Мы представляем концепт и решение по кейсу «Автоматизация проверки требований в сертифицируемом контуре».

Концепт:
1. Создание классификатора, содержащего объекты регулирования (далее ОР)(тормоза, дворники и омыватель, сиденья) с отнесением каждого из них к регламенту (стандарту), а так же определению в них системы в соответствии с регламентами. В классификаторе учитываются все альтернативные названия объектов и систем входящих в состав ОР и сопоставление названий из Use Case (далее UC) с определениями из регламентов. Такой классификатор обеспечит:
- на стадии входа ОР в систему выявить не подлежащие сертификации системы, подсистемы и функции;
- обеспечить привязку ОР к определенному спектру требований.

2. Работа с регламентами:
2.1. В систему загружаются регламенты в формате pdf, docx, txt с ручным выбором ОР, к которым/му относится данный регламент. Загрузка осуществляется пользователями, имеющими доступ к данному разделу. Привязка ОР к регламентам должна выполняться из списка, строго внесеного в классификатор. Это позволит снизить ошибки, связаные с человеческим фактором.
2.2. Из регламента извлекаются части «Технические требования» и «Испытания» и производится их разделение на подпункты (возможна автоматическая реализация на python при условии одинаковой структуры регламентов). Каждый пункт должен иметь уникальный код идентификатор.
2.3. К подпунктам добавляются метаданные, содержащие название регламента и наименование ОР.
2.4. На основании подпунктов создаются эмбединги (с помощью энкодеров или уже готовых моделей типа BERT).
2.5.3. Сохранение эмбедингов происходит в векторную БД (FAISS, Chroma, Qdrant) по принципу: один ОР = одна таблица.

3. Работа с UC:
3.1. При загрузке UC на основании пункта «OC Subsection» получаем ОР;
3.2. На основании содержания UC (например, получаем самые частые слова после очистки текста и по косинусному сходству) подбирается система. В идеале в самом UC в описании должно содержаться название системы, как в классификаторе. (требуется доработка UC в части стандартизации наполнения)
3.3. В минимальной версии реализуется:
- подбор по каждому пункту из разделов «Сценарий», «Альтернативный Сценарий» и «Пост Условия» по семантическому сходству из соответствующей ОР таблицы в векторной БД;
- сортировка их по релевантности в соответствии со схожестью, реализованной на SpaCy Python;
- вывод на экран пунктов, соответствующих UC;
- сохранение пунктов в файл.
3.4. В расширенной версии добавить дообучение (RAG) LLM (например, T-lite от Т-банка, которая сопоставима с импортными аналогами). Для LLM ключевым моментом будет написание промта, в него необходимо включить:
- структуру самого UC (можно загрузить в модель файл с описанием этой системы);
- проблематику:
-- сравнение по смыслу, тональности;
--поиск несоответствий в самих регламентах;
--все ли необходимые пункты из регламента прописаны в загруженных UC.

Дополнительные фичи:
Пример полного цикла управления требованиями в части системы очистки лобового стекла.

Определив, что UC отражает функцию очистки лобового стекла, на данный UC будут распространяться требования ГОСТ 33993-2016 пункт 3.1. В случае, если UC описывает омыватель стекла, то на него будут распространятся требования пункта 3.2 того же ГОСТ. Так же вполне возможно определить выполненные требования ГОСТ в описании UC.
Например, п.3.1.1 требует чтобы любое транспортное средство оснащалось очистителем ветрового стекла, если мы имеем UC, описывающий данную функцию, то данный пункт заведомо выполнен. Данная информация может быть представленна как цветовая дифференциация набора требований к UC.
Требования регламента в части испытаний могут использоваться для построения предотчетов по верификации требований. Пример, UC описывает функции подсистемы, на UC навязываются требования, разработчик UC дорабатывает его. После загружает в систему и UC сравнивается с разделом БД в части испытаний соответствующего регламента. Формируется предотчет о верификации UC.

Получилось реализовать:
Решение оформлено в виде веб-приложения (FastAPI) со следующими функциями:
- загрузка регламентов в векторную БД (Langchain, FAISS, модель all-MiniLM-L6-v2);
- загрузка Use Case;
- поиск по семантическому сходству пунктов в регламентах, соответствующих тексту UC;
- вывод результата на экран.