставки следующих версий программного продукта, предварительная высылка нотификаций, сопровождение необходимой проектной документацией
Возможность обнаружения ошибок на стороне заказчика после проведения тестирования	Низкая	Высокое	Ответственный за риск: группа тестирования. Методы предупреждения: Перед проведением тестирования производится согласование с заказчиком тестовых сценариев. При этом заказчиком производится оценка степени покрытия тестами программного приложения.

8 ОТЧЕТЫ О РЕЗУЛЬТАТАХ ТЕСТИРОВАНИЯ

Отчеты о ходе тестирования будут высылаться еженедельно (в конце рабочей недели) и включать в себя перечень активностей команды тестирования за прошедшую неделю (по соглашению с заказчиком возможно дополнительно предоставление списка артефактов для доставки на следующей неделе).

8.1 Отчет о зарегистрированных дефектах

Данный отчет дублирует информацию о зарегистрированных дефектах и служит для удобства заказчика и разработчиков. Документ создается в формате Excel и содержит следующие столбцы:

ID	Состояние	Модуль	Название	Описание	Owner	Важность	Приоритет
№ дефекта	Текущее состояние дефекта	Модуль в котором обнаружен дефект	Краткое описание дефекта	Полное описание дефекта	Текущий владелец дефекта	Важность дефекта	Приоритет исправления