<Rocket.Chat >
Тестовая стратегия

Версия <1.0>

История изменений

Дата	Версия	Описание	Автор
05.08.2023	1.0.	Тестовая стратегия для web- версии Rocket.Chat	Коновалов Александр

Оглавление

1. Введение

Тестирование программы проводит Коновалов Александр Николаевич.

Логин: **roacpatc@student.21-school.ru**
Студент НИТГУ
Выполнил следующие проекты Школы 21:

1) Введение в тестирование (https://sbermarket.ru/ функционал модального окна авторизации);
2) Тестовые артефакты (https://www.saucedemo.com/ тестовые сценарии, тест-кейсы, чек-листы);
3) Тест-дизайн (https://sbermarket.ru/ сценарии использования окна авторизации, попарное тестирование);
4) Работа с дефектами и Bug-tracking tools (тест-кейсы в TestIT, работа в Notion);
5) Front-end тестирование (работа с XPath);
6) Back-end тестирование №1 (работа со Swagger, создание эндпоинтов);
7) Back-end тестирование №2 (работа в Postman, тестирование API);
8) Дополнительные инструменты для тестирования (работа DevTools, PowerShell, Webminal,);
9) Тестирование мобильных приложений (работа с эмулятором Appetize, Android Studio, тестирование приложения tutu);
10) Базы данных (SQL, объекты "Local Storage" и "Session Storage");

Тестируется программа Rocket.Chat.

Rocket.Chat - это открытая платформа для обмена сообщениями и коллаборации в режиме реального времени. Она предоставляет возможность создания собственного сервера обмена сообщениями, который может быть развернут на вашем собственном оборудовании или в облачной инфраструктуре.
Вот некоторые ключевые особенности Rocket.Chat:
Групповые чаты: Rocket.Chat позволяет создавать групповые чаты, в которых пользователи могут обмениваться сообщениями, файлами, ссылками и медиафайлами. Вы можете создавать несколько каналов для разных команд или проектов.
Прямые сообщения: Пользователи могут отправлять прямые сообщения друг другу для приватных обсуждений или конфиденциальной коммуникации.
Видео- и аудиозвонки: Rocket.Chat поддерживает видео - аудиозвонки в режиме реального времени. Вы можете установить соединение с коллегами или друзьями и общаться с ними через видео или аудио.
Интеграция с другими сервисами: Rocket.Chat интегрируется с различными сервисами и

инструментами, такими как GitHub, Jira, Trello, Google Docs иногими другими. Это позволяет вам получать уведомления, обновления и информацию из разных источников прямо в Rocket.Chat.

Мобильные приложения: Rocket.Chat предлагает мобильные приложения для iOS Android, что позволяет вам оставаться на связи и общаться с командой даже в движении.

Пользовательские настройки и расширения: Rocket.Chat предоставляет возможность настройки интерфейса и функциональности под ваши потребности. Вы можете добавлять расширения, создаватьобственные темы оформления и настраивать уведомления.

Rocket.Chat является открытым программным обеспечением, что означает, что его исходный код доступен для всех и может быть изменен идаптирован подаши нужды.то также означает, что вы можете самостоятельно развернуть сервер Rocket.Chat или воспользоваться услугами хостинга, предлагаемыми различными провайдерами.

2. Типы проводимого тестирования

2.1.Функциональное тестирование

- Проверка функциональности: Выполняются тесты, чтобы убедиться, что все функции программного продукта работают должным образом и соответствуют требованиям.
- Тестирование сценариев использования: Используются различные сценарии, чтобы проверить, как программный продукт взаимодействует с пользователями или другими компонентами системы.
- Входные данные и выходные данные: Проверяются правильность обработки входных данных и соответствие ожидаемым результатам.
- Обработка ошибок: Тестируется способность программного продукта обнаруживать и обрабатывать ошибки, а также восстанавливаться после них.

2.2.UI- тестирование

- Проверка внешнего вида: Проверяется, соответствует ли пользовательский интерфейс дизайну и стандартам компании.
- Взаимодействие элементов управления: Тестируется работа кнопок, ссылок, полей ввода и других элементов интерфейса.
- Навигация: Проверяется логика переходов между различными экранами и страницами.
- Отзывчивость интерфейса: Тестируется время отклика и общая производительность интерфейса при выполнении различных действий пользователем.

2.3. Тестирование производительности

- Нагрузочное тестирование: Проверяется поведение программного продукта при работе под нагрузкой, чтобы определить его пределы и способность поддерживать требуемую производительность.
- Стресс-тестирование: Проверяется стабильность системы при ее перегрузке или работе в условиях, близких к предельным.
- Профилирование производительности: Измеряются различные показатели производительности, такие как использование ресурсов (памяти, процессора), время отклика и пропускная способность.

2.4. Тестирование удобства пользования

- Эргономика: Оценивается удобство расположения элементов интерфейса, размеры шрифтов, цветовые схемы и другие аспекты, влияющие на комфорт пользователя.
- Логика и последовательность: Проверяется логичность и последовательность действий пользователя при взаимодействии с программным продуктом.
- Навигация и интуитивность: Тестируется, насколько легко пользователь может найти нужные функции и перемещаться по интерфейсу без дополнительных объяснений или инструкций.
- Понятность сообщений и подсказок: Проверяется понятность текстовых сообщений, ошибок и подсказок, которые предоставляет программный продукт для пользователя.

2.5. Тестирование безопасности

Тестирование безопасности - это процесс исследования и анализа системы, приложения или сети на предмет выявления уязвимостей, которые могут быть использованы злоумышленниками для несанкционированного доступа, повреждения данных или атаки на систему. Вот некоторые основные аспекты тестирования безопасности:

- Подготовка: Этап подготовки включает определение целей тестирования, сбор информации о системе или приложении, разработку тестовых сценариев и выбор методологии тестирования.

- Сканирование: На этом этапе проводится сканирование системы с помощью автоматизированных инструментов для обнаружения открытых портов, слабых точек в конфигурации системы и других потенциальных уязвимостей.

- Анализ уязвимостей: На основе результатов сканирования проводится глубокий

анализ каждой обнаруженной уязвимости. Оцениваются ее потенциальные последствия и вероятность эксплуатации злоумышленниками.

- Эксплуатация уязвимостей: В случае, если требуется дальнейшее исследование уязвимостей, тестировщики могут попытаться эксплуатировать их для получения несанкционированного доступа или выполнения атаки, чтобы показать наличие реальной угрозы.

- Анализ результатов: После завершения тестирования проводится анализ всех собранных данных и результатов. Оцениваются важность и воздействие каждой уязвимости, а также предлагаются рекомендации по устранению обнаруженных проблем безопасности.

- Создание отчета: Финальным этапом является создание подробного отчета о тестировании безопасности, который содержит описание методологии, найденные уязвимости, рекомендации по исправлению и другую полезную информацию для разработчиков и администраторов системы.

- Тестирование безопасности является непрерывным процессом и должно выполняться регулярно, особенно при изменении системы, добавлении новых функций или обновлении программного обеспечения. Цель заключается в обеспечении надежности и защищенности системы от потенциальных угроз безопасности.

3. Части системы, которые будут протестированы

3.1. Учетная запись - Авторизация/- Просмотр и редактирование профиля, статуса/- Присутствие пользователя/- Язык/- Тема рабочего пространства/- Безопасность/- Уведомления

3.2. Главная страница- Создание каналов/- Создание команд/- Создание личных переписок/- Создание обсуждений/- Присоединение к существующим командам (каталог)/- Добавление пользователей/- Настройки внешнего вида

3.3. Работа с чатом - Работа чата/- Работа с участниками и обновлениями/- Создание звонков/- Уведомления/- Экспорт файлов/- Робот строки подсказки

3.4. Работа с сообщениями- Загрузка файлов/- Отправка сообщений (текстовые, голосовые)/- Редактирование сообщений/- Настройки

3.5. Администрирование - Общие настройки/- Звуковое оповещение/- Экспорт и

импорт данных/- Панель администрирования.

4. Окружение для работы. Описание операционных систем, версии приложения и браузеров

- Операционная система: Windows 10
- Браузер: Google Chrome
- Версия Rocket.Chat 1.0

5. Спецификация

5.1 Виды тестовой документации:

- **Тестовая стратегия:** Документ, описывающий общий подход к тестированию, определяющий цели, методы, ресурсы и расписание для проведения тестирования в проекте.

- **Тестовые сценарии:** Документ, содержащий последовательность шагов, которые должны быть выполнены для проверки определенного функционального или нефункционального требования системы. Каждый сценарий описывает конкретный сценарий использования системы и ожидаемые результаты.

- **Тест-кейсы:** Документ, содержащий информацию о конкретных входных данных, предусловиях, ожидаемых действиях и ожидаемых результатов для проведения тестирования определенной функциональности или модуля системы.

- **Тест-планы**: Документ, определяющий общий план проведения тестирования, включая цели, описание тестируемых компонентов, аппаратные и программные требования, расписание тестирования, ответственных лиц и риски, связанные с тестированием.

- **Тестовые отчеты:** Документы, содержащие результаты выполненных тестов, включая информацию о пройденных тест-кейсах, выявленных проблемах (багах), статусе тестирования и рекомендациях для дальнейших действий.

- **Баг репорты:** Документы, содержащие информацию о выявленных проблемах (багах) в системе, включая описание проблемы, шаги для воспроизведения, ожидаемое поведение и актуальное поведение системы, а также другую сопутствующую информацию, необходимую разработчикам для исправления проблемы.

5.2 Техники тест-дизайна:

- **Классы эквивалентности:**

  Классы эквивалентности - это группы входных данных или условий тестирования, которые можно рассматривать одинаковыми для целей тестирования. Каждый класс эквивалентности имеет общие свойства и ожидаемое поведение программы при использовании данных из этого класса. Тестирование на основе классов эквивалентности позволяет сократить количество тестовых случаев, которые необходимо выполнить, чтобы проверить функциональность программы.

- **Попарное тестирование:**

  Попарное тестирование (или тестирование всех пар) — это методика тестирования программного обеспечения, при которой проверяются все возможные комбинации входных данных (параметров) путем сочетания их попарно. Целью попарного тестирования является выявление дефектов, которые проявляются только при определенных комбинациях входных данных, а также эффективное использование ресурсов и времени при проведении тестирования.

- **Тестирование времени реакции:**

  Тестирование времени реакции — это процесс проверки производительности программного обеспечения или системы, связанной с его способностью отвечать на запросы или выполнять операции в заданное время. Целью тестирования времени реакции является определение, соответствует ли производительность программы или системы требованиям по временным параметрам, таким как быстродействие, задержки или время отклика. В ходе тестирования проводятся эксперименты и измерения для оценки времени реакции при различных нагрузках или сценариях использования программного обеспечения или системы.

6. Время проведения тестирования. Когда тестирование можно будет считать завершённым?

- Время начала тестирования - 02.08.2023
- Функциональное тестирование - 02.08.2023
- UI- тестирование - 03.08.2023.
- Тестирование производительности -03.08.2023
- Тестирование удобства пользования - 04.08.2023
- Тестирование безопасности - 05.08.2023

- Фиксирование результатов и составление отчетности - 05.08.2023

Тестирование можно считать завершенным, когда выполнены все заданные тест-кейсы и проверки всех требований.