Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

**Финансовый университет при Правительстве Российской Федерации**

**(Финансовый университет)**

**Департамент анализа данных и машинного обучения**

**Факультета информационных технологий и анализа больших данных**

**Лабораторная работа №6**

**по дисциплине «Управление качеством программных систем»**

Направление подготовки - 09.03.09 «Прикладная информатика»

на тему: Автоматизация тестирования.

**Подготовил**

Студент группы ЗБ-ПИ20-2

Миловидов Владислав Игоревич

(Ф.И.О.)

**Проверил**:

ассистент ДАДиМО ФИТиАБД,

Клочков Евгений Юрьевич

(Ф.И.О.)

Москва 2024

# **Цель работы:**

Целью данной лабораторной работы является, основываясь на ранее разработанной тестовой документации, разработать автотесты для проверки функциональности программной системы для классификации текста в соответствии с требованиями.

# **Описание реализованных автотестов - инструменты, подходы:**

**Инструменты:**

* Тест валидации входных данных (Pytest): тест проверяет

способность системы обрабатывать корректные и некорректные текстовые данные.

* Тест обработки больших объемов данных (JMeter): проверяет

способность системы обрабатывать и классифицировать большие текстовые данные без задержек или сбоев.

* Тест классификации данных (Selenium WebDriver): проверяет

способность системы правильно классифицировать текстовые данные. Проводятся тесты соответствия между выходными данными и данными категории.

* Тест нагрузки/производительности (LoadRunner): оценивает

скорость обработки системы на различном объеме данных и при различных нагрузках, проверяя стабильность и производительность системы.

* Тест интерфейса пользователя (Selenium WebDriver): тесты

направлены на проверку правильности отображения интерфейса, его отклика на взаимодействие с пользователем и корректности отображения результатов классификации.

* Тест корректности взаимодействия с базой данных (SQL):

проверяет, что входящие и исходящие данные корректно сохраняются и извлекаются из базы данных.

* Тест совместимости (Browser Stack): проверяет совместимость

системы с различными браузерами, операционными системами и устройствами.

* Тест безопасности (OWASP ZAP): проверяет безопасность

системы, ее способность противостоять различным видам атак и сохранять конфиденциальность данных пользователя

**Подходы:**

* TDD (Test-Driven Development)/разработка через тестирование:

подход, когда сначала пишутся тесты, а затем код программы. Это позволяет убедиться, что написанный код действительно отвечает его цели. Инструмент Pytest, описанный ранее, отлично подходит для этой цели.

* BDD (Behavior-Driven Development)/разработка через поведение:

ориентируется на поведение программы из точки зрения пользователя. В данном проекте это особенно актуально в части классификации текстовых данных, где можно проверять, насколько хорошо система обрабатывает вводимый текст пользователя и правильно его категоризирует.

* Databased testing (тестирование на основе данных): очень важно

тестировать систему с различными наборами данных для обеспечения корректной работы системы в различных ситуациях.

* Стратегия голубого/зелёного развертывания: это подход

предполагает две отдельные среды (голубую и зелёную), постоянно взаимодействующие друг с другом. Всё новое внедряется и тестируется в зелёной среде, а затем, если тесты проходят успешно, изменения переносятся в голубую среду.

* Непрерывное тестирование: Этот подход предполагает

автоматизированное выполнение тестов при каждом изменении кода, чтобы быстро идентифицировать и исправить проблемы.

* Mocking и Stubbing: это подходы к созданию условных объектов,

которые имитируют поведение реальных объектов в контролируемом окружении. Они особенно полезны при тестировании взаимодействий между различными аспектами системы.

* Негативное тестирование: это тип тестирования, который

проверяет, как система справляется с неправильными, непредвиденными или неполными входными данными. Важно не только убедиться, что система функционирует правильно при ожидаемом поведении, но и выявить, как она будет реагировать на ошибочные данные или действия.

# **Код автотестов:**

В качестве примера мы будем использовать Python и его библиотеку для тестирования pytest.

1. **TDD (Test-Driven Development)**

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

1. **BDD (Behavior-Driven Development)**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Mocking и Stubbing**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Негативное тестирование**

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

# **Отчет о тестировании:**

| **№** | **Шаги тест-кейса** | **Ожидаемый**  **результат** | **Фактический**  **результат** | **Статус** |
| --- | --- | --- | --- | --- |
| 1 | Проверить, что функция категоризации правильно классифицирует текстовые данные | Функция возвращает верный класс текстовых данных | Программа возвращает верный класс текстовых данных | Успешно |
| 2 | Проверить, что система корректно выполняет категоризацию недообученной модели | Функция классификации возвращает ошибку из-за недостаточности данных для обучения | Функция классификации действительно возвращает ошибку из-за недостаточности данных для обучения | Успешно |
| 3 | Проверить, что система успешно обрабатывает большой объем текстовых данных | Функция категоризации успешно обрабатывает и классифицирует большой объем данных | Функция категоризации успешно обрабатывала и классифицировала большой объем данных | Успешно |
| 4 | Проверить, что система регулярно обновляет и улучшает обученную модель по мере поступления новых данных | Функция обновления модели успешно внедряет новые данные для обучения | Функция обновления модели успешно внедрила новые данные для обучения | Успешно |
| 5 | Проверить, получает ли система прикладное обучение при появлении новых категорий текстовых данных | Функция обучения модели включает новые категории данных в процесс обучения | Функция обучения модели успешно включила новые категории данных в процесс обучения | Успешно |

# **Выводы по работе:**

Цель данной лабораторной работы - разработка автотестов для проверки функциональности программной системы для классификации текстовых данных и дальнейшее усовершенствование существующей тестовой документации. В ходе работы были использованы различные инструменты для проверки разнообразных аспектов системы, включая валидацию входных данных, обработку больших объемов текста, классификацию данных, безопасность, взаимодействие с базой данных и совместимость с различными браузерами.

Благодаря подходам, таким как разработка через тестирование (TDD), разработка через поведение (BDD), тестирование на основе данных, голубое/зеленое развертывание, непрерывное тестирование, использование заглушек и фальшивок, а также негативное тестирование, мы смогли провести полное и осмысленное тестирование.

Все тесты были успешно пройдены, свидетельствуя о правильной функциональности системы и ее способности успешно обрабатывать и классифицировать текстовые данные различных объемов и категорий.

В итоге эта лабораторная работа показала, что наша система успешно провела все тесты без каких-либо задержек или сбоев, демонстрируя стабильность и эффективность нашей системы классификации текстовых данных. Это, в свою очередь, говорит об успешной разработке автотестов и эффективности использованных подходов и тестовых стратегий в обеспечении качества многомерной функциональности нашей системы.

# **Список используемых источников:**

1. Учебное пособие «Основы управления качеством программных средств»
2. С. А. Фиросва, А. В. Пакшин «Организация процесса автоматизированного тестирования».