2 слайд

В современном мире развитие веб-технологий и увеличение количества цифровых продуктов происходят с невероятной скоростью. Однако с ростом сложности веб-приложений повышаются требования к их качеству и стабильности.

Одной из ключевых проблем является необходимость оперативного и качественного тестирования приложений, чтобы обеспечить их надежность, избежать багов и увеличить удовлетворенность конечных пользователей. Традиционный ручной подход требует значительных усилий, времени и ресурсов, что усложняет процесс разработки.

Автоматизированное тестирование выступает эффективным решением, которое позволяет ускорить процесс тестирования, сократить вероятность человеческих ошибок и повысить производительность разработческой команды. Интеграция системы для автоматизированного тестирования в жизненный цикл веб-приложения становится неотъемлемым элементом современного IT.

Таким образом, тема разработки и интеграции системы для автоматизированного тестирования веб-приложения является крайне актуальной, так как она способствует оптимизации процессов обеспечения качества и улучшает процессы разработки веб-продуктов.

3 слайд

Цель работы разработать и интегрировать систему автоматического тестирования для веб-приложения.

В результате выполнения дипломной работы будет разработана и интегрирована система автоматизированного функционального тестирования веб-приложения. Система будет автоматически запускаться после внесения изменений в код (коммита в ветку репозитория), проводить комплексное тестирование всех ключевых аспектов работы приложения. Кроме того, система будет автоматически формировать детализированные отчеты о результатах тестирования, что позволит разработчикам своевременно выявлять и устранять ошибки, а также поддерживать стабильность и высокое качество приложения на всех этапах его разработки.

4 слайд

Для реализации данного проекта были выбраны следующие инструменты:

Язык программирования Python — как основной язык программирования

Фреймворк Selenium — для автоматизации взаимодействия с браузером и проведения верхнеуровневого тестирования..

Библиотека Python-docx — для удобного формирования отчётов в формате DOCX с сохранением структуры и стилей документа.

Фреймворк Boost.Test — для написания модульных тестов, обеспечивающих надёжность и стабильность кодовой базы.

Система Gitea Actions — для автоматизации запуска тестов, что позволило организовать непрерывную интеграцию и оперативно выявлять ошибки.

5 слайд

Первым этапом работы стало проектирование тестовых сценариев. Были детально изучены функционал приложения, его архитектуру и требования, чтобы определить ключевые сценарии использования. На основе этого были разработаны тесты, каждый из которых содержал:

Тестируемую функцию,

Шаги воспроизведения,

Ожидаемый результат,

Критерии успешности

6 слайд

Для автоматизации тестирования были использованы следующие инструменты

pytest – как основной фреймворк для структурирования тестов;

Selenium WebDriver – для взаимодействия с интерфейсом приложения.

Процесс автоматизации включал в себя:

Подготовку окружения – настройку браузерных драйверов и структуры проекта.

Разработку тестов – преобразование ручных сценариев в код с модульной организацией.

Интеграцию Selenium – автоматизацию взаимодействия с UI и проверку результатов.

Это позволило :

сократить время регрессионных проверок, повысить точность за счёт исключения человеческого фактора, подготовить проект к интеграции в CI/CD.

7 слайд

Следующим важным этапом стала разработка модульных тестов для проверки работы отдельных компонентов системы. Мы использовали:

* Boost.Test – мощный фреймворк для модульного тестирования на C++
* FakeIt – для создания mock-объектов и заглушек
* CMake – для удобной интеграции тестов в процесс сборки

Работа велась в несколько этапов:

1. Проектирование тестовой инфраструктуры
2. Настройка окружения
3. Создание реалистичных заглушек
4. Реализация комплексных тестовых случаев

8 слайд

В итоге была разработана автоматизированая система запуска тестов и формирования отчётов на Python, которая:

Запускает модульные тесты (Boost.Test) и анализирует их результаты

Парсит данные из XML-отчётов по модульным тестам

Формирует структурированный отчёт в формате DOCX

Запускает системные тесты и дополняет отчёт новыми данными

Пример готового отчета вы можете увидеть на экране.

9 слайд

В ближайшей перспективе планируется:

Интеграцию с CI/CD – автоматический запуск тестов при каждом коммите в Git, что позволит выявлять ошибки ещё до попадания кода в основную ветку.

Расширение тестового покрытия – добавление новых модульных и системных тестов для повышения надёжности системы.

Улучшение отчётности – улучшить и доработать структуру отчёта.