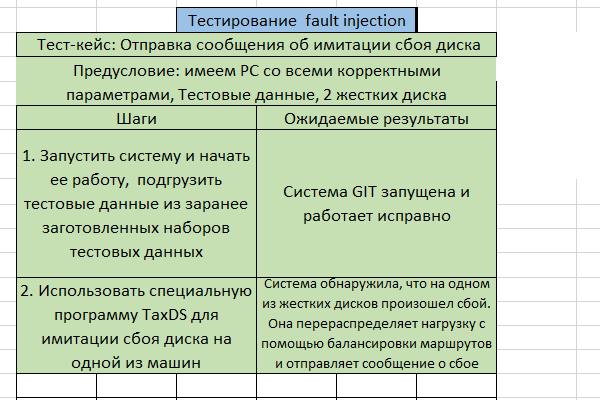
****

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

**Тестирование юзабилити:**

1. Тест кейс: Создание новой ветки в репозитории

- Шаги:

1. Открыть приложение GitSpace.

2. Войти в систему, используя корректные учетные данные

3. Зайти в репозиторий

4. Создать в репозитории новую ветку Example1

- Ожидаемый результат: Пользователь успешно войдет в систему и использует функцию создания новой ветки в репозитории.

2. Тест кейс: Создание репозитория за 5 минут

- Шаги:

1. Войти в систему

2. Создать новый репозиторий с названием test.

3. Посмотреть, как отреагирует приложение на создание уже существующего репозитория

- Ожидаемый результат: Репозиторий будет создан и будет отображаться в списке репозиториев пользователя за 5 минут

3. Тест кейс: Откат одного из файлов в репозитории к прошлому камиту

- Шаги:

1. Войти в систему

2. Зайти в репозиторий

3. Найти функцию отката к предыдущей версии файла.

4. Выполнить откат для одного из файлов в репозитории.

- Ожидаемый результат: Пользователь выполнил откат для выбранного файла.

**Тестирование безопасности:**

1. Тест кейс: Невалидные учетные данные

- Шаги:

1. Открыть веб-приложение GitSpace.

2. Попытаться войти в систему, используя неверные учетные данные.

- Ожидаемый результат: Система должна корректно обработать неудачную попытку входа и сообщить пользователю о неверных данных.

2. Тест кейс: SQL-инъекция в поле поиска

- Шаги:

1. Открыть приложение GitSpace.

2. Войти в систему, используя корректные учетные данные user=admin, password=123123, отображается \*\*\*\*\*

3. Выполнить SQL-инъекцию через поле поиска.

- Ожидаемый результат: Система должна предотвратить SQL-инъекцию и не допустить выполнения вредоносного кода в поле поиска.

3. Тест кейс: Изменение данных в другом репозитории без необходимых на то прав

- Шаги:

1. Открыть приложение GitSpace.

2. Попробовать изменить данные в репозитории другого пользователя без прав выдачи.

- Ожидаемый результат: Система должна блокировать попытку изменения данных другого пользователя и предоставлять доступ только авторизованным пользователям.

**Функциональное тестирование:**

1. Тест кейс: Создание репозитория с именем, которым уже назван другой репозиторий

- Шаги:

1. Войти в систему, используя корректные учетные данные user=admin, password=123123, отображается \*\*\*\*\*.

2. создать новый репозиторий test.

3. добавить файлы Test.py в репозиторий и выполнить коммит изменений.

- Ожидаемый результат: Репозиторий не будет создан, во всплывающем окне будет указано, что репозиторий с таким именем уже существует у пользователя

2. Тест кейс: Создание нового репозитория с уникальным именем

- Шаги:

1. Войти в систему, используя корректные учетные данные user=admin, password=123123, отображается \*\*\*\*\*.

2. создать новый репозиторий test (Данного репозитория нет среди других репозиториев).

3. выполнить коммит изменений.

- Ожидаемый результат: Репозиторий будет успешно создан, файлы добавлены и изменения зафиксированы.

3. Тест кейс: Выполнение merge основной и новой ветки в репозитории без конфликтов

Предусловие: папка Test для коммитов

- Шаги:

1. Войти в систему, используя корректные учетные данные user=admin, password=123123, отображается \*\*\*\*\*.

2. Создать новую ветку в репозитории pa2.

3. Выполнить несколько коммитов из заранее заготовленной папки Test в созданной ветке и смержить ее с основной веткой без конфликтов в файлах.

- Ожидаемый результат: Ветка успешно создана, коммиты выполнены и ветка успешно смержена с основной.