Тест-план

1.Введение

1.1 Основная информация

Документ описывает методы и подходы к тестированию, которые будут использоваться тестировщиком отдела тестирования, для тестирования web-приложения KazanExpress.

KazanExpress — российский маркетплейс, основанный в 2017 году. По количеству заказов за сутки занимает 5-е место среди маркетплейсов в России.

План тестирования может использоваться как тестировщиком, так и менеджерами, разработчиками.

Объект тестирования – это функциональная работоспособность web-приложения KazanExpress, направленная на получение прибыли со стороны Заказчика и удовлетворение потребностей со стороны клиента.

1.2 Цель

Тест-план проекта преследует следующие цели:

1. Определить существующую информацию о проекте и описание того, что нужно Заказчику.
2. Описать стратегии тестирования, которые будут использоваться.
3. Определить необходимые ресурсы для проведения работ по тестированию.
4. Провести результаты тестирования.

Результаты будут представлены в виде отчетов (Qase). Все найденные ошибки будут отслеживаться с помощью баг-трекинговой системы (Jira)

2. Область тестирования функциональность web-приложения KazanExpress 24/7.

Работоспособность production версии web-приложения KazanExpress 24/7, а именно авторизация, корректное сохранение информации о клиенте (контактные данные), корректное отображение изображений, удобство использования web-приложения, смена языка, для разных национальностей, удобство использования для людей с ограниченными возможностями.

3.Тест-план и стратегии тестирования-тест-кейсы пройдены и цели закрыты.

* 1. Классический набор методик тест-дизайна, которые можно применять при тестировании веб-приложений:

- разбиение на классы эквивалентности; (traceability matrix)

- попарное тестирование; (кроссбраузерное тестирование)

- тестирование переходов состояний;

- анализ граничных значений;

- тестирование с помощью таблиц решений;

* 1. Основные типы тестирования, которые должны быть выполнены:

Чек-лист для тестирования WEB приложений состоит из шести разделов:  
-функциональное тестирование (кнопки, открытие изображений, формы ввода, поиск и т.д.)  
-интеграционные (реклама, API, веб сервисы и т.д.)  
-локализация/глобализация (смена языка, дата и время, номера телефонов для разных стран, местоположение)

-UI/UX (расположение кнопок, удобство использования, метод 3-х кликов, отображение содержимого на странице)  
-кроссбраузерность/кроссплатформенность (тестирование поведения web приложения/web сайта на разных браузерах. Браузеры: Firefox, Chrome, Edge (Microsoft Explorer), Safari.)

* 1. Сообщение об ошибках

Отчеты об ошибках создаются для того, чтобы предоставить команде разработчиков и руководителю проекта исчерпывающую информацию об обнаруженных ошибках.

Каждый баг-репорт содержит следующую информацию о дефекте будет представлен баг-трекинговой сиcтеме Jira:

* Название Баг-репорта
* Краткое описание, представляющее собой краткое описание проблем
* Предусловие
* Шаги для воспроизведения ошибки
* Ожидаемый результат
* Фактический результат
* Дополнительная информация о дефекте в виде прикрепленных скриншотов или видеозаписей
* Браузер, в котором проводились тесты

Обязательным условием при составлении баг\*репорта-это приоритезация задачи:

**Приоритет**— срочность, с которой требуется выполнить задачу. По умолчанию все задачи создаются со средним приоритетом (***Medium***).

* ***Blocker*** *—* используется, если обнаружена ошибка, которая мешает компании получать доход. Например, не оформляется заказ на сайте/не работает телефония/не проходит регистрация клиента на сайте/не отправляется или не приходит смс/письмо об оформленном заказе. Такая задача должна быть решена в течении рабочего дня. **Разработчики все бросают и делают данную задачу**.
* ***Highest***— такие задачи должны быть решены **в течении текущего спринта**(неделя-две).
* **High** — срочная, но не критичная задача. Задачу должны взять **в работу в следующем спринте**.
* **Medium**— средний приоритет. Задача должна быть взята в работу после выполнения задач с более высоким приоритетом и в соответствии с датой создания задач со средним приоритетом (от старых к новым).
* ***Low***или ***Lowest****—*если задача может подождать неограниченное время, т.е. совсем не срочна.

4. Ресурсы

4.1 Инструменты

|  |  |
| --- | --- |
| Наименование процесса | Инструмент |
| Баг трекинговая система | jira |
| Тест-кейсы | Qase |
| Выполнение автотестов API | Collection Postman |

4.2 Список браузеров

Chrome Версия 111.0.5563.65 (Официальная сборка), (64 бит)

Firefox Browser 111.0

Safari используются последние версии

4.3 Роли и ответственность

SCRAM MASTER обязанности - следит за соблюдением скрам-процессов, коммуницирует с членами команды, планирует с ними спринты, проводит ежедневные встречи — стендапы, участвует в формировании бэклога, предоставляет участникам процесса необходимые инструкции и так далее.

PRODUCT OWNER обязанности:

* разрабатывает Цель Продукта и делает ее явной;
* создает и четко объясняет Элементы Бэклога Продукта;
* упорядочивает Элементы Бэклога Продукта;
* обеспечивает прозрачность, доступность и понимание всеми Бэклога Продукта.

DEVELOPER TEAM:

FRONT-END DEVELOPER обязанности:

* верстка приложения с помощью HTML/CSS;
* разработка логики компонента или приложения;
* продумывание архитектуры приложения;
* поиск и оценка решений;
* чтение документации (в том числе на английском);
* рефакторинг и оптимизация текущего функционала;
* настройка и оптимизация сборки проекта;
* верстка email;
* другие задачи.

BACK-END DEVELOPER - это специалист, который занимается программно-административной частью веб-приложения, внутренним содержанием системы, серверными технологиями — базой данных, архитектурой, программной логикой.

DESIGNER обязанности:

создает идею и разрабатывает макет сервера;

обеспечивает наилучшее восприятие web-документа на мониторе;

определяет правила компоновки web-страниц;

работает с узлом Internet;

управляет гипертекстовыми документами;

пишет программные части и коды страниц;

создает интерактивные web-приложения;

обновляет и модернизирует web-документы;

проводит информационную политику компании в WWW (World Wide Web) и пр.

ANALYTIC (Обязанности веб-аналитика):

* Установка и настройка основных инструментов, в частности, Google Analytics и Яндекс. ...
* Сбор статистики, составление прогноза и советов по улучшениям;
* A/B-тестирование сайтов;
* Формирование отчетности и дашбордов;
* Поиск KPI сайта, разработка рекомендаций для модернизации и расширения бизнеса.

DEV-OPS SPECIALIST обязанности- Этот специалист занимается автоматизацией на всех этапах: от разработки, сборки и тестирования до развертывания, выпуска и доставки продукта пользователям. Среди обязанностей DevOps — налаживание взаимодействий между разработчиками, системными администраторами, тестировщиками и менеджерами

WEB DEVELOPER Основные обязанности:

* Пишет код. Под каждый проект используется свой язык программирования (JavaScript, PHP, Python и др.). ...
* Работает с серверной частью: поступление запроса, поиск информации в базе данных, сбор статистики и др. ...
* Тестирует, вносит исправления.

QA-ENGINEER обязанности- специалист, который следит за качеством продукта на всех этапах его разработки. В современных реалиях работа QA-инженера начинается ещё на стадии написания технической документации: он тестирует её и проверяет требования к продукту на наличие ошибок, тем самым помогая компании экономить на их исправлении.

5.Критерии качества

«решение» должно работать в соответствии с требованиями и техническим заданием (исходными данными).

* Приемочные критерии - успешное прохождение тест кейсов по разработанной документации (100-90%, 90-80%,80-70%)
* Критерии качества – какой продукт и с каким количеством багов можно выпускать решается на собрании после каждого спринта.
* Критерии начала – авторизация клиента web-приложения KazanExpress
* Критерии приостановления/возобновления тестирования – обнаружение и исправление дефектов (отказ в доступе при регистрации/входе, кол-во посетителей превышает 1000 000)
* Критерии окончания тестирования успешное выполнение 80% запланированных тес-кейсов.

6.Риски процесса тестирования

* Непроработанные до конца требования, что влияет на проведение регрессионного тестирования (Невозможно провести оплату, сохранить карту, некорректно отображаются местоположение на карте, что приводит некорректному выбору пунктов доставки, отсутствует информация о допустимом объеме, кол-ве заказов на 1 клиента).
* У тестировщика мало опыта.
* Разработчики долго чинят баги после тестирования.
* Разработчики делают слишком много багов.
* Разработка в параллель с тестированием.
* Постоянные изменения в требованиях.

7.Документация

Требования:

1. Проверить вход/выход из web-приложения KazanExpress

2. Изменение контактных данных (телефон, электронная почта)

3. Добавление в корзину товаров/товаров

4. Отображение баннеров на главной странице сайта (размеры, шрифты, функционал)

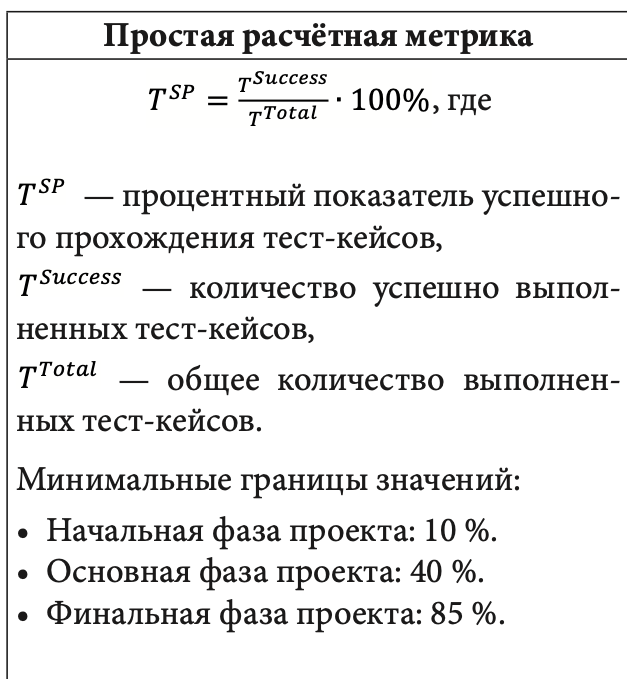
5. Отображение акционных товаров, скидок

Дополнительная информация по Qase:

Сценарии в прогоне тест -кейсов могут иметь следующие статусы:

* **Untested** — еще нет результатов по проверке
* **Passed** — успешно пройден
* **Failed** — обнаружен баг при прохождении / заведен дефект
* **Blocked** — дальнейшее прохождение заблокировано (например, дефектом в другом кейсе
* **Retest** — тест кейс нужно проверить еще раз
* **Skipped** — временно пропущен

8.Метрикит- при запросе команды



9.Отчет о результатах тестирования – краткое описание основные достижения результаты, проблемы, выводы и рекомендации ( при запросе):

-описание процесса тестирования

-статистика по новым дефектам

-список новых дефектов

-статистика по всем дефектам