|  |  |
| --- | --- |
| **Заголовок** | Engineer проверяет покрытие FSR тест-кейсами и генерирует недостающие тесты в SafeValBot (Telegram Mini App / Web). |
| **Акторы** | - Engineer (основной актор)  - SafeValBot (Telegram mini app / Web UI)  - LLM-агент (RAG + генерация тестов)  - Бэкенд API (FastAPI)  - СУБД PostgreSQL  - Zephyr/Jira (через адаптер) |
| **Предусловия** | - Проект создан; в PostgreSQL есть справочники \*\*Safety Goals (SG)\*\* и \*\*FSR\*\* (минимум 1 SG и 1 FSR).  - У пользователя есть доступ (RBAC) к проекту.  - Настроена интеграция с Zephyr (опционально) или включён экспорт в файлы.  - Поднят RAG-индекс с артефактами проекта (описания SG/FSR, политики валидации, форматы метрик).  - Доступен интерфейс Web или Telegram mini app. |
| **Постусловия** | - Инженер видит актуальную матрицу `SG → FSR → Test Case` со статусами.  - Для FSR со статусом \*\*Missing\*\* созданы шаблоны \*\*Zephyr Test Case\*\* или подготовлены к экспорту.  - Связи `FSR ↔ Test Case` сохранены в PostgreSQL.  - Итоговый отчёт (PDF/CSV/SQL) доступен по ссылке и/или отправлен в Telegram. |
| **Ограничения** | - Максимум 2 000 FSR в одном расчёте покрытия (для ответа ≤ 2 с).  - Фильтрация по компоненту/ASIL/статусу доступна только для текущего проекта.  - Telegram-бот обрабатывает команды и кнопки; сложные графики — в Web.  - Генерация LLM ограничена длиной промпта и политиками безопасности (RAG обязателен). |
| **Триггер** | **Engineer**:  - Открывает SafeValBot и выбирает «\*\*Проверить покрытие\*\*» в Telegram;  - или в Web нажимает «\*\*Coverage\*\*» на дашборде проекта. |
| **Основной сценарий** | **Web-интерфейс**  1. \*\*Авторизация и проект.\*\* Пользователь заходит в веб-приложение, авторизуется (SSO/логин), выбирает проект → система грузит дашборд.  2. \*\*Старт покрытия.\*\* Нажимает «Coverage» → API запрашивает из PostgreSQL список SG и FSR, строит матрицу.  3. \*\*Список FSR.\*\* Пользователь выбирает один или несколько FSR для проверки (фильтры: Component/ASIL/Status).  4. \*\*Расчёт статусов.\*\* Сервис Coverage извлекает связанные Test Cases из БД и определяет статусы: 🟢 Covered / 🟡 Missing / 🔴 Failed. Отображаются KPI.  5. \*\*Генерация для Missing.\*\* Для FSR со статусом \*\*Missing\*\* система предлагает «Сгенерировать тест-кейсы (с учётом ASIL)».  6. \*\*LLM+RAG генерация.\*\* LLM-агент подтягивает контекст (FSR, SG, ASIL, политики), формирует шаблоны \*\*Zephyr Test Case\*\*: Preconditions, Steps, Expected, Metrics (SG-009-стиль).  7. \*\*Сохранение.\*\* Пользователь подтверждает; бэкенд сохраняет TC и связи `FSR ↔ TC` в PostgreSQL и (если включено) создаёт записи в Zephyr.  8. \*\*Экспорт.\*\* Пользователь выбирает формат экспорта (PDF/CSV/SQL) и параметры (фильтры, снапшот времени) → получает ссылку/файл.  9. \*\*Результат.\*\* Матрица покрытия обновлена; Missing → Covered (для сгенерированных TC). Отчёт доступен и в Web, и в Telegram.  **Telegram**-**бот**  1. \*\*Старт.\*\* Пользователь пишет `/start` → бот показывает меню: «Проверить покрытие», «Импорт SG/FSR», «Экспорт отчёта».  2. \*\*Выбор покрытия.\*\* Нажимает «Проверить покрытие» → бот запрашивает у API список FSR (с SG/ASIL/Component) и отображает карточками/кнопками.  3. \*\*Выбор FSR.\*\* Пользователь отмечает FSR → бот отправляет запрос на расчёт статусов и показывает 🟢/🟡/🔴 в короткой таблице.  4. \*\*Генерация.\*\* Если есть \*\*Missing\*\*, бот предлагает «Сгенерировать тесты (с ASIL-пояснением)» → по подтверждению запускает генерацию.  5. \*\*Сохранение.\*\* После генерации бот сообщает о созданных TC и связях в БД/Zephyr.  6. \*\*Экспорт.\*\* Пользователь выбирает «PDF/CSV/SQL» → получает файл/ссылку в чат. |
| **Альтернативные сценарии** | 1. \*\*Нет FSR в проекте\*\* → предложение импортировать CSV/JSON (шаблон + пример).  2. \*\*Частично заполнены связи\*\* → статус `Partial`; бот предлагает дозаполнить/сгенерировать недостающие TC.  3. \*\*Engineer выбирает только Covered\*\* → система показывает KPI и предлагает экспорт без генерации.  4. \*\*Нет интеграции с Zephyr\*\* → создать TC локально и отдать экспорт для ручного импорта. |
| **Исключительные сценарии** | 1. \*\*Ошибка БД/недоступность\*\* → показать баннер «Сервис временно недоступен», предложить повторить.  2. \*\*LLM недоступен/таймаут\*\* → вернуть шаблонные skeleton-TC на основе RAG без reasoning; пометить «Draft».  3. \*\*Ошибка Zephyr API\*\* → сохранить TC и связи локально, поставить задачу на повторную публикацию; уведомить пользователя.  4. \*\*Неверная команда в боте\*\* → подсказка `/help` с примерами. |
| **Критерии успеха** | - Время расчёта покрытия ≤ \*\*2 сек\*\* для ≤ \*\*2 000 FSR\*\*.  - Доля корректной генерации TC (валидная структура Zephyr) ≥ \*\*95%\*\*.  - Снижение \*\*Missing\*\* после автогенерации ≥ \*\*80%\*\* для выбранного набора.  - Синхронизация Web ↔ Telegram ≤ \*\*1 сек\*\* по снапшоту отчёта. |

**Технические детали**

\*\*Web:\*\*

- Frontend: React (или Next.js), DataGrid, экспорт (pdfmake/canvas + server-side PDF).

- Backend: Python FastAPI; эндпоинты: `/fsr`, `/coverage`, `/generate`, `/export`.

- RAG: pgvector/FAISS; источники — SG/FSR, политики, шаблоны метрик.

- Хранилище: PostgreSQL (основные сущности), S3/MinIO (PDF/CSV/SQL).

\*\*Telegram-бот:\*\*

- Библиотека: `aiogram` (Python) + Telegram WebApp initData (валидация подписи).

- Команды:

`/start` — меню;

`/coverage` — проверка покрытия;

`/export [pdf|csv|sql]` — экспорт;

`/help` — справка.

- Ограничения UI: сложные графики — по ссылке в Web; в боте — компактные таблицы и кнопки.

## Диаграмма взаимодействия (Mermaid)

```mermaid

sequenceDiagram

actor Eng as Engineer

participant TG as SafeValBot (TG/Web)

participant API as Backend API

participant DB as PostgreSQL

participant LLM as LLM+RAG

participant ZEP as Zephyr

Eng->>TG: Старт «Проверить покрытие»

TG->>API: list FSR / SG

API->>DB: SELECT SG, FSR

DB-->>API: rows

API-->>TG: список FSR

Eng->>TG: Выбор FSR → Далее

TG->>API: coverage(FSR\_ids)

API->>DB: JOIN FSR↔TC статус

DB-->>API: статусы (Covered/Missing/Failed)

API-->>TG: KPI + статусы

alt Есть Missing

Eng->>TG: Сгенерировать TC

TG->>API: generate(FSR\_missing)

API->>LLM: контекст (ASIL, SG/FSR, политики)

LLM-->>API: черновики TC

API->>DB: INSERT TC + links

DB-->>API: ok

API-->>TG: TC создано, связи сохранены

end

Eng->>TG: Экспорт PDF/CSV/SQL

TG->>API: export(format, filters)

API->>DB: snapshot coverage

API-->>TG: ссылка/файл отчёта