Пояснения к диаграмме:

Основной успешный сценарий:

1. Триггер: Пользователь инициирует сценарий, вводя текст и нажимая кнопку.

2. Интерфейс передает действие на бэкенд через API-запрос.

3. Проверка предусловий. Критически важный шаг, который защищает систему от некорректных данных.

3.1. Выполняются все проверки: наличие текста и его длина.

4. Сохранение факта начала анализа в БД. Важно, что запись создается сразу со статусом processing.

5. Получение ID созданной записи для связи всех последующих данных (нарушений).

6. Вызов внешнего сервиса (LLM) — ключевое действие системы. Это основная бизнес-логика.

7. Обработка запроса внутри LLM-провайдера.

8. Получение результата от LLM в структурированном виде (напр., JSON с массивом нарушений).

9. Обновление ответа данными из RAG. Для каждого нарушения ищется подходящий пример-подсказка.

10. Получение примера из БД.

11. Сохранение результата по каждому нарушению в связующую таблицу analysis_violations.

12. Фиксация успешного завершения операции в БД путем обновления статуса на completed.

13. Возврат сформированного отчета на фронтенд.

14. Завершение — показ финального отчета пользователю.

Альтернативный сценарий (3a):

Единый блок обработки любой ошибки валидации. Обрабатывается до любых обращений к БД и внешним сервисам, что предотвращает лишнюю работу и обеспечивает эффективность. Система мгновенно возвращает понятную ошибку пользователю.

Исключительный сценарий (6-8):

Обработка ошибки внешней зависимости (LLM). Важно, что система не просто падает, но еще и:

6-8.1: Фиксирует факт ошибки (таймаут или код ответа 5xx).

6-8.2: Меняет статус проверки на failed в базе данных, что позволяет анализировать сбои.

6-8.3: Корректно возвращает понятную ошибку на уровень API.

6-8.4: Показывает пользователю сообщение об ошибке.