Документация для тестирования серверной части приложения

Тест план

# **Перечень работ.**

Перечень функциональных областей API, которые будут подвергаться тестированию:

1. Ввод данных: проверка корректности ввода данных, включая проверку на допустимые значения (например, при некорректном вводе даты или не заполненном обязательном поле должна вернуться соответствующая ошибка).
2. Авторизация в системе: возможность создать аккаунт, авторизоваться и получать данные об авторизованном пользователе.
3. Интерфейс API: можно ли делать запросы прямо на странице API.
4. Совместимость: проверка совместимости с другими приложениями, например, какие данные при подключении к API можно получить через Postman.
5. Безопасность: доступ к закрытым методам только для авторизованных пользователей, получение только информации авторизованного пользователя.
6. Документация: проверка наличия и качества документации, включая понятное описание API и всех методов.

# **Критерии качества.**

1. Надежность: способность программного продукта выполнять свои функции без сбоев в течение определенного времени или количества операций.
2. Безопасность: обеспечение защиты информации и ресурсов от несанкционированного доступа, изменений.
3. Производительность: способность программного продукта эффективно выполнять свои функции в разумные сроки и с минимальным потреблением ресурсов.
4. Масштабируемость: способность программного продукта адаптироваться к увеличению объема данных или количества пользователей без значительного снижения производительности.
5. Удобство использования: уровень удобства и простоты использования программного продукта для пользователя, включая понятный интерфейс, интуитивную навигацию и доступность функций.
6. Сопровождаемость: способность программного продукта быть легко модифицируемым, исправляемым и обновляемым без необходимости переписывания большого количества кода.
7. Поддерживаемость: возможность программного продукта быть поддержанным и обслуживаемым в течение всего его жизненного цикла, включая обновления, исправления ошибок и обучение пользователей.
8. Эффективность: способность программного продукта выполнять свои функции с минимальным использованием ресурсов, таких как память, процессорное время и сетевой трафик.

# **Оценка рисков.**

Таблица 2. Оценка рисков

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Оценка риска | Описание риска | Выход из ситуации |
| Некорректный ввод данных | Высокий | Неправильный ввод данных может привести к ошибочным результатам и недовольству пользователей. | Провести тщательную проверку ввода данных , проверяя возможность ввода пустой строки, null, отрицательных значений (для числовых полей), специальных символов, отсутствие поля в json запросе |
| Нестабильная работа приложения | Средняя | Нестабильная работа приложения может привести к потере данных и недовольству пользователей. | Провести тщательное тестирование на различных устройствах и в различных условиях использования - использовать механизмы отладки и мониторинга для выявления и исправления ошибок |
| Низкая производительность | Средняя | Низкая производительность приложения может привести к недовольству пользователей и потере клиентов. | Оптимизировать код, использовав эффективные алгоритмы - произвести тестирование производительности и мониторинг нагрузки на приложение. |

# **Стратегия тестирования. (GUI & Ux-testing, Positive, Negative, White-box, Black-box).**

Стратегия тестирования – это план действий, который определяет, как будет проводиться тестирование приложения. Она включает в себя выбор методов и инструментов тестирования, а также определение приоритетов и целей тестирования.

GUI & UX-testing – это тестирование интерфейса пользователя и его удобства использования. Оно включает проверку навигации, доступности элементов управления и понятности сообщений об ошибках.

Positive testing – это тестирование, при котором приложение проверяется на корректную работу при вводе правильных данных и выполнении ожидаемых функций.

Negative testing – это тестирование, при котором приложение проверяется на работу при вводе неправильных или некорректных данных.

White-box testing – это тестирование, при котором проверяются внутренние компоненты и функциональность приложения. Оно включает проверку алгоритмов, обработку ошибок и безопасность.

Black-box testing – это тестирование, при котором проверяется работа приложения без знания его внутренних компонентов и функциональности. Оно включает проверку интерфейса пользователя, совместимости с различными операционными системами и браузерами, а также проверку производительности и надежности.

# **Ресурсы.**

Для разработки приложения по определению типа треугольника могут быть задействованы следующие ресурсы:

Человеческие ресурсы: человек, который будут заниматься разработкой приложения. Программисты, тестировщики, дизайнеры интерфейса, менеджеры проекта и другие специалисты. Они будут заниматься написанием кода, тестированием, дизайном интерфейса, управлением проектом и другими задачами.

Механические ресурсы: это оборудование, которое будет использоваться для разработки приложения. Это могут быть компьютеры, ноутбуки, планшеты, смартфоны и другие устройства, на которых будет тестироваться и использоваться приложение.

Технические ресурсы: это программное обеспечение и инструменты, которые будут использоваться для разработки приложения. Это могут быть языки программирования, фреймворки, библиотеки, среды разработки, инструменты тестирования и другие программные продукты. А если конкретнее, наличие среды разработки visual studio для компиляции проекта, наличие системы удаленной версии, на локальном компьютере (для скачивания проекта из системы удаленной версии) или наличие разархиватора, чтобы разархивировать проект.

Финансовые ресурсы: это деньги, которые будут потрачены на разработку приложения. Это может включать зарплату тестировщиков, дизайнеров, разработчиков, покупку оборудования и программного обеспечения, аренду офиса и другие расходы.

Информационные ресурсы: это данные, которые будут использоваться для разработки приложения. Это могут быть данные о типах треугольников, их свойствах и другие информационные материалы, которые помогут разработчикам создать приложение.

Организационные ресурсы: это структура и процессы, которые будут использоваться для разработки приложения. Это может включать управление проектом, планирование, коммуникацию между разработчиками и другими участниками проекта.

# **Метрики**

Точность: Программа должна правильно определять тип треугольника на основе предоставленных данных. Это может быть измерено путем сравнения результатов программы с известными типами треугольников.

Скорость: Программа должна быть быстрой и эффективной в обработке данных.

Надежность: Программа должна быть надежной и стабильной, то есть она не должна вылетать или вызывать ошибки при обработке данных. Это может быть измерено путем тестирования программы на различных наборах данных и проверки ее стабильности.

Использование ресурсов: Программа должна потреблять минимальное количество ресурсов, таких как память и процессорное время.

Простота использования: Программа должна быть простой и интуитивно понятной в использовании.

Поддержка: Программа должна иметь хорошую поддержку, включая документацию, обучающие материалы и возможность обращения за помощью. Это может быть измерено путем опроса пользователей о том, насколько они удовлетворены доступной поддержкой.

# **Расписание и ключевые точки.**

Таблица 3. Расписание и ключевые точки.

|  |  |
| --- | --- |
| **Этап** | **Количество пар** |
| Разработка тестового набора и тест-кейсов | 1 |
| Реализация 10 unit-test к API | 1 |