AI-агент для проверки User Stories по INVEST

1. Функциональные требования

1.1 Критерии приемки

Функциональность: Анализ User Story в Telegram-боте

Сценарий 1: Успешный анализ User Story

Дано: Пользователь открывает Telegram-бота командой /start

Когда: Пользователь отправляет валидный текст User Story

Тогда: Бот возвращает структурированный отчет с нарушениями критериев INVEST и рекомендациями по исправлению

Сценарий 2: Обнаружение неизмеримых критериев

Дано: Пользователь отправил User Story со словами "быстро", "удобно"

Когда: Бот анализирует текст

Тогда: Бот отмечает нарушение критерия "Estimable" и предлагает заменить на конкретные метрики

Сценарий 3: Проверка ценности для пользователя

Дано: Пользователь отправил User Story с фразой "Как система, я хочу..."

Когда: Бот анализирует текст

Тогда: Бот отмечает нарушение критерия "Valuable" и предлагает переформулировать с точки зрения пользователя

Сценарий 4: Проверка размера Story

Дано: Пользователь отправил User Story длиннее 500 символов или короче 10 символов

Когда: Бот анализирует текст

Тогда: Бот предупреждает о нарушении критерия "Small" и ограничениях длины

Сценарий 5: Обработка невалидного ввода

Дано: Пользователь отправил пустое сообщение или текст не в формате User Story

Когда: Бот проверяет ввод

Тогда: Бот возвращает понятное сообщение об ошибке с примером корректного формата

1.2 User Story

Как системный аналитик, я хочу отправлять текст User Story в Telegram-бота, чтобы получать мгновенный отчет о нарушениях критериев INVEST с рекомендациями по улучшению.

2. Нефункциональные требования

2.1 Производительность

Время отклика бота: < 3 секунд

Время анализа через LLM: < 30 секунд

Время обработки запроса: < 5 секунд (исключая LLM)

Поддержка пользователей: до 10 одновременных сессий

2.2 Надежность

Доступность системы: 99.5% в месяц

Время восстановления: < 15 минут

Успешность запросов к LLM: ≥ 95%

Retry механизм: до 3 попыток при временных ошибках

2.3 Безопасность

Шифрование данных: TLS 1.2+ для всех коммуникаций

Валидация входных данных: защита от XSS и SQL-инъекций

Rate limiting: не более 10 запросов в час на пользователя

Хранение ключей: в environment variables

2.4 Масштабируемость

Горизонтальное масштабирование: поддержка до 5 инстансов

Балансировка нагрузки: автоматическое распределение запросов

Масштабирование БД: read replicas для запросов истории

2.5 Удобство использования

Время обучения: < 3 минут для освоения функций

Количество шагов: 1 шаг для основного сценария

Ясность сообщений: понятные формулировки на русском языке

Обратная связь: индикаторы прогресса операций

2.6 Совместимость

Telegram: поддержка всех версий Telegram Messenger

Мобильные устройства: адаптивный интерфейс бота

API: REST API спецификация

Форматы данных: JSON для API, Markdown для отчетов

3. Архитектура системы

3.1 Компоненты

Telegram Bot → Backend API (Python) → LLM → Database (SQLite)

3.2 База данных (актуальная схема)

users - пользователи Telegram

user_stories - анализируемые User Stories

violations - выявленные нарушения INVEST

reference_examples - эталонные примеры

bot_sessions - сессии и состояния бота

3.3 API Эндпоинты

POST /analyze - анализ User Story

GET /history - получение истории анализов

GET /examples - получение обучающих примеров

4. Процесс анализа

Основной сценарий:

Валидация входных данных

Сохранение в БД со статусом "processing"

Вызов LLM для анализа INVEST критериев

Обогащение результатов через поиск похожих примеров

Сохранение нарушений и рекомендаций

Возврат структурированного отчета

Ответ включает:

Общую оценку ("Хорошо", "Средне", "Плохо")

Балльную оценку (0-100)

Детализацию по каждому критерию INVEST

Конкретные нарушения с фрагментами текста

Рекомендации по исправлению

Похожие эталонные примеры

5. Метрики успеха

Производительность: < 3 сек время отклика для 95% запросов

Надежность: 99.5% доступность, MTTR < 15 мин

Качество: точность анализа INVEST ≥ 90%

Безопасность: 0 критических уязвимостей

Удобство: 90% пользователей осваивают за 3 минуты

6. Мониторинг и логирование

Метрики: Prometheus + Grafana

Логи: структурированное логирование с контекстом

Алерты: автоматические уведомления о проблемах

Аналитика: отслеживание использования функций