**Опишите общий процесс тестирования отдельного модуля SPA приложения.**

Не совсем ясен вопрос , если это не автоматизированное тестирование, то так же как и любое веб приложение, если речь идёт о модульном тестирование то, тестируются отдельные компоненты кода.

1. Планирование тестирования: определение целей тестирования, составление плана тестирования, выбор методов и инструментов для тестирования.

2. Анализ требований: изучение требований к модулю, определение функциональности, которую необходимо протестировать.

3. Разработка тестовых случаев: создание набора тестовых случаев, которые покрывают функциональность модуля и проверяют его корректность.

4. Подготовка тестовых данных: подготовка данных, необходимых для проведения тестирования, например, входных параметров, ожидаемых результатов и т.д.

5. Выполнение тестов: запуск тестовых случаев и анализ результатов тестирования.

6. Отслеживание дефектов: регистрация и отслеживание дефектов, найденных в процессе тестирования.

7. Анализ результатов тестирования: анализ результатов тестирования и принятие решения о готовности модуля к релизу.

8. Отчетность: подготовка отчета о результатах тестирования и передача его разработчикам для исправления дефектов.

В процессе тестирования можно использовать различные методы и инструменты, такие как ручное тестирование, автоматизированное тестирование, функциональное тестирование, нагрузочное тестирование и др. Важно также учитывать особенности SPA приложения, такие как асинхронность, динамическое обновление страницы и т.д. для более эффективного и полного тестирования.

**Что должен включать репорт найденной ошибки, какую структуру должен иметь?**

1. Название ошибки: краткое описание ошибки, которое позволит быстро идентифицировать проблему.

2. Описание ошибки: подробное описание ошибки, включая шаги, которые привели к ее возникновению.

3. Поведение приложения: описание того, как приложение ведет себя при возникновении ошибки.

4. Ожидаемое поведение: описание того, как должно работать приложение в данной ситуации.

5. Скриншоты: скриншоты экрана, которые показывают ошибку и ее последствия.

6. Приоритет ошибки: оценка важности ошибки и ее приоритет для исправления.

7. Среда тестирования: информация о конфигурации среды тестирования, такая как операционная система, браузер и версия приложения.

8. Шаги для воспроизведения: подробное описание шагов, которые нужно выполнить для воспроизведения ошибки.

**Как будет выглядеть баг-репорт, если, к примеру, не работает электрический чайник?**

1. Название ошибки: Не работает электрический чайник.

2. Описание ошибки: При попытке включить электрический чайник, он не включается и не кипятит воду.

3. Поведение приложения: При нажатии на кнопку включения, ничего не происходит. Индикатор не светится, а вода не кипит.

4. Ожидаемое поведение: При нажатии на кнопку включения, чайник должен начать кипятить воду.

5. Скриншоты: не применяется .

6. Приоритет ошибки: Высокий приоритет, так как это основная функция чайника.

7. Среда тестирования: Электрический чайник модели ABC, подключен к розетке в стандартной домашней среде.

8. Шаги для воспроизведения:

- Заполнить чайник водой.

- Подключить чайник к розетке.

- Нажать на кнопку включения.

- Убедиться, что индикатор не светится и вода не кипит.

Примечание: Если возможно, также следует указать серийный номер и другую информацию о чайнике, которая может быть полезна для быстрого выявления проблемы.

**Составить баг репорт страниц и разделов, включая корневой по иерархии сайта** [**https://site.domain.ru**](https://site.domain.ru)

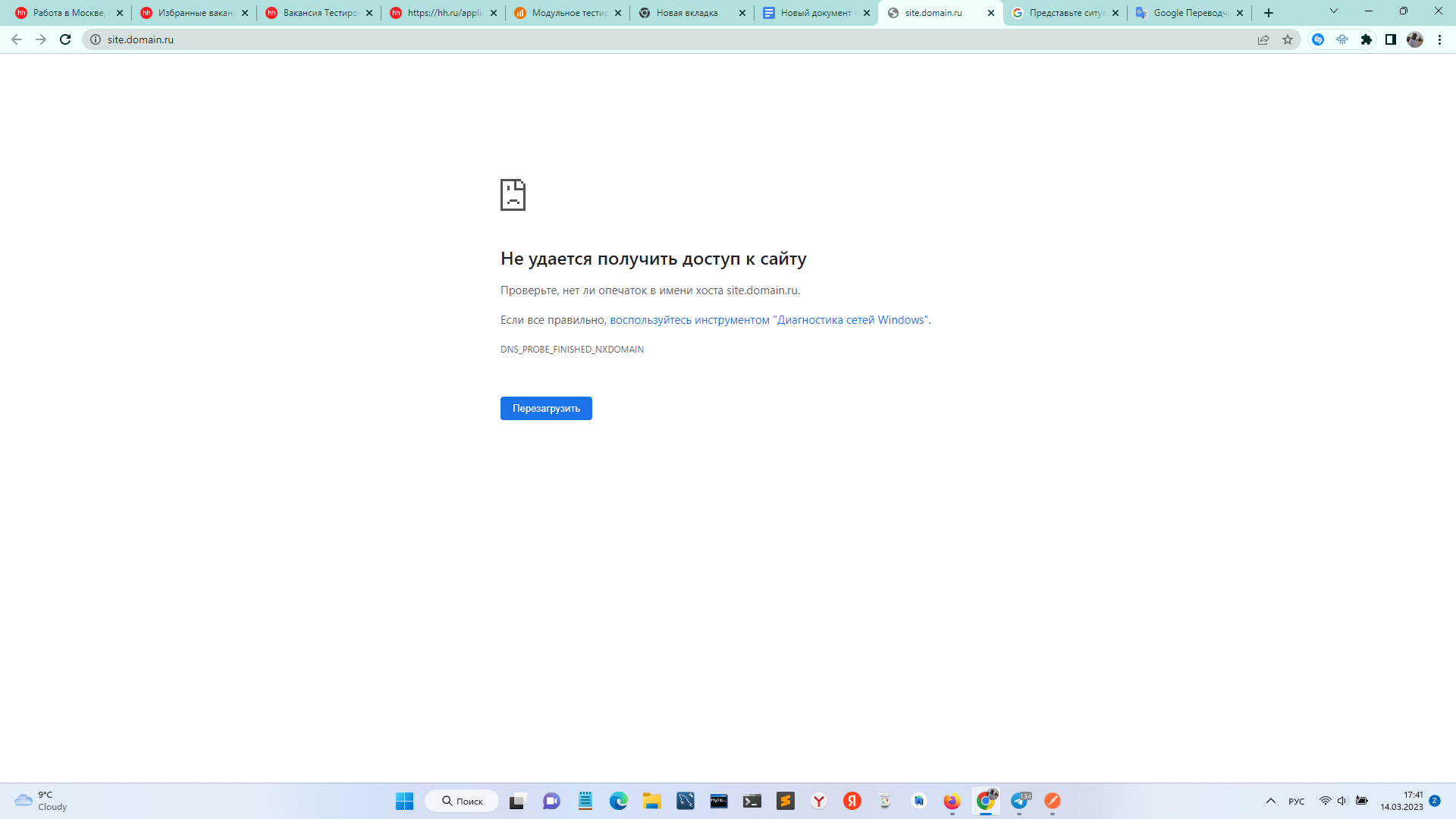
1. Название ошибки: Не удаётся получить доступ к сайту

2. Описание ошибки: При попытке зайти на сайт [**https://site.domain.ru**](https://site.domain.ru) выходит сообщение Не удается получить доступ к сайту

3. Поведение приложения: Сайт не загружается.

4. Ожидаемое поведение: Сайт должен загрузиться.

5. Скриншоты:

****.

6. Приоритет ошибки: Высокий приоритет, так как это основная функция .

7. Среда тестирования:

- Браузер: Google Chrome версии 90.0.4430.212.

- ОС: Windows 10.

- URL: [**https://site.domain.ru**](https://site.domain.ru)

8. Шаги для воспроизведения:

- Открыть браузер.

- В адресной строке ввести URL.

- Нажать Enter.

- Убедиться, что переход на сайт не осуществляется.

**Необходимо протестировать регистрацию нового пользователя в приложении. Расскажите, как вы будете это делать? (описать все этапы)**

1. Подготовка тестовых данных: создание тестового аккаунта с уникальными данными (электронная почта, пароль и т.д.).

2. Запуск приложения и переход на страницу регистрации нового пользователя.

3. Ввод тестовых данных в поля регистрации.

4. Нажатие на кнопку "Зарегистрироваться".

5. Проверка корректности введенных данных: проверка соответствия формата электронной почты, проверка длины пароля и т.д.

6. Проверка появления сообщения об успешной регистрации.

7. Проверка возможности входа в систему с использованием созданных тестовых данных.

8. Проверка отправки письма на указанный при регистрации адрес электронной почты.

9. Проверка возможности изменения данных учетной записи после регистрации (изменение пароля, добавление/удаление информации профиля).

10. Проверка возможности удаления учетной записи.

После выполнения всех этапов я буду анализировать результаты тестирования и документировать найденные ошибки и недостатки для дальнейшего улучшения функционала приложения.

**Представьте ситуацию, что у разрабатываемого приложения еще нет интерфейса, но реализован REST API. Разработчик просит вас создать какую-то сущность в базе и проверить, что она создалась с нужными параметрами. Опишите ход ваших действий в данной ситуации: что и как вы бы проверили, опираясь на имеющееся описание API, с указанием конкретики (название типов запросов и т.д.)**  
В данной ситуации я бы выполнил следующие действия:

1. Ознакомление с документацией API: изучение доступных типов запросов, параметров и форматов ответов.

2. Создание тестовых данных: определение параметров, необходимых для создания сущности в базе данных (например, название, описание, дата создания и т.д.).

3. Выполнение запроса на создание сущности в базе данных с использованием метода POST и передачей тестовых данных в теле запроса.

4. Проверка корректности ответа сервера: проверка кода состояния HTTP (например, 200 OK) и наличия необходимых параметров в ответе (например, ID созданной сущности).

5. Проверка созданной сущности в базе данных: выполнение запроса на получение списка всех сущностей и проверка наличия созданной ранее сущности с необходимыми параметрами.

6. Оформление результатов тестирования и документирование найденных ошибок и недостатков.

При выполнении данных действий я бы использовал инструменты для работы с REST API, такие как Postman или cURL, а также документацию API для определения необходимых параметров запросов и форматов ответов.

**Поле username должно быть обязательным, но оно не обязательно. Приведите пример баг-репорта, созданного для этой ошибки.**

Баг-репорт: Регистрация нового пользователя без указания имени проходит успешно

Описание проблемы: При регистрации нового пользователя через веб-интерфейс не требуется обязательное заполнение поля "Имя пользователя" (username), хотя это поле является обязательным для успешной регистрации. Это может привести к созданию учетных записей без имени, что затруднит дальнейшую работу с ними.

Шаги для воспроизведения:

1. Открыть страницу регистрации нового пользователя на сайте.

2. Заполнить все поля, кроме поля "Имя пользователя".

3. Нажать кнопку "Зарегистрироваться".

4. Проверить, была ли создана новая учетная запись.

Ожидаемый результат: При попытке зарегистрироваться без указания имени пользователя должно возникать сообщение об ошибке или предупреждение, указывающее на необходимость заполнения этого поля.

Фактический результат: Регистрация нового пользователя проходит успешно, без ошибок или предупреждений. Созданная учетная запись не содержит значения в поле "Имя пользователя".

Приложенные материалы: Скриншоты страницы регистрации и созданной учетной записи.

Предложенное решение: Добавить валидацию поля "Имя пользователя" при регистрации нового пользователя через веб-интерфейс и обозначить его как обязательное поле на странице регистрации.

**Можно ли передавать в json бинарные данные? Почему?**

Да, можно передавать двоичные данные в JSON. Однако двоичные данные нельзя напрямую включать в строку JSON, поскольку это недопустимая кодировка символов. Поэтому двоичные данные необходимо преобразовать в текстовый формат, например в кодировку Base64, прежде чем включать их в строку JSON. Когда двоичные данные закодированы в Base64, их можно включить в строку JSON как обычное строковое значение. На принимающей стороне строка в кодировке Base64 может быть декодирована обратно в исходный двоичный формат.

**Написать тестовые наборы данных для поля ввода даты, которое отсеивает пользователей в возрасте до 18 лет.**

1. Дата рождения: 01.01.2005 - Этот пользователь младше 18 лет и должен быть отсеян.

2. Дата рождения: 31.12.2002 - Этому пользователю больше 18 лет и его нельзя отсеивать.

3. Дата рождения: 15.05.2010 - Этому пользователю нет 18 лет и он должен быть отсеян.

4. Дата рождения: 20.09.1999 - Этому пользователю больше 18 лет и его нельзя отсеивать.

5. Дата рождения: 29.02.2004 - Этот пользователь младше 18 лет и должен быть отсеян (поскольку это високосный год, ему не исполнится 18 лет до 2022 года).

6. Дата рождения: 04.07.1998 - Этому пользователю больше 18 лет и его нельзя отсеивать.

7. Дата рождения: 11.11.2003 - Этому пользователю нет 18 лет и он должен быть отсеян.

8. Дата рождения: 22.03.1997 - Этому пользователю больше 18 лет и его нельзя отсеивать.

9. Дата рождения: 16.08.2006 - Этому пользователю нет 18 лет и он должен быть отсеян.

10. Дата рождения: 02.06.2000 - Этому пользователю больше 18 лет и его нельзя отсеивать.

**Как протестировать скачивание файла при недостаточном количестве места на диске.**

1. Проверить доступное дисковое пространство на устройстве, зайдя в проводник или инструменты управления дисками.

2. Попытаться загрузить файл, размер которого превышает доступное место на диске.

3. Обратить внимание на сообщение об ошибке, которое появляется при попытке загрузить файл.

4. Убедиться, что сообщение об ошибке ясно указывает на то, что для завершения загрузки недостаточно места на диске.

5. Если сообщение об ошибке непонятно, попробовать освободить место на устройстве и повторите попытку загрузки.

6. Если сообщение об ошибке не исчезает, попробовать загрузить файл на другое устройство с достаточным объемом дискового пространства, чтобы убедиться, что проблема связана с дисковым пространством.

**Как протестировать загрузку файла при недостаточном количестве места на сервере.**

1. Проверить доступное дисковое пространство на сервере, обратившись к файловой системе сервера или с помощью инструментов управления сервером.

2. Попытаться загрузить файл, размер которого превышает доступное место на диске сервера.

3. Обратить внимание на сообщение об ошибке, которое появляется при попытке загрузить файл.

4. Убедиться, что сообщение об ошибке ясно указывает на то, что на сервере недостаточно места на диске для завершения загрузки.

5. Если сообщение об ошибке непонятно, попробовать освободить место на сервере и повторите попытку загрузки.

6. Если сообщение об ошибке не исчезает, попробовать загрузить файл на другой сервер с достаточным дисковым пространством, чтобы убедиться, что проблема связана с дисковым пространством на исходном сервере.

**Есть веб-страница с полями: e-mail, password и кнопкой submit. Необходимо привести примеры отрицательных тест-кейсов, по которым можно проверить эту страницу.**

1. Ввод недопустимого формата электронной почты (например, отсутствует символ «@» или неверное доменное имя).

2. Ввод слишком короткого или слишком длинного пароля.

3. Ввод пароля, который не соответствует требованиям к паролю веб-сайта (например, не содержит специального символа или заглавной буквы).

4. Оставить поле электронной почты пустым и попытаться отправить форму.

5. Оставить поле пароля пустым и попытаться отправить форму.

6. Ввод действительного адреса электронной почты, но неверный пароль.

7. Ввод несуществующего адреса электронной почты и любого пароля.

8. Попытка отправить форму без заполнения полей.

9. Ввод специальных символов или сценариев в поля электронной почты или пароля для проверки потенциальных уязвимостей системы безопасности.

**Имеется функционал калькулятора, доступный через веб браузер по ссылке. Он имеет только функцию делить, так сказать, MVP-версию. Диапазоны для вписывания в числитель и делитель от 0,1 до 99,9. Вывод значения происходит автоматически, потому что front-end реализован на React JS. Как вы будете тестировать этот функционал? Какие виды тестирования примените? Какие техники тест-дизайна используете?**

Для проверки работоспособности калькулятора я бы использовал следующие виды тестирования:

1. Функциональное тестирование. Это позволит убедиться, что калькулятор правильно выполняет основные арифметические операции (сложение, вычитание, умножение и деление) в заданном диапазоне значений.

2. Юзабилити-тестирование: это гарантирует, что пользовательский интерфейс интуитивно понятен и прост в использовании. Я бы проверил, может ли пользователь легко вводить значения и понимать вывод.

3. Тестирование на совместимость: Это гарантирует, что калькулятор работает правильно в разных веб-браузерах и на разных устройствах.

4. Тестирование производительности: Это гарантирует, что калькулятор будет предоставлять точные результаты в течение разумного периода времени.

Для методов тестирования я бы использовал:

1. Разделение эквивалентности: я бы разделил входные значения на допустимые и недопустимые разделы, чтобы гарантировать, что калькулятор правильно обрабатывает оба случая.

2. Анализ граничных значений. Я бы протестировал калькулятор со значениями в верхней и нижней границах диапазона ввода, чтобы убедиться, что он правильно обрабатывает эти значения.

3. Угадывание ошибок: я намеренно вводил неверные данные, чтобы посмотреть, как калькулятор обрабатывает ошибки.

4. Исследовательское тестирование. Я бы исследовал функциональность калькулятора, чтобы выявить любое неожиданное поведение или дефекты.