МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра информационных систем и технологий

Отчет к практической работе №1

по курсу «Обеспечение качества программного обеспечения»

Выполнили:

Немова Н. А.

Иваев Д. Ш.

Сластунов Я. Г.

Поздеев С. А.

группа 3204-090401D

Проверил:

Тимофеев А. В.

Самара 2024

**Sustain Buy**

Маркетплейс для покупки экологичных и биоразлагаемых товаров с интегрированным калькулятором углеродного следа и рекомендациями по компенсации вреда окружающей среде через экологические проекты.

**Стратегия тестирования**

**История изменений**

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Версия | Автор | Описание |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Содержание**

[1. Введение 3](#_Toc114341430)

[2. Подход к тестированию 4](#_Toc114341431)

[3. Тестовое окружение 5](#_Toc114341432)

[4. Инструменты тестирования 6](#_Toc114341433)

[5. Управление выпуском 7](#_Toc114341434)

[6. Анализ рисков 8](#_Toc114341435)

[7. Согласование и подписи 9](#_Toc114341436)

# Тезаурус

* **Sustain Buy** – Маркетплейс, предлагающий экологичные и биоразлагаемые товары с функцией расчета углеродного следа и рекомендациями по его компенсации через эко-проекты.
* **QA Lead** – Лидер команды тестировщиков, отвечающий за разработку стратегии тестирования, контроль сроков и результатов.
* **Project Manager (PM)** – Менеджер проекта, ответственный за координацию проекта, управление ресурсами и соблюдение сроков.
* **Technical Lead (TL)** – Руководитель технической части проекта, отвечающий за качество кода, архитектурные решения и взаимодействие между командами разработчиков.
* **Тестирование совместимости** – Проверка корректной работы системы на различных устройствах, платформах, операционных системах и браузерах.
* **Функциональное тестирование** – Проверка системы на соответствие заявленным требованиям и её функциональным возможностям.
* **Регрессионное тестирование** – Тестирование, проводимое для проверки того, что новое изменение в коде не повлияло на уже работающие функции системы.
* **Нагрузочное тестирование** – Проверка производительности системы при увеличении числа пользователей или нагрузки на серверы.
* **Тестирование безопасности** – Поиск уязвимостей в системе, которые могут быть использованы для взлома или утечки данных.
* **Автоматизированное тестирование** – Тестирование с использованием программных инструментов (например, Selenium, JMeter), что позволяет автоматизировать выполнение тестов.
* **CI/CD (Continuous Integration/Continuous Deployment)** – Непрерывная интеграция и развертывание, автоматизация процессов сборки и тестирования новых версий кода.
* **Тестовое окружение** – Среда, в которой проводится тестирование (разработка, тестирование, продакшн). Настройки и конфигурации должны быть максимально приближены к реальной эксплуатации.
* **Mock-сервисы** – Виртуальные имитации сервисов, которые используются для тестирования интеграций, когда реальные сервисы недоступны.
* **Smoke-тестирование** – Поверхностное тестирование для проверки работоспособности основных функций перед началом более детальных проверок.
* **OWASP ZAP** – Инструмент для тестирования безопасности веб-приложений и обнаружения уязвимостей.
* **Postman** – Инструмент для тестирования API, создания запросов и анализа их выполнения.
* **GitLab CI/CD** – Система автоматизации тестирования и развертывания кода с поддержкой интеграции и непрерывного тестирования.

# Введение

* Кто будет проверять документ? QA Lead
* Кто будет утверждать документ? Project Manager и Technical Lead: гарантируют, что стратегия тестирования соответствует общей цели проекта и техническим требованиям.
* Действия по тестированию программного обеспечения с указанием сроков:
* Анализ требований (1 неделя):

QA команда анализирует требования к проекту, чтобы разработать тест-кейсы и определить основные сценарии использования.

* Разработка тестов (2 недели):

Подготовка тест-кейсов, планов тестирования и стратегии автоматизации. Разработка тестов для ручного и автоматического тестирования.

* Настройка тестового окружения (1 неделя):

Развертывание необходимого окружения для тестирования, включая создание тестовой базы данных, конфигурацию серверов и настройку инструментов.

* Тестирование (ручное и автоматизированное) (3 недели):

Проведение различных этапов тестирования: функциональное тестирование, тестирование на совместимость, нагрузочное тестирование, регрессионное тестирование и другие.

* Анализ результатов и исправление багов (2 недели):

Сбор и анализ данных тестирования, отчетность по найденным багам и проблемам. Исправление дефектов командой разработчиков и повторное тестирование.

* Заключительное тестирование и выпуск (1 неделя):

Последний цикл тестирования перед выпуском, включая Smoke-тестирование, чтобы убедиться в готовности продукта к релизу.

# Подход к тестированию

**Подход к тестированию**

**Процесс тестирования:**

Тестирование будет проводиться поэтапно, начиная с анализа требований и завершаясь регрессионным тестированием перед релизом. Основные этапы:

1. **Анализ требований**: QA команда анализирует требования к проекту, чтобы убедиться в их тестируемости и выявить ключевые области.
2. **Создание тест-кейсов**: Разработка сценариев тестирования, охватывающих функциональные и нефункциональные аспекты системы.
3. **Ручное и автоматизированное тестирование**: Проведение тестов, выполнение автоматизированных скриптов.
4. **Регистрация дефектов**: Все найденные баги фиксируются в системе отслеживания с присвоением приоритета.
5. **Повторное тестирование**: После исправления багов проводится повторное тестирование для проверки исправлений.
6. **Регрессионное тестирование**: Проверка, что изменения не нарушили другие части системы.
7. **Заключительный этап**: Оценка стабильности и готовности продукта к релизу.

**• Уровни тестирования:**

1. **Модульное тестирование**: Тестирование отдельных компонентов системы. Ответственные — разработчики и QA инженеры.
2. **Интеграционное тестирование**: Проверка взаимодействия между компонентами системы.
3. **Системное тестирование**: Полное тестирование системы в реальном окружении.
4. **Приемочное тестирование**: Оценка готовности системы для конечных пользователей.

**• Роли и обязанности каждого члена команды:**

**QA Lead**: Управляет процессом тестирования, утверждает тестовую стратегию, мониторит прогресс.  
**QA Engineers**: Разрабатывают и выполняют тест-кейсы, занимаются ручным и автоматизированным тестированием.

**Automation Engineers**: Разрабатывают и поддерживают автоматизированные тесты. **Project Manager**: Следит за сроками выполнения тестирования, разрешает любые организационные вопросы.  
**Разработчики**: Исправляют найденные дефекты, разрабатывают модульные тесты.

**• Типы тестирования:**

**Функциональное тестирование**: Проверка соответствия работы системы требованиям. **Нагрузочное тестирование**: Проверка системы под высокой нагрузкой.  
**Тестирование безопасности**: Выявление уязвимостей в системе.  
**Тестирование производительности**: Оценка скорости работы системы.

**Регрессионное тестирование**: Проверка, что недавние изменения не нарушили старую функциональность.  
**Тестирование совместимости**: Проверка корректной работы системы на различных устройствах и платформах.

**• Подход к тестированию и инструмент автоматизации (если применимо):**

**Подход**: Для функциональных тестов будет использовано как ручное, так и автоматизированное тестирование. Основное внимание будет уделено автоматизации регрессионного и производительного тестирования.  
**Инструменты автоматизации**: Использование **Selenium** для автоматизации UI тестирования, **JMeter** для нагрузочного тестирования и **OWASP ZAP** для тестирования безопасности.

**• Описание процессов:**

**Добавление новых дефектов**: Все найденные дефекты регистрируются в системе отслеживания (например, **Jira** или **Trello**) с детальным описанием, шагами для воспроизведения и скриншотами.  
**Повторное тестирование**: После исправления дефекта разработчиком QA команда повторяет тест, чтобы убедиться в его устранении.

**Сортировка дефектов**: Все дефекты сортируются по приоритету и критичности (блокирующие, критичные, обычные, низкого приоритета).  
**Регрессионное тестирование**: Проводится перед каждым релизом, чтобы убедиться, что внесенные изменения не повлияли на уже работающие функции.

**Завершение теста**: После успешного выполнения всех тестов и исправления критических багов продукт передается на финальное приемочное тестирование.

# Тестовое окружение

* Определите количество требований и настроек, необходимых для каждой среды
* Определить резервное копирование тестовых данных и стратегию восстановления

**• Определение требований и настроек для каждой среды:**

Для проекта **SustainBuy** потребуется несколько тестовых окружений, чтобы гарантировать корректное функционирование приложения в различных сценариях:

1. **Разработка (Development Environment)**:  
**Назначение**: Используется для разработки и первоначального тестирования новых функций.  
**Требования**:

База данных: копия производственной базы данных с анонимизированными данными.  
Конфигурации серверов: минимальные ресурсы для тестирования новых функций. Виртуальные машины или контейнеры для запуска сервисов.

**Настройки**: Поддержка CI/CD (например, с использованием Jenkins или GitLab CI) для автоматического деплоя изменений после каждого коммита.

2. **Тестирование (Testing/Staging Environment)**:  
**Назначение**: Окружение, максимально приближенное к производственному, где проводится полное тестирование, включая регрессионное, нагрузочное и тестирование безопасности.  
**Требования**:

База данных: актуальные данные, синхронизированные с производственной базой, но с маскировкой персональной информации.  
Конфигурации серверов: те же настройки, что и в продакшн-среде, с аналогичным объемом ресурсов (CPU, память, сеть).

Интеграция всех сервисов, используемых в продакшн-окружении (API, платежные системы и т. д.).

**Настройки**: Автоматическое разворачивание последней стабильной версии приложения с использованием CI/CD, подключение к внешним API и микросервисам.

3. **Производственное окружение (Production Environment)**:  
**Назначение**: Финальная среда, где приложение будет доступно пользователям. **Требования**:

Настройки идентичны тестовой среде.  
Обеспечение безопасности данных пользователей и шифрования. Системы мониторинга и логирования (например, Prometheus, ELK Stack).

**Настройки**: Высокая доступность, поддержка резервного копирования и автоматического масштабирования (например, с использованием Kubernetes).

**• Определение резервного копирования тестовых данных и стратегия восстановления:**

**Резервное копирование тестовых данных**:  
Для тестовой и разработческой среды необходимо регулярное резервное копирование базы данных (ежедневные и еженедельные бэкапы).  
Использование автоматизированного инструмента для создания бэкапов (например, **pg\_dump** для PostgreSQL).  
Все тестовые данные должны быть анонимизированы для защиты конфиденциальности пользователей.

**Стратегия восстановления**:

В случае сбоев или повреждения тестовых данных в среде разработки или тестирования необходимо иметь возможность быстрого восстановления с резервных копий.  
Сценарии восстановления должны быть протестированы заранее, чтобы убедиться в работоспособности системы восстановления.  
Автоматизированные скрипты для восстановления данных (например, разворачивание базы данных из бэкапа через **pg\_restore**).  
Периодические тесты процесса восстановления данных (например, ежеквартальные проверки восстановления из резервной копии).

# Инструменты тестирования

**Инструменты автоматизации и управления, необходимые для выполнения тестов**

**• Инструменты автоматизации и управления:**

1. **Инструменты автоматизации тестирования:  
Selenium**: Используется для автоматизации тестирования веб-интерфейса. Поддерживает множество браузеров и платформ. <https://www.selenium.dev/>

<https://www.selenium.dev/downloads/>

Версия 4.24

**Пользователи**: Обычно один или несколько QA инженеров и автоматизаторов.

**Тип**: Открытый исходный код.

**JMeter**: Используется для проведения нагрузочного тестирования и тестирования производительности.

<https://jmeter.apache.org/>

5.6.3 java > 8

**Пользователи**: Один или несколько тестировщиков и инженеров по производительности.

**Тип**: Открытый исходный код.

**OWASP ZAP (Zed Attack Proxy)**: Инструмент для тестирования безопасности приложений.

<https://www.zaproxy.org/download/> 2.15.0

**Пользователи**: Один или несколько специалистов по безопасности и QA инженеров.  
**Тип**: Открытый исходный код.

**Postman**: Используется для тестирования API, создания и выполнения запросов.

<https://www.postman.com/>

**Пользователи**: Один или несколько тестировщиков и разработчиков API. **Тип**: Имеет как бесплатную, так и коммерческую версию.

**Cypress**: Инструмент для автоматизированного тестирования фронтенда, особенно для приложений на JavaScript.

<https://www.cypress.io/>

**Пользователи**: Один или несколько автоматизаторов тестов.

**Тип**: Открытый исходный код с коммерческими функциями.

2. **Инструменты управления тестированием:  
Jira**: Используется для отслеживания дефектов, планирования тестирования и управления задачами.

<https://www.atlassian.com/ru/software/jira>

**Пользователи**: Обычно включает всю команду (QA инженеры, разработчики, проектный менеджер).  
**Тип**: Коммерческий инструмент с различными планами для команд разного размера.

**TestRail**: Инструмент для управления тест-кейсами и планирования тестирования. **Пользователи**: Один или несколько QA инженеров и тест-менеджеров.  
**Тип**: Коммерческий инструмент.

<https://www.testrail.com/>

**GitLab CI/CD**: Для автоматизации процессов сборки, тестирования и развертывания. **Пользователи**: Один или несколько разработчиков и тестировщиков.  
**Тип**: Имеет как бесплатную, так и коммерческую версию.

<https://docs.gitlab.com/ee/ci/>

**SonarQube**: Используется для анализа кода и обеспечения качества кода. **Пользователи**: Один или несколько разработчиков и QA инженеров. **Тип**: Имеет как бесплатную, так и коммерческую версию.

<https://www.sonarsource.com/products/sonarqube/>

**• Количество необходимых инструментов и их планирование:**

**Открытый исходный код:  
Selenium**: Один инструмент, используемый несколькими тестировщиками.

**JMeter**: Один инструмент для нагрузки, используемый одним или несколькими инженерами.  
**OWASP ZAP**: Один инструмент для безопасности, используемый специалистами по безопасности.

**Postman**: Один инструмент для API тестирования, используется несколькими пользователями.  
**Cypress**: Один инструмент для фронтенд-тестирования, используемый автоматизаторами.

**Коммерческие инструменты:  
Jira**: Один инструмент для управления проектом и дефектами, используемый всей командой.  
**TestRail**: Один инструмент для управления тест-кейсами, используемый QA инженерами и тест-менеджерами.  
**GitLab CI/CD**: Один инструмент для автоматизации сборки и тестирования, используется командой разработчиков и тестировщиков.  
**SonarQube**: Один инструмент для анализа кода, используемый разработчиками и QA инженерами.

**Планирование:**

Убедитесь, что все коммерческие инструменты имеют лицензии, соответствующие количеству пользователей в вашей команде.  
Оцените потребности в обучении для пользователей инструментов с открытым исходным кодом и коммерческих решений.

Регулярно проверяйте и обновляйте инструменты, чтобы обеспечить их соответствие текущим требованиям и стандартам тестирования. .

# Управление выпуском

* План управления выпуском с соответствующей историей версий, который обеспечит выполнение тестов для всех изменений в этом выпуске.

**Управление выпуском План управления выпуском:**

1. **Цели управления выпуском:**Обеспечить бесперебойный процесс выпуска обновлений и новых версий. Обеспечить выполнение всех необходимых тестов для каждой версии перед её выпуском.  
Поддерживать четкую историю версий и документировать все изменения.

2. **Процесс управления выпуском:**

**1. Планирование версии:**

**Сбор требований**: Определите, какие функции и исправления должны быть включены в следующий релиз.  
**Оценка рисков**: Оцените потенциальные риски и влияние изменений на существующую функциональность.

**Создание плана релиза**: Составьте план, включая сроки, задачи, ответственных лиц и необходимые тесты.

**2. Разработка и интеграция:  
Кодирование**: Разработчики реализуют запланированные функции и исправления. **Интеграция**: Внедрение изменений в основную ветку разработки с использованием системы контроля версий (например, Git).

**3. Тестирование:  
Модульное тестирование**: Проверка новых функций на уровне модулей. **Интеграционное тестирование**: Проверка взаимодействия новых и существующих функций.  
**Системное тестирование**: Проверка всей системы в целом в тестовом окружении. **Приемочное тестирование**: Проверка, соответствуют ли новые функции и исправления требованиям заказчика.  
**Регрессионное тестирование**: Убедитесь, что изменения не нарушили старую функциональность.

**4. Подготовка к выпуску:  
Документация**: Обновите документацию, включая заметки о релизе и инструкции по установке.  
**Сборка релиза**: Создайте сборку приложения для выпуска (например, сборка артефактов или контейнеров).

**5. Выпуск:  
Развертывание**: Выполните развертывание новой версии в производственной среде.  
**Мониторинг**: Наблюдайте за системой после развертывания для выявления возможных проблем.

**6. Поддержка после выпуска:  
Отслеживание проблем**: Убедитесь, что все проблемы, выявленные после релиза, фиксируются и решаются.  
**Обратная связь**: Собирайте отзывы пользователей для улучшения будущих релизов.

3. **История версий:  
Формат истории версий**: Версии будут нумероваться по принципу MAJOR.MINOR.PATCH (например, 1.2.3), где:

**MAJOR**: Основные изменения и новые функции, которые могут нарушить совместимость.  
**MINOR**: Новые функции, которые обратно совместимы.  
**PATCH**: Исправления ошибок и улучшения, которые не добавляют новую функциональность.

**Документирование изменений**: Все изменения в версиях будут задокументированы в заметках о релизе, которые будут включать:

Описание новых функций. Исправления ошибок.  
Улучшения и обновления. Известные проблемы и решения.

4. **Инструменты и ресурсы:  
Инструменты управления версиями**: Git (GitHub, GitLab, Bitbucket).  
**Инструменты CI/CD**: Jenkins, GitLab CI/CD, CircleCI.  
**Инструменты мониторинга**: Prometheus, Grafana для наблюдения за системой после выпуска.

План управления выпуском поможет гарантировать, что каждая версия проходит необходимые тесты и готова к выпуску, минимизируя риски и обеспечивая высокое качество продукта. Если все устраивает, можем двигаться дальше.

# Анализ рисков

* Перечислите все риски, которые вы можете оценить
* Предоставьте четкий план по снижению рисков, а также план действий в непредвиденных обстоятельствах.

**Анализ рисков**:

1. **Риск недостаточного тестового покрытия:  
Описание:** Недостаточное тестирование функций экосистемы SustainBuy может привести к невыявленным дефектам, которые повлияют на пользовательский опыт и точность расчета углеродного следа.  
**План по снижению:** Обеспечить полное покрытие всех критичных функций, регулярно обновлять тест-кейсы, использовать анализ покрытия тестов для выявления пробелов.

2. **Риск задержек в разработке:  
Описание:** Задержки в разработке могут привести к несвоевременному выпуску функциональности и задержке в выполнении тестирования.  
**План по снижению:** Регулярный мониторинг прогресса, использование гибких методов управления проектом (Agile), установление четких сроков и контроль их соблюдения.

3. **Риск проблем с интеграцией внешних сервисов:  
Описание:** Проблемы с интеграцией сервисов для расчета углеродного следа и экологических рекомендаций могут повлиять на функциональность.  
**План по снижению:** Проведение интеграционного тестирования, использование mock- сервисов для проверки интеграций, обеспечение документации API.

4. **Риск уязвимостей в безопасности:  
Описание:** Возможные уязвимости в системе могут привести к утечке данных пользователей или атаке.  
**План по снижению:** Проведение регулярного тестирования безопасности, использование инструментов анализа уязвимостей (например, OWASP ZAP), внедрение лучших практик безопасности.

5. **Риск проблемы с производительностью:  
Описание:** Приложение может не справляться с нагрузкой, особенно при увеличении числа пользователей и транзакций.  
**План по снижению:** Проведение нагрузочного тестирования, оптимизация кода и архитектуры, использование методов масштабирования.

6. **Риск неполной или неверной информации о продукции:  
Описание:** Некорректные данные о экологичности продуктов могут ввести пользователей в заблуждение.  
**План по снижению:** Валидация данных от поставщиков, регулярные проверки и обновления данных, создание системы подтверждения данных.

7. **Риск отказов в тестовом окружении:  
Описание:** Проблемы с тестовым окружением могут задержать процесс тестирования  
и обнаружение дефектов.  
**План по снижению:** Настройка резервных тестовых окружений, регулярное резервное копирование данных и автоматизация развертывания.

8. **Риск недостатка квалификации команды:  
Описание:** Недостаток знаний или опыта у членов команды может повлиять на качество тестирования и разработки.  
**План по снижению:** Регулярное обучение и развитие команды, привлечение экспертов при необходимости.

**• План действий в непредвиденных обстоятельствах:**

1. **Обнаружение проблемы:  
Шаг 1:** Немедленное уведомление всех заинтересованных сторон о возникшей проблеме.  
**Шаг 2:** Сбор подробной информации о проблеме, включая логи и воспроизведение ошибки.

2. **Оценка влияния:  
Шаг 1:** Определите серьезность проблемы (влияние на функциональность, безопасность и пользовательский опыт).  
**Шаг 2:** Оцените, насколько проблема влияет на сроки и планы релиза.

3. **Разработка решения:  
Шаг 1:** Определите возможные решения или обходные пути для устранения проблемы. **Шаг 2:** Выберите наилучшее решение с учетом влияния на проект и ресурсов.

4. **План действий:**

**Шаг 1:** Назначьте ответственных за устранение проблемы и определите ресурсы.

**Шаг 2:** Установите сроки для исправления проблемы и документируйте действия.

**Шаг 3:** Внедрите временные или постоянные исправления в тестовую среду для проверки.

5. **Мониторинг и проверка:  
Шаг 1:** Следите за процессом исправления проблемы и проверяйте его эффективность.  
**Шаг 2:** Убедитесь, что исправление не вызвало новых проблем.

6. **Документирование и анализ:  
Шаг 1:** Документируйте все действия по устранению проблемы, включая причины и решения.  
**Шаг 2:** Проведите анализ причин возникновения проблемы и разработайте меры по предотвращению аналогичных ситуаций в будущем.

* Если все устраивает, можем перейти к следующему разделу.

# Согласование и подписи

* Все эти действия проверяются и утверждаются бизнес-командой, руководством проекта, командой разработчиков и т. д.
* Сводка изменений обзора должна быть прослеживается в начале документа вместе с утвержденной датой, именем и комментарием.

**Процесс проверки и утверждения:**

1. **Участники проверки:  
Бизнес-команда:** Проверяет, соответствуют ли тестовые стратегии бизнес- требованиям и целям проекта.  
**Руководство проекта:** Утверждает соответствие стратегии управления проектом и ресурсов.  
**Команда разработчиков:** Убеждается, что тестовая стратегия включает все необходимые аспекты для проверки кода и интеграции.  
**QA команда:** Проверяет, что стратегия тестирования охватывает все необходимые типы тестов и подходы.

2. **Процесс согласования:  
Первичное рассмотрение:** Документ стратегии тестирования предоставляется бизнес- команде, руководству проекта, команде разработчиков и QA команде для первоначального обзора.  
**Обсуждение:** Собрание всех заинтересованных сторон для обсуждения возможных изменений и улучшений.  
**Корректировка:** Внесение необходимых изменений в стратегию тестирования на основе полученных отзывов.  
**Утверждение:** Финальная версия документа утверждается всеми участниками проверки.

**Сводка изменений и обзор:**

1. **Сводка изменений:**

**Дата:** [Дата внесения изменений]  
**Имя:** [Имя ответственного за изменения]  
**Комментарий:** [Краткое описание внесенных изменений]

**Пример:  
Дата:** 2024-09-20  
**Имя:** Иван Иванов  
**Комментарий:** Добавлены детали по плану тестирования производительности и обновлены риски с учетом новых требований безопасности.

1. **Обновление документа:**

**Обновление в начале документа:** В начале документа размещается таблица или раздел, где фиксируются все изменения, включая дату, имя ответственного лица и комментарии.

**Пример:**

Сводка изменений и обзор

| Дата | Имя | Комментарий |

| 2024-09-20 | Иван Иванов | Добавлены детали по тестированию производительности и

обновлены риски. |

| 2024-09-15 | Ольга Петрова | Исправлены неточности в разделе тестового окружения. |

Этот процесс обеспечивает прозрачность и отслеживаемость всех изменений в стратегии тестирования, а также гарантирует, что все ключевые участники согласны с содержанием и планом тестирования. Если все устраивает, можем переходить к следующему разделу или дополнительно проработать любые детали.