**1. Введение:**

Тест-план описывает процесс тестирования сервиса Go REST API. Сервис Go REST API предоставляет конечные точки для различных функциональностей, таких как управление пользователями, извлечение данных и их изменение.

**2. В рамках тестирования:**

- Аутентификация и авторизация пользователей.

- Операции CRUD для управления пользователями (создание, чтение, обновление, удаление).

- Конечные точки для извлечения данных.

- Конечные точки для изменения данных.

- Обработка ошибок и коды ответов.

- Меры безопасности, такие как валидация ввода, проверка аутентификации и авторизации.

**3. Вне рамок тестирования:**

- Детальное тестирование интеграций с сторонними сервисами.

- Тестирование нагрузки за пределами базовой оценки производительности.

- Тестирование совместимости с различными окружениями клиентов.

**4. Цель качества:**

Цель тестирования - обеспечить надежность, безопасность и производительность сервиса Go REST API, стремясь к минимальному количеству дефектов и соблюдению отраслевых стандартов.

**5. Роли и обязанности:**

- Инженер по тестированию: отвечает за создание тест-плана, выполнение тестов, отслеживание ошибок и обеспечение общего качества.

- Разработчики: отвечают за исправление выявленных проблем и предоставление необходимой поддержки во время тестирования.

**6. Методология тестирования:**

Тестирование будет проводиться в основном вручную с использованием инструментов, таких как Postman для тестирования конечных точек API. Будут использоваться различные уровни тестирования:

- Модульное тестирование: проверка отдельных функций и компонентов на корректность и правильную обработку данных.

- Интеграционное тестирование: проверка взаимодействия различных компонентов и сервисов, включая операции с базой данных и внешние интеграции.

- Системное тестирование: обеспечение общей функциональности и производительности сервиса API в реалистичных сценариях использования, включая тестирование нагрузки и проверку соответствия документации API.

**7. Триаж ошибок:**

Ошибки будут классифицироваться по уровню серьезности и влиянию на функциональность системы. Критические ошибки, влияющие на основные функции, будут приоритетными для немедленного решения, за которыми следуют существенные и минорные проблемы.

1. Функциональное тестирование:

- Критические ошибки (высокий приоритет):

- Ошибка при создании нового пользователя: Новый пользователь не добавляется в базу данных или не получает подтверждение о успешном создании.

- Ошибка при получении информации о пользователе: Система не возвращает корректные данные о пользователе или возвращает ошибку.

- Серьезные ошибки (средний приоритет):

- Ошибка при обновлении информации о пользователе: Обновления не сохраняются правильно в базе данных.

- Ошибка при удалении пользователя: Система не удаляет пользователя или удаляет неправильного пользователя.

- Минорные ошибки (низкий приоритет):

- Ошибка при аутентификации пользователя: Проблемы с аутентификацией при попытке доступа к конечным точкам.

2. Тестирование производительности:

- Критические ошибки (высокий приоритет):

- Медленное выполнение запросов: Запросы на получение или обновление информации о пользователях выполняются слишком медленно или приводят к таймаутам.

- Серьезные ошибки (средний приоритет):

- Низкая стойкость к высокой нагрузке: Система не справляется с обработкой большого числа одновременных запросов на доступ к данным пользователя.

- Минорные ошибки (низкий приоритет):

- Медленная загрузка при малой нагрузке: Загрузка данных о пользователях занимает слишком много времени даже при низкой нагрузке.

3. Тестирование контента:

- Критические ошибки (высокий приоритет):

- Отсутствие данных в ответе: Неправильный или пустой ответ при запросе информации о пользователях.

- Серьезные ошибки (средний приоритет):

- Неполная информация: Отображение неполных данных о пользователях при запросе.

- Минорные ошибки (низкий приоритет):

- Некорректные данные в ответе: Отображение неверных или некорректных данных при запросе информации о пользователях.

4. Визуальное тестирование:

- Критические ошибки (высокий приоритет):

- Некорректное отображение интерфейса: Ошибки в стилизации или расположении элементов на странице.

- Серьезные ошибки (средний приоритет):

- Неправильное выравнивание: Проблемы с выравниванием текста или элементов интерфейса.

- Минорные ошибки (низкий приоритет):

- Стилистические ошибки: Незначительные ошибки в дизайне, не влияющие на функциональность.

5. Тестирование на краш:

- Критические ошибки (высокий приоритет):

- Краш при некорректном запросе: Система крашится при передаче неправильных данных в запросе.

- Серьезные ошибки (средний приоритет):

- Краш при недоступности API: Обработка ситуации, когда API становится недоступным.

- Минорные ошибки (низкий приоритет):

- Редкие краши: Иногда возникающие краши, которые не сильно влияют на работоспособность сервиса.

**8. Критерии приостановки и возобновления:**

Критерии приостановки:

- Неприемлемое время ответа: Если среднее время ответа от сервиса превышает установленный порог, указывающий на проблемы с производительностью.

- Высокий уровень ошибок: Если процент ошибок в ответах от API превышает допустимый уровень, указывая на серьезные проблемы с функциональностью или стабильностью.

- Неполадки с доступностью: Если сервис становится недоступным для пользователей из-за технических проблем или отказа сервера.

- Обнаружение серьезных уязвимостей безопасности: Если обнаруживаются критические уязвимости, которые могут привести к компрометации данных или нарушению безопасности системы.

Требования для возобновления:

- Функциональная стабильность: Все функциональные возможности должны быть восстановлены и функционировать без критических сбоев или ошибок.

- Пользовательский опыт: Пользовательский опыт должен восстанавливаться до уровня, обеспечивающего удобство использования и удовлетворение потребностей пользователей.

- Производительность: Приложение должно поддерживать оптимальную производительность, обеспечивая быстрые и плавные взаимодействия с пользователем.

- Безопасность и управление доступом: Необходимо восстановить правильную работу механизмов безопасности и управления доступом, чтобы обеспечить защиту данных и предотвратить несанкционированный доступ.

- Видимость данных: Должна быть восстановлена точность и последовательность представления данных для пользователей, с учетом их ролей и разрешений.

**9. Результаты тестирования:**

- Документация тест-плана.

- Чек-листы.

- Отчеты об ошибках с подробным описанием и шагами для воспроизведения.

**10. Ресурсы и требования к окружению:**

- Рабочее окружение с доступом к сервису Go REST API.

- Наборы тестовых данных для различных сценариев.

- Инструменты совместной работы для коммуникации между участниками команды.

- Jira для заведения баг-репортов