Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка чат-бота GPT для анализа и сравнения документации по объектам»

по дисциплине «Проектный практикум»

Заказчик: Чекалова Э. Р.

Куратор: Кузнецов Даниил Сергеевич
Сотрудник компании «ИнПАД»

Студенты команды MIечный путь
Косторной Д. В.
Соболев Е. В.
Тимофеев И. Д.

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель проекта – разработка чат-бота на основе GPT, способного анализировать и сравнивать техническую документацию по строительным объектам для выявления расхождений между Техническим заданием (ТЗ) и проектной документацией.

Основные задачи:

– Реализация механизма загрузки и обработки документов (PDF);

– Разработка алгоритма извлечения и семантического сравнения текстов с использованием генеративных нейросетей (GPT-4, DeepSeek);

– Создание системы категорированного анализа (архитектурный, конструктивный, инженерный разделы);

– Разработка модуля формирования отчетов в различных форматах (PDF, TXT, XLS);

– Интеграция базы данных для хранения результатов анализа и истории запросов пользователей;

– Обеспечение удобного интерфейса взаимодействия через Telegram-бота;

– Загрузка решения на хостинг для стабильной работы.

В строительной сфере ручное сравнение ТЗ и проектной документации – трудоёмкий и подверженный ошибкам процесс. Специалисты тратят часы на поиск несоответствий, что приводит к:

– Рискам пропуска критических ошибок из-за человеческого фактора;

– Задержкам сроков и увеличению бюджета из-за позднего обнаружения проблем;

– Неэффективности существующих инструментов (например, Diff-анализаторы не учитывают смысловую нагрузку параметров).

Автоматизация процесса с помощью ИИ позволяет:

– Ускорить анализ (сокращение времени с часов до минут);

– Повысить точность за счет использования генеративных нейросетей;

– Снизить затраты на рутинные проверки.

Продукт предназначен для:

– Строительных компаний (контроль соответствия проектов ТЗ);

– Проектировщиков и инженеров (оперативная проверка документации);

– Заказчиков строительства (независимый аудит исполнения контракта).

Примеры использования:

– Сравнение материалов и толщин стен (архитектурный раздел);

– Проверка соответствия инженерного оборудования (вентиляция, отопление);

– Анализ конструктивных решений (несущие конструкции).

По завершении проекта должен быть реализован бот с функционалом:

– Минимальный уровень: Загрузка документов, базовый анализ расхождений;

– Базовый уровень: Сравнение по заданным параметрам (разделам), генерация отчетов;

– Оптимальный уровень: Высокая точность анализа.

Планируемые достижения:

– Сокращение времени проверки документации;

– Минимизация человеческих ошибок за счет алгоритмов ИИ;

– Готовое к использованию решение;

– Подтверждённая работоспособность (куратором, заказчиком и тестированием на реальных документах).

1 Основная часть

1.1 Разбор требований заказчика. Составление бэклога задач

На основании технического задания (ТЗ) от компании «ИнПАД» (приложение А) были выделены следующие ключевые требования к чат-боту:

– Возможность выбора конкретных разделов для анализа (архитектурный, конструктивный, инженерный);

– Сравнение параметров: материалы стен и полов, толщина конструкций, производители инженерного оборудования и так далее;

– Формирование отчётов в форматах PDF, TXT и XLS с указанием страниц несоответствий;

– Обеспечение конфиденциальности обрабатываемых данных;

– Интеграция генеративной модели для семантического анализа текста;

– Удобный интерфейс взаимодействия через Telegram [1].

Для выполнения требований был сформирован бэклог задач (приложение Б), распределённый по трём контрольным точкам: аналитика, разработка, разработка 2.0 – тестирование и доработка.

1.2 Анализ и сопоставление аналогов разрабатываемого продукта

Для определения уникальности и конкурентных преимуществ разрабатываемого чат-бота был проведён анализ существующих решений, предназначенных для сравнения технической документации. Рассматривались как специализированные инструменты, так и общие платформы с возможностью обработки документов.

1.2.1 Обзор аналогов

1) DiffPDF:

Функционал: Позволяет находить текстовые различия между двумя PDF-файлами [2].

Недостатки:

– Сравнение происходит на уровне символов, без учёта семантики;

– Не поддерживает категорированный анализ (например, только по архитектурному разделу);

– Требует ручной постобработки для интерпретации результатов.

2) ChatGPT / Deepseek:

Функционал: Позволяет загружать документы и задавать вопросы по их содержанию [3].

Недостатки:

– Нет специализации на технической документации — некорректно обрабатывает таблицы, чертежи, ГОСТы;

– Не формирует структурированные отчёты с указанием страниц расхождений;

– Ограничения на длину обрабатываемого текста (например, 128K токенов у GPT-4o mini).

3) Специализированные BIM-системы (Revit, Navisworks):

Функционал: Встроенные инструменты для проверки коллизий и соответствия стандартам [4].

Недостатки:

– Работают только с моделями, а не с текстовой документацией;

– Высокий порог входа (требуют обучения);

– Не поддерживают семантическое сравнение ТЗ и проектной документации.

4) ПО для управления требованиями (IBM DOORS, Jama Connect):

Функционал: Трекинг изменений в требованиях, интеграция с документами.

Недостатки:

– Ориентированы на ПО, а не на строительные объекты;

– Не автоматизируют поиск расхождений в инженерных разделах.

1.2.2 Конкурентные преимущества разрабатываемого решения

Сравнительный анализ функциональных возможностей аналогов и разрабатываемого решения представлен в таблице 1.

Таблица 1 – Сравнительный анализ аналогов и разрабатываемого решения

Критерий	Аналоги (DiffPDF, ChatGPT)	Разрабатываемый чат-бот
Семантический анализ	Только текстовые различия	Сравнение по смыслу
Категорированная проверка	Нет	Выбор разделов (архитектурный, конструктивный, инженерный)
Формат отчётов	Нет или простой текст	PDF, XLS, TXT с указанием страниц
Интеграция с мессенджерами	Требует отдельного ПО	Работа в Telegram
Обработка больших файлов	Обрезает текст	Чанкинг + асинхронный анализ

Таким образом разрабатываемый продукт устраняет ключевые недостатки аналогов:

– Автоматизация анализа с пониманием контекста – в отличие от Diff-инструментов, работающих на уровне текстовых совпадений, бот интерпретирует смысл параметров (например, распознает, что «бетон В20» и «бетон класса В20» обозначают один материал);

– Специализация на строительной отрасли – шаблоны для разных разделов строительной документации;

– Доступность – решение не требует сложного внедрения (доступ через Telegram).

1.3 Обзор архитектуры программного продукта

1.3.1 Компонентная архитектура решения

Разработанный чат-бот реализован по модульной архитектуре с четким разделением функциональных слоёв (рисунок 1) и включает в себя:

1) Презентационный слой:

– Telegram-интерфейс на aiogram 3.19.0;

– Система интерактивных клавиатур;

– Finite State Machine для управления диалогами.

2) Бизнес-логика:

– Модуль обработки PDF-документов;

– Аналитический движок на базе GPT-4o mini;

– Подсистема векторного поиска;

– Модуль поиска ответа на вопросы на базе GPT-4o mini;

– Генератор отчетов.

3) Слой данных:

– Асинхронная SQLite база данных;

– Временное файловое хранилище;

– Векторные индексы FAISS.

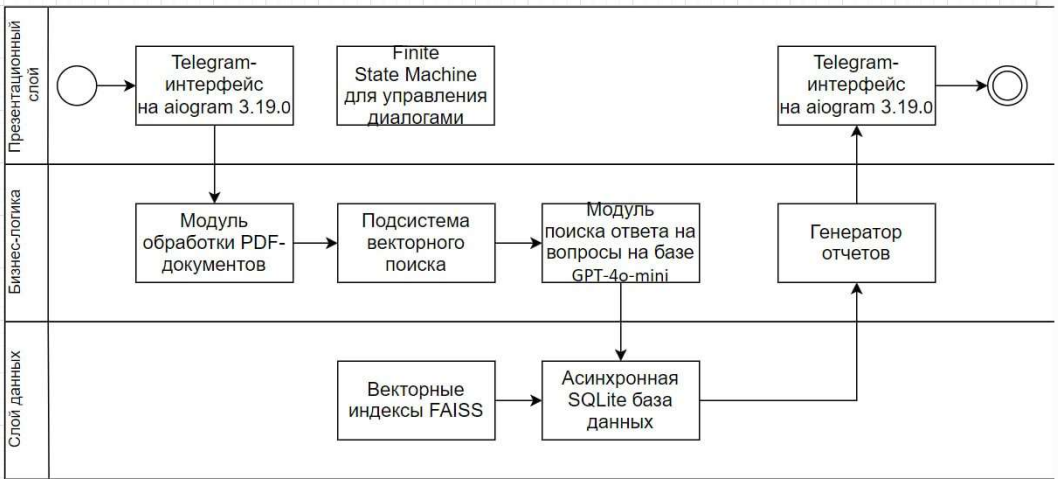


Рисунок 1 – Схема архитектуры решения

1.3.2 Ключевые технологические решения

Обработка документов реализована с применением:

– PyPDF2 для извлечения текста [5];

– Кастомного чанкера с сохранением структуры страниц;

– Механизма маркировки страниц.

Семантический анализ включает:

– Двухэтапную обработку (векторный поиск + LLM);

– OpenAI Embeddings для создания векторных представлений;

– FAISS для эффективного поиска похожих фрагментов [6].

Работа с GPT организована через:

– Асинхронный OpenAI API клиент;

– Динамические тексты запросов для разных типов анализа;

– Механизм повторных попыток с экспоненциальным ожиданием.

1.3.3 Потоки данных

Основные процессы обработки документов представлены на упрощённой BPMN-диаграмме процесса (приложение В). На диаграмме выделены следующие компоненты и их взаимодействие:

Исполнители процесса:

1) Пользователь – инициатор процесса анализа

2) Чат-бот – основной интерфейс взаимодействия

3) Сервер – база данных, хранилище документов и результатов

4) ИИ-модуль – ядро аналитической системы

Ключевые этапы обработки:

1) Инициация процесса:

– Запуск бота и выбор команды анализа;

– Выбор конкретного раздела для сравнения;

– Загрузка документов во временное хранилище.

2) Аналитическая обработка:

– Парсинг PDF с сохранением структуры страниц;

– Нормализация текстовых данных;

– Разметка разделов документа;

– Разбиение текста на смысловые чанки;

– Построение векторных представлений (FAISS);

– Сравнение через GPT-4o mini с учетом контекста.

3) Формирование и доставка отчета:

– Группировка результатов по типам расхождений;

– Привязка к страницам исходных документов;

– Генерация отчетных форм в необходимом формате (txt, xlsx, pdf).

4) Финализация процесса:

– Отправка отчёта пользователю;

– Удаление временных файлов.

1.3.4 Обоснование выбора технологий

1) Aiogram 3.19.0 [7]:

– Нативная поддержка асинхронности;

– Гибкая система middleware;

– Встроенные механизмы FSM;

– Поддержка создания пользовательской сессии для загрузки больших файлов.

2) Async SQLite [8]:

– Достаточная производительность для нагрузки;

– Простота развертывания;

– Совместимость с Python asyncio.

3) FAISS:

– Оптимизированные векторные операции;

– Эффективный поиск по сходству;

– Локальное исполнение без внешних зависимостей.

4) GPT-4o mini [9]:

– Большой контекст (128k токенов);

– Высокая точность анализа;

– Предсказуемая стоимость запросов.

1.3.5 Обеспечение надежности

Система включает:

– Трехкратные повторные попытки для API-запросов;

– Валидацию входных данных;

– Автоматическую очистку временных файлов;

– Логирование ключевых операций.

Таким образом архитектура системы обеспечивает выполнение ключевых требований:

– Обработку больших документов через чанкинг;

– Точный анализ через комбинацию векторного и LLM-поиска;

– Гибкость через настраиваемые сценарии анализа;

– Масштабируемость за счёт асинхронной реализации.

1.4 Методология разработки и процесс тестирования

1.4.1 Методология разработки

Проект разрабатывался по классической каскадной (waterfall) методологии, что соответствовало требованиям заказчика. Данный подход был выбран по следующим причинам:

– Чёткое фиксирование требований на начальном этапе (техническое задание не изменялось в процессе разработки);

– Необходимость последовательного выполнения этапов разработки;

– Ограниченный срок реализации (1 семестр).

1.4.2 Тестирование

В проекте были задействованы следующие виды тестирования:

– Модульное тестирование (unit testing) – проверка отдельных функций, например, извлечение текста из pdf, разбиение на чанки, загрузка информации в базу данных (приложение Г);

– Интеграционное тестирование: Проверка взаимодействия модулей (например, загрузка файлов, сохранение в базу данных и анализ того, что было сохранено);

– Приёмочное тестирование: проводилось заказчиком несколько раз за семестр на большом количестве реальных документов с целью найти примеры некачественного анализа или недоработок.

1.4.3 Результаты тестирования

На основном этапе тестирования были выявлены и устранены следующие типы проблем:

1) Юзабилити и взаимодействие с пользователем:

– Обнаружены трудности в навигации (отсутствие кнопки возврата);

– Неясные пользовательские инструкции при загрузке документов;

– Плохая читаемость отчётов;

2) Стабильность и воспроизводимость результатов:

– Различия в формируемых отчётах для идентичных документов, сгенерированных с небольшим временным промежутком;

– Ошибки в обработке страниц документа, иногда ведущие к ссылкам на несуществующие страницы.

3) Корректность обработки данных:

– Некорректная интерпретация аналогов материалов (ложные срабатывания при проверке соответствий);

– Неточности в определении соответствия разделов (например, указания по вентиляции или изоляции);

– Ошибки в разграничении анализа разделов (например, попадание данных из КР в АР).

4) Ошибки экспорта и формирования отчётов:

– Неожиданные исключения, прерывающие генерацию отчётов;

– Не выявленные ошибки (аэраторы, слои гидроизоляции) из-за недостаточной точности анализа.

5) Формулировки и представление ошибок:

– Сообщения об ошибках были недостаточно ясны для пользователя и требовали уточнений.

6) Долгое время анализа документов:

– Изначально анализ 100 страниц занимал до 25 минут. Благодаря оптимизации (параллелизация запросов, кэширование векторных представлений и так далее) время удалось сократить до 5 – 7 минут.

1.4.4 Процесс разработки

Основные проблемы, которые возникли в процессе разработки:

1) Выбор ИИ-модели для анализа:

Изначально предполагалось использовать бесплатную версию DeepSeek [10], но она оказалась менее эффективной, чем GPT-4o mini, так как обладает меньшим контекстом [11]. Переход с одной модели на другую потребовал

полного переписывания запросов, но увеличил эффективность поиска несоответствий примерно на 20%.

2) Недостаточное тестирование:

Из-за специфики строительной документации (конфиденциальность) и недостатка строительной компетенции команды, тестирование самого анализа в основном проводилось заказчиком и куратором на основе комментариев. Это привело к тому, что некоторые ошибки были выявлены достаточно поздно, что вызвало необходимость дополнительных доработок.

3) Смена архитектуры проекта:

В связи с тем, что изначально для бота использовалась библиотека Aiogram 2.6.0, а потом потребовалось перевести её на 3.19.0 (из-за необходимости использования пользовательской сессии для загрузки файлов) в код были внесены значительные изменения в обработчики, классы состояний и так далее.

Переход на новую версию библиотеки привел к увеличению производительность бота примерно на 30% и позволил загружать более большие файлы для анализа, но также потребовал дополнительных часов разработки.

4) Сложности с ограничениями API:

В процессе разработки были обнаружены ограничения по количеству запросов или контексту при работе с различными API (в частности, размер контекста для GPT-4o), что потребовало перепроектирования алгоритмов обработки данных.

1.5 Планирование деятельности и распределение задач

1.5.1 Команда

Проектная команда состояла из трёх участников с четким распределением ролей:

– Косторной Д. В. – тимлид и разработчик, ответственный за всю логику и архитектуру бота, интеграцию в него остальных алгоритмов;

– Соболев Е. В. – разработчик баз данных и тестировщик, ответственный за разработку и внедрение базы данных, тестирование;

– Тимофеев И. Д. – ML-разработчик, ответственный за интеграцию нейросетей и разработку алгоритмов анализа и сравнения.

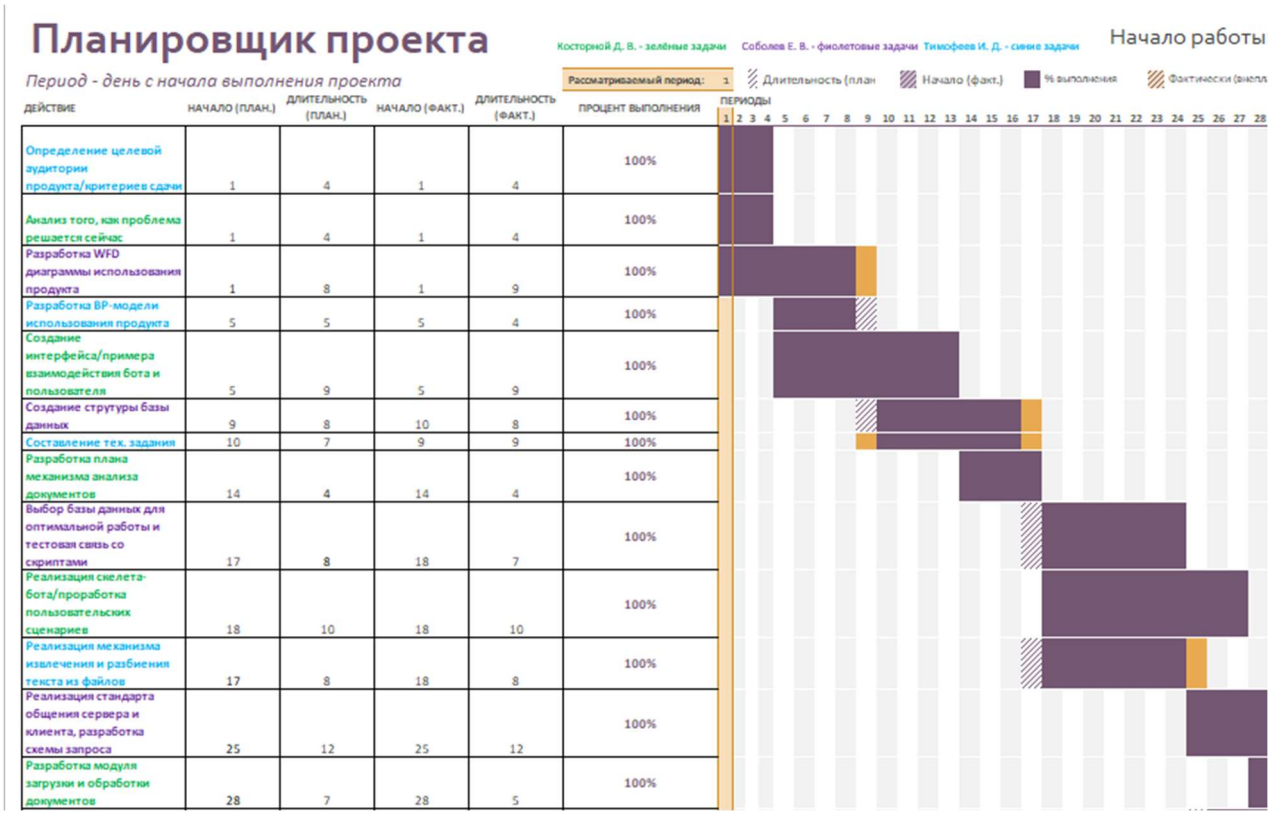
Для управления проектом применялись:

– Диаграмма Ганта (рисунок 2) – для визуализации сроков выполнения этапов;

– Бэклог всех задач (приложение Б);

– Таблица трудозатрат (приложение Д);

– Командное хранилище в Google Drive и репозиторий в Github для контроля версий и синхронизации работы.



ДЕЙСТВИЕ	НАЧАЛО (ПЛАН.)	ДЛИТЕЛЬНОСТЬ (ПЛАН.)	НАЧАЛО (ФАКТ.)	ДЛИТЕЛЬНОСТЬ (ФАКТ.)	ПРОЦЕНТ ВЫПОЛНЕНИЯ
Определение целевой аудитории продукта/критериев сдачи	1	4	1	4	100%
Анализ того, как проблема решается сейчас	1	4	1	4	100%
Разработка WFD диаграммы использования продукта	1	8	1	9	100%
Разработка BP-модели использования продукта	5	5	5	4	100%
Создание интерфейса/примера взаимодействия бота и пользователя	5	9	5	9	100%
Создание струтуры базы данных	9	8	10	8	100%
Составление тех. задания	10	7	9	9	100%
Разработка плана механизма анализа документов	14	4	14	4	100%
Выбор базы данных для оптимальной работы и тестовая связь со скриптами	17	8	18	7	100%
Реализация скелета-бота/проработка пользовательских сценариев	18	10	18	10	100%
Реализация механизма извлечения и разбиения текста из файлов	17	8	18	8	100%
Реализация стандарта общения сервера и клиента, разработка схемы запроса	25	12	25	12	100%
Разработка модуля загрузки и обработки документов	28	7	28	5	100%

Рисунок 2 – Фрагмент диаграммы Ганта

Подробное разделение задач между участниками представлено в бэклоге задач проекта (приложение Б).

Благодаря таблице трудозатрат (приложение Д) можно оценить трудозатраты в часах для каждого участника команды:

– Косторной Д. В. – 58 часов;

– Соболев Е. В. – 54 часа;

– Тимофеев И. Д. – 54 часа.

Потраченные каждым участником часы полностью соответствуют часам дисциплины в учебном плане.

1.5.2 Отчёт о работе участника. Косторной Дмитрий Вадимович

Аналитический этап:

1) Создание плана работы над проектом:

Было выполнено разделение задач между участниками команды и создан план работ в Excel планировщике проектов на основе диаграммы Ганта, считая периоды за дни с начала работы над проектом. План работ представлен на рисунке 2.

2) Анализ текущих методов решения проблемы:

Были изучены ручное сравнение документов и Diff-инструменты. Выявлены их ограничения (символьное сравнение, отсутствие семантики). Результаты легли в основу требований к ИИ-модулю и в таблицу конкурентных преимуществ решения (таблица 1).

3) Создание интерфейса взаимодействия бота и пользователя:

Были разработаны прототипы диалоговых сценариев (рисунок 3). Реализована система кнопок для выбора разделов (архитектурный, конструктивный и так далее).

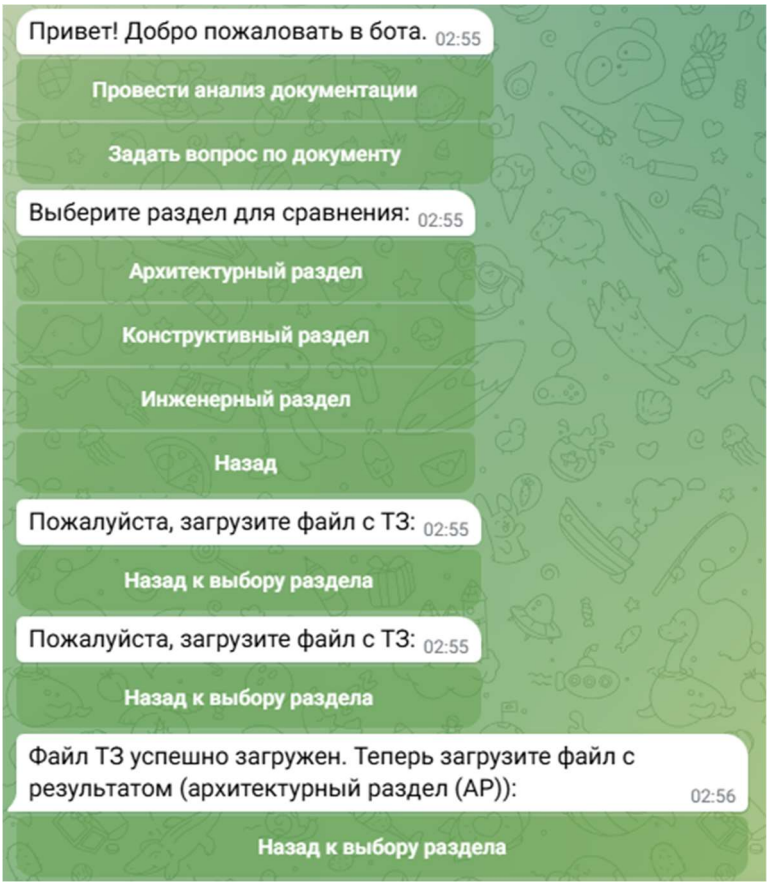


Рисунок 3 – Интерфейс основных клавиатур бота

4) Разработка плана механизма анализа документов:

Был спроектирован алгоритм обработки PDF, состоящий из этапов:

– Извлечение текста из файлов;

– Разбитие текста на меньшие части с контролем токенов;

– Создание векторных баз для каждого файла;

– Семантический поиск схожих частей;

– Сравнение наиболее схожих частей генеративной моделью;

– Структурирование всех сравнений в общий отчёт.

Данный алгоритм в дальнейшем лёг в основу bpmn-диаграммы (приложение В).

Разработка:

1) Реализация скелета бота:

Был создан базовый каркас с помощью библиотеки aiogram 3.19.0 с асинхронной обработкой запросов. Интегрированы FSM для управления

диалогами. Написаны все основные обработчики событий бота (приложение Е);

2) Проработка пользовательских сценариев:

Были реализованы сценарии взаимодействия пользователя с ботом:

– загрузка документов,

– выбор раздела,

– анализ,

– генерация отчёта.

Также была добавлена валидация форматов файлов (PDF).

3) Модуль загрузки и обработки документов:

Было настроено временное хранилище для файлов и написана логика сохранения в него загруженных файлов с уникальными закодированными именами;

4) Система создания отчётов:

Была реализована генерация отчётов в pdf, txt, xls форматах (приложение Ж).

5) Сценарий ответа на вопросы по документу:

Была добавлена кнопка «Задать вопрос по документу», реализующая сценарий ответа нейросети по содержанию pdf файла. Также реализован сам алгоритм запроса и получения ответа от нейросети с помощью openai api.

Тестирование и доработка:

1) Фиксы и доработка скрипта бота:

Были исправлены ошибки навигации (добавлена кнопка «Назад», добавлено удаление клавиатур после перехода на следующую).

2) Оптимизация запросов к API:

Оптимизирована работа с большими файлами (уменьшено время обработки с 25 до 5–7 минут за счёт параллелизации запросов).

3) Доработка сценария ответа на вопрос по содержанию:

По требованию заказчика в алгоритме получения ответа от нейросети были реализованы правила:

– Если вопрос слишком общий, не содержит конкретики, не относится к документу, непонятен или бессмыслен, то нейросеть не должна искать ответ на такой вопрос в документе;

– Если вопрос корректен, то должны быть использованы только предоставленные фрагменты, также в ответе должны быть указаны страницы, а при запросе сравнений должны быть указаны оба источника.

4) Хостинг финальной версии бота:

Финальная версия бота была развёрнута на облачном хостинге «Amvera Cloud» [12]. Настроен веб-хук для Telegram.

1.5.3 Отчёт о работе участника. Соболев Егор Владимирович

Аналитический этап:

1) Разработка workflow-диаграммы:

Была создана WFD-диаграмма, описывающая этапы взаимодействия пользователя и бота при различных сценариях (рисунок 4).

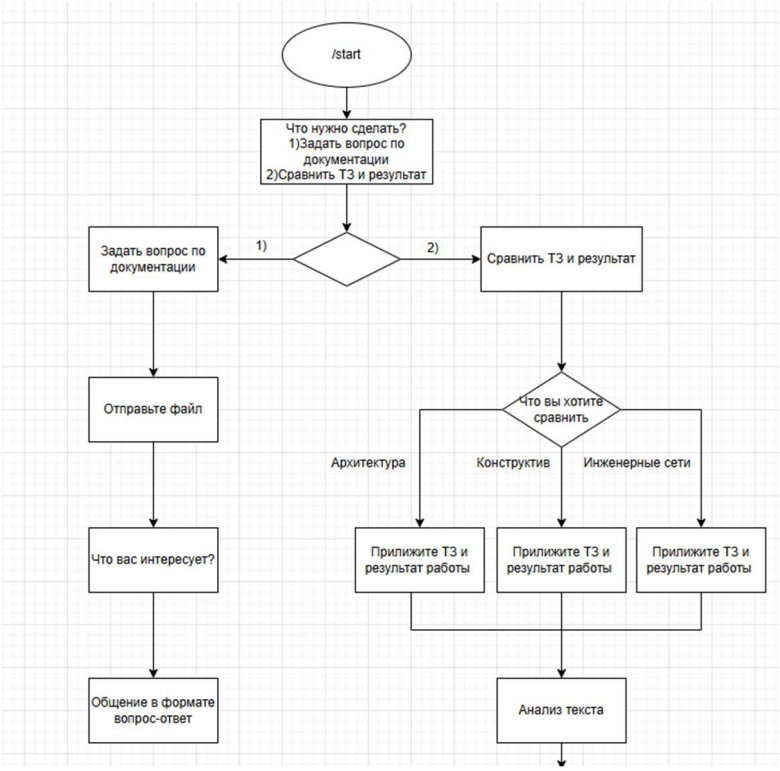


Рисунок 4 – WFD-диаграмма

2) Создание структуры базы данных:

Была спроектирована структура базы данных, включающая: таблицы для пользователей, документов, настроек анализа, истории запросов, результатов анализа. Структура представлена на рисунке 5.

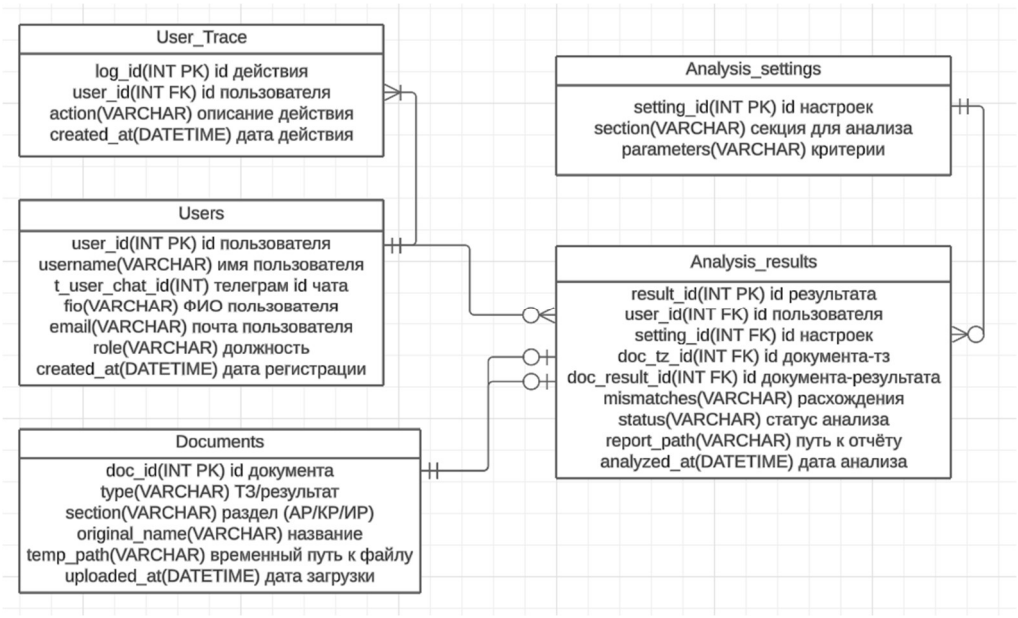


Рисунок 5 – Структура базы данных

Разработка:

1) Интеграция SQLite:

Была реализована асинхронная работа с базой данных через aiosqlite. Настроено хранение результатов анализа и метаданных документов.

2) Реализация стандарта общения сервер-клиент:

Разработана схема API для передачи данных между ботом и сервером (форматы запросов/ответов в JSON).

3) Внедрение базы данных в работу бота и тестирование её работы:

Был написан вспомогательный класс, содержащий в себе все необходимые методы для получения, изменения и записи данных в базу. Проведены unit-тесты (приложение Г) на корректность CRUD-операций.

Тестирование и доработка:

1) Тестирование базового сравнения:

Были выявлены многочисленные ошибки в привязке страниц (например, ссылки на несуществующие страницы), различия в отчётах при анализе одних и тех же файлов. Была оказана помощь в исправлении логики маркировки частей текста.

2) Оптимизация БД:

Запросы к базе данных были ускорены за счёт индексации, за счёт чего было уменьшено время работы промежуточных этапов алгоритма (где происходит сохранение данных) на 10 – 15%.

3) Тестирование на mock-сервере:

Была проверена работа системы при высокой нагрузке (параллельная обработка 10 документов). Тестирование показало успешность работы алгоритма при достаточном балансе аккаунта openai (если хватает оплаченных токенов).

1.5.4 Отчёт о работе участника. Тимофеев Иван Дмитриевич

Аналитический этап:

1) Определение целевой аудитории и критериев сдачи продукта:

Были выделены следующие категории пользователей, составляющих целевую аудиторию:

– Архитекторы,

– Проектировщики,

– Инженеры,

– Малые и средние архитектурные и строительные компании;

– Заказчики строительства (девелоперы).

Выделены следующие критерии сдачи продукта:

– Разработать интерфейс взаимодействия с пользователем на основе bp-модели и wfd-диаграммы, то есть: выбор раздела для анализа, загрузка файлов, вопросы по документу и так далее;

– Реализовать механизм парсинга необходимых частей документа;

– Интегрировать DeepSeek/ChatGPT для анализа и сравнения документов;

– Реализовать механизм составления отчёта по анализу;

– Реализовать механизм хранения всех необходимых данных на сервере (определить стандарт общения сервера и клиента, разработать схему запроса);

– Провести тестирование и убедиться, что бота можно эксплуатировать.

2) Разработка bp-модели:

Была создана bp-модель (приложение В), удовлетворяющая необходимым требованиям заказчика и хранящая информацию о необходимых исполнителях, этапах, промежуточных результатах процесса использования продукта пользователем.

Разработка:

1) Реализация механизма извлечения и чанкинг текста:

Был реализован алгоритм разбиения текста на части с сохранением структуры. Использован PyPDF2 и кастомный токенизатор. Часть функций, созданных для извлечения текста, представлены в листинге:

```
async def extract_text_from_pdf(pdf_path: str) -> str:
    """Асинхронная обёртка для синхронного извлечения текста из PDF"""
    loop = asyncio.get_event_loop()
    return await loop.run_in_executor(None, _sync_extract_text_from_pdf,
                                      pdf_path)
def _sync_extract_text_from_pdf(pdf_path: str) -> str:
    """Синхронная реализация извлечения текста из PDF"""
    reader = PdfReader(pdf_path)
    text = []
    for page_num, page in enumerate(reader.pages, start=1):
        content = page.extract_text() or ""
        if page_num <= len(reader.pages):
```

```
            text.append(f"\n===  НАЧАЛО  СТРАНИЦЫ  {page_num}
===\n{content}\n"
                        f"=== КОНЕЦ СТРАНИЦЫ {page_num} ===")
    return "\n".join(text)
```

2) Реализация базового функционала сравнения документов:

Был разработан двухэтапный алгоритм анализа. На первом этапе текст документов разбивается на смысловые фрагменты с сохранением структуры страниц. Для каждого фрагмента создаются векторные представления с помощью OpenAIEmbeddings. Поиск схожих участков между ТЗ и проектной документацией выполняется через FAISS (по 3 ближайших соответствия для каждого фрагмента). На втором этапе найденные пары фрагментов передаются в GPT-4o-mini для семантического сравнения. Запросы формируются динамически в зависимости от типа раздела (архитектурный, инженерный и др.) с обязательным учётом контекста страниц.

3) Детализированный анализ расхождений:

В систему был добавлен категорированный анализ по разделам документа:

– Для архитектурных решений: сравнение материалов стен, толщин конструкций;

– Для инженерных систем: проверка оборудования (производители, технические параметры);

– Для конструктивных решений: анализ соответствия бетонных смесей (например, "B20" против "класса B20").

Все выявленные несоответствия привязываются к страницам исходных документов в формате "ТЗ стр. X / РР стр. Y". Это позволяет пользователям быстро находить проблемные места в исходных файлах.

Тестирование и доработка:

1) Доработка алгоритма сравнения:

Основные улучшения были сосредоточены на трёх направлениях:

– Оптимизация обработки контекста: увеличение размера обрабатываемого блока до 128k токенов и реализация механизма чанкинга с перекрытием (overlap=5000 токенов) для сохранения связности анализа.

– Точное извлечение требований: усовершенствовано извлечение разделов ТЗ через комбинацию регулярных выражений и LLM. Добавлена чёткая маркировка страниц ("=== НАЧАЛО СТРАНИЦЫ X ===") для корректной привязки результатов.

– Структурирование отчётов: внедрён механизм пост-обработки, который устраняет дубликаты и объединяет схожие пункты. Отчёты теперь имеют чёткую структуру и состоят из трёх разделов: ключевые несоответствия с группировкой по разделам, технические ошибки, нарушения нормативных требований

2) Оптимизация сравнения:

Было внедрено кэширование эмбеддингов, что ускорило семантический поиск примерно на 50% раза.

3) Создание отчёта и презентации:

Были подготовлены материалы для защиты проекта: отчёт и презентация.

ЗАКЛЮЧЕНИЕ

Разработанный чат-бот соответствует ключевым требованиям заказчика, заявленным в техническом задании, демонстрируя:

– Автоматизацию семантического анализа: в отличие от ручных методов и Diff-инструментов, бот выявляет контекстуальные расхождения (например, синонимичные обозначения материалов «бетон В20» и «класс В20»);

– Интеграцию с пользовательскими сценариями: telegram-интерфейс обеспечил доступность решения без сложного внедрения, а система интерактивных клавиатур упростит навигацию для специалистов строительной отрасли;

– Гибкость генерации отчётов: формирование структурированных выводов в PDF/XLS/TXT с привязкой к страницам источников позволяет быстро локализовать проблемы в документации.

Реализованы все уровни функционала:

– Минимальный: Загрузка документов, базовый анализ;

– Базовый: Категорированная проверка (архитектурный, инженерный разделы), экспорт результатов;

– Оптимальный: Точность анализа достигла 85–90% на тестовых данных благодаря комбинации FAISS и GPT-4o mini.

Результаты тестирования выявили следующие ключевые аспекты:

1) Стабильность:

– После оптимизации время обработки 100 страниц сокращено до 5–7 минут (против изначальных 25);

– Механизм повторных попыток API-запросов снизил частоту ошибок примерно на 40%.

2) Дефекты и их влияние:

– Ложные срабатывания (5–7% случаев): Например, некорректное сопоставление параметров вентиляции из-за различий в терминологии. Это требует дополнительной верификации результатов пользователем;

– Ошибки привязки страниц (2–3% случаев): Связаны с особенностями маркировки текста в PDF. Могут затруднить локализацию несоответствий, но не блокируют основной функционал.

3) Безопасность данных:

– Временное хранилище и автоматическая очистка файлов обеспечивают базовую конфиденциальность, но отсутствие сквозного шифрования может стать риском для документов с повышенной секретностью.

Предложения по улучшению и развитию

1) Расширение функционала:

– Поддержка форматов DOCX и изображений (OCR) для обработки сканированных документов;

– Интеграция с облачными хранилищами (Google Drive, Яндекс.Диск) для прямого импорта файлов.

2) Повышение точности:

– Добавление модуля проверки соответствия ГОСТам и СП с автоматическим обновлением нормативной базы;

– Использование ансамбля моделей (например, комбинация GPT-4o и специализированных RuBERT для строительной терминологии).

3) Оптимизация пользовательского опыта:

– Внедрение дашборда с визуализацией статистики по проектам;

– Добавление примеров корректных вопросов для снижения доли некорректных запросов к ИИ.

4) Масштабируемость:

– Переход на распределённую архитектуру (микросервисы) для обработки нескольких документов параллельно;

– Внедрение подписки с тарификацией по объёму анализа для коммерциализации решения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Telegram Bot API [Электронный ресурс] / Telegram. – 2025. – URL: https://core.telegram.org/bots/api (дата обращения: 26.03.2025).

2. Документация DiffPDF [Электронный ресурс] / M. Summerfield. – 2025. – URL: https://mark-summerfield.github.io/diffpdf.html (дата обращения: 26.03.2025).

3. Технический отчёт OpenAI GPT-4 [Электронный ресурс] / OpenAI. – 2023. – URL: https://cdn.openai.com/papers/gpt-4.pdf (дата обращения: 02.04.2025).

4. Документация Autodesk Revit API [Электронный ресурс] / Autodesk. – URL: https://help.autodesk.com/view/RVT/2025/ENU/?guid=Revit_API_Revit_API_Developers_Guide_Introduction_Getting_Started_Using_the_Autodesk_Revit_API_html (дата обращения: 01.04.2025).

5. Документация PyPDF2 [Электронный ресурс] / M. Fenniak. – 2008. – URL: https://pypdf2.readthedocs.io/en/3.x/ (дата обращения: 07.04.2025).

6. FAISS Github репозиторий [Электронный ресурс] / Facebook Research. – 2023. – URL: https://github.com/facebookresearch/faiss (дата обращения: 08.04.2025).

7. Документация Aiogram [Электронный ресурс] / Aiogram Team. – 2025. – URL: https://docs.aiogram.dev/ (дата обращения: 07.04.2025).

8. Документация SQLite [Электронный ресурс] / D.R. Hipp. – 2025. – URL: https://www.sqlite.org/docs.html (дата обращения: 09.04.2025).

9. Справка OpenAI API [Электронный ресурс] / OpenAI. – 2023. – URL: https://platform.openai.com/docs/api-reference (дата обращения: 20.05.2021).

10. Документация DeepSeek API [Электронный ресурс] / DeepSeek. – 2023. – URL: https://api-docs.deepseek.com/ (дата обращения: 08.04.2025).

11. Технический обзор GPT-4 [Электронный ресурс] / OpenAI. – 2023. – URL: https://openai.com/research/gpt-4 (дата обращения: 02.04.2025).

12. Документация Amvera Cloud [Электронный ресурс] / Amvera Team. – 2025. – URL: http://www.ecsoc.msses.ru/pdf/ecsoc003.pdf (дата обращения: 08.05.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

Техническое задание на разработку

Чат-бот предполагает сравнение технического задания от заказчика, где есть информация по проектируемому объекту и проектной документации. Анализ проводится с помощью ИИ.

ИИ должна изучить и проанализировать всю информацию по объекту и выполнить сравнение ТЗ от заказчика и Проектной документации.

Перед загрузкой файлов бот должен предложить список того, что именно нужно сравнивать:

- Архитектурный раздел,
- Конструктивный раздел,
- Инженерный раздел.

После того как пользователь выбрал нужный раздел, он подгружает необходимый документ. Бот производит анализ при помощи ИИ, все сравнивает и сопоставляет. После чего пользователь получает результат сравнения, который можно выгрузить в форматах: .pdf, .txt, .xls.

Задача на семестр – полностью создать данного бота, который может выполнять данный функционал.

Задача №1. Сравнение Архитектурного раздела с готовым ТЗ. Бот проводит сравнение всех материалов в стенах и полах. Кроме материала, так же сравнивает толщину этих стен.

Задача №2. Сравнение Конструктивного раздела с готовым ТЗ. Сравниваются материалы стен.

Задача №3. Сравнение Инженерного раздела с готовым ТЗ. Сравниваются производители оборудования отопления, вентиляции, кондиционирования.

ПРИЛОЖЕНИЕ Б

(обязательное)

Бэклог задач проекта

1) Аналитический этап (1 контрольная точка):

– Определение целевой аудитории и критериев сдачи (Тимофеев И. Д.);

– Анализ текущих методов решения проблемы (ручное сравнение, Diff-инструменты) (Косторной Д. В.);

– Разработка workflow-диаграммы (Соболев Е. В.);

– Разработка модели бизнес-процессов (Тимофеев И. Д.);

– Создание интерфейса, примера взаимодействия бота и пользователя (Косторной Д. В.);

– Создание структуры базы данных (Соболев Е. В.);

– Разработка плана механизма анализа документов (Косторной Д. В.).

2) Разработка (2 контрольная точка):

– Реализация скелета-бота (Косторной Д. В.);

– Проработка и реализация пользовательских сценариев (Косторной Д. В.);

– Реализация механизма извлечения и разбиения текста из файлов (Тимофеев И. Д.);

– Выбор базы данных и тестовая связь со скриптами (Соболев Е. В.);

– Реализация стандарта общения сервера и клиента, разработка схемы запроса (Соболев Е. В.);

– Разработка модуля загрузки и обработки документов (Косторной Д. В.);

– Реализация базового функционала сравнения документов (Тимофеев И. Д.);

– Внедрение базы данных в работу бота и тестирование её работы (Соболев Е. В.);

– Реализация системы создания отчётов (Косторной Д. В.);

– Реализация детализированного анализа расхождений (Тимофеев И. Д.);

– Реализация сценария с обычным вопросом по документу (Косторной Д. В.).

3) Разработка 2.0 – тестирование и доработка (3 контрольная точка):

– Тестирование базового сравнения (Соболев Е. В.);

– Фиксы и доработка скрипта бота (Косторной Д. В.);

– Доработка алгоритма сравнения (доп. категории) (Тимофеев И. Д.);

– Оптимизация БД (Соболев Е. В.);

– Доработка алгоритма ответа на вопрос по содержанию (Косторной Д. В.);

– Оптимизация сравнения (Тимофеев И. Д.);

– Проведение тестирования на mock-сервере (Соболев Е. В.);

– Хостинг финальной версии бота (Косторной Д. В.);

– Создание отчёта и презентации для защиты (Тимофеев И. Д.).

ПРИЛОЖЕНИЕ В

(обязательное)

BPMN-диаграмма основного процесса

BPMN-диаграмма основного процесса представлена на рисунках 6 – 7.

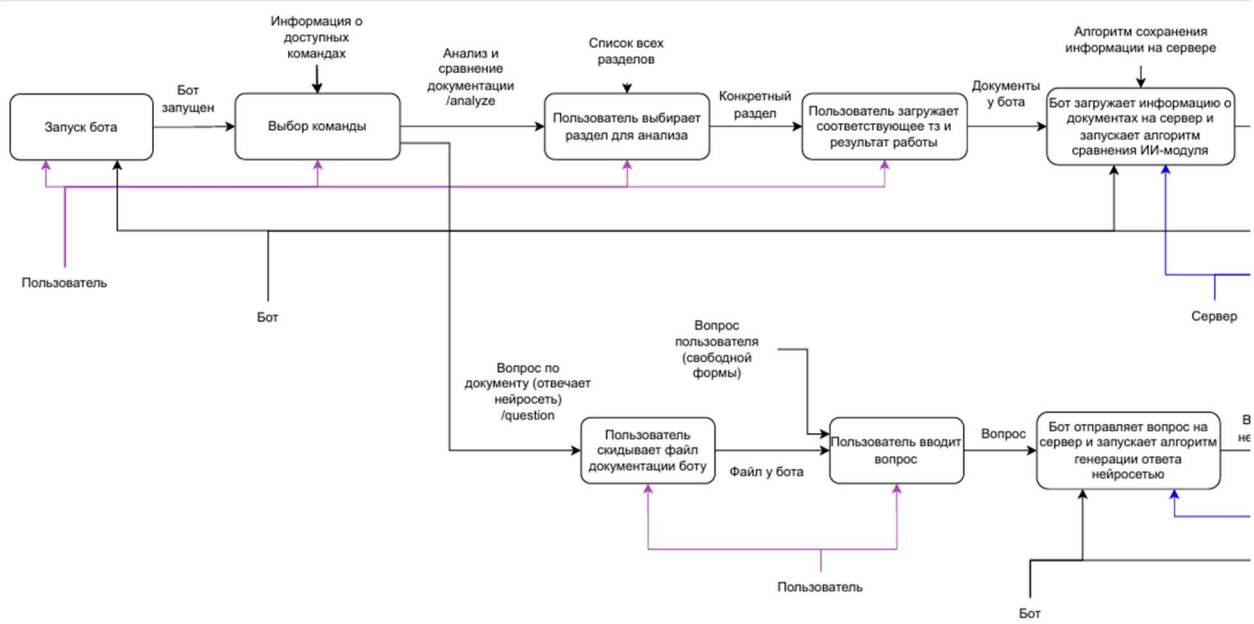


Рисунок 6 – Фрагмент BPMN-диаграммы основного процесса (1)

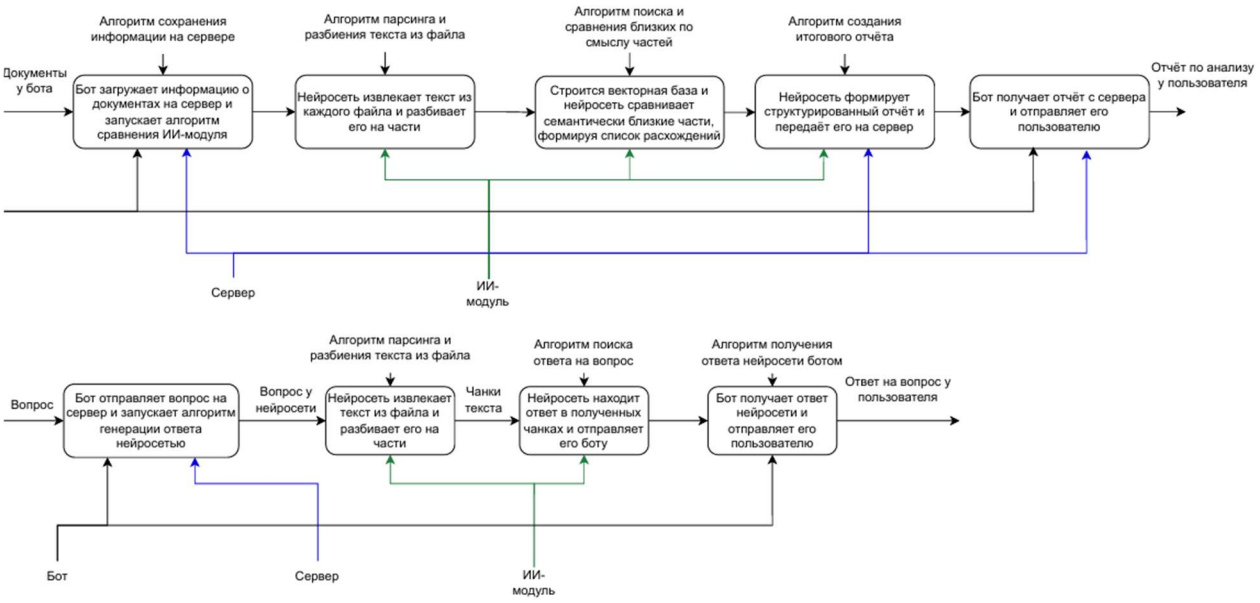


Рисунок 7 – Фрагмент BPMN-диаграммы основного процесса (2)

ПРИЛОЖЕНИЕ Г

(обязательное)

Листинг части unit-тестов

Листинг части unit-тестов для тестирования работы скрипта базы данных:

```
import unittest
import asynctest
from unittest.mock import patch, AsyncMock, MagicMock
from database import Database, db, project_dir, db_file
import os
from aiosqlite import connect, Row
import sqlite3

class TestDatabase(asynctest.TestCase):
    async def tearDown(self):
        if os.path.exists(self.test_db_path):
            os.remove(self.test_db_path)

    async def test_get_user(self):
        async with self.db.get_connection() as conn:
            await conn.execute(
                "INSERT INTO Users (username, t_user_chat_id, cre-
ated_at) VALUES (?, ?, DATETIME('now'))",
                ('test_user', 123)
            )
            await conn.commit()
        user = await self.db.get_user(123)
        self.assertIsNotNone(user)
        self.assertEqual(user['username'], 'test_user')
        self.assertEqual(user['t_user_chat_id'], 123)
```

```
    async def test_create_user(self):
        await self.db.create_user('new_user', 456)
        user = await self.db.get_user(456)
        self.assertIsNotNone(user)
        self.assertEqual(user['username'], 'new_user')

    async def test_update_user(self):
        await self.db.create_user('user_to_update', 789)
        await self.db.update_user(789, username='updated_user')
        user = await self.db.get_user(789)
        self.assertEqual(user['username'], 'updated_user')

    async def test_add_question(self):
        await self.db.create_user('question_user', 101112)
        await self.db.add_question(101112, 'Вопрос?', 'Ответ.')
        questions = await self.db.get_user_questions(101112)
        self.assertEqual(len(questions), 1)
        self.assertEqual(questions[0]['question'], 'Вопрос?')
        self.assertEqual(questions[0]['answer'], 'Ответ.')

    async def test_add_analysis_result(self):
        await self.db.create_user('analysis_user', 131415)
        await self.db.add_analysis_result(131415, 'АР', 'tz.pdf',
'result.pdf', 'Несоответствия: ...')
        history = await self.db.get_user_analysis_history(131415)
        self.assertEqual(len(history), 1)
        self.assertEqual(history[0]['section'], 'АР')
        self.assertEqual(history[0]['doc_tz_file'], 'tz.pdf')
        self.assertEqual(history[0]['doc_result_file'],     're-
sult.pdf')
```

ПРИЛОЖЕНИЕ Д

(обязательное)

Таблица трудозатрат участников

Таблица 2 – Таблица трудозатрат участников команды

Задача	Исполнитель	Объем трудозатрат в часах
Определение целевой аудитории/критериев сдачи	Тимофеев И. Д.	2
Анализ того, как проблема решается сейчас	Косторной Д. В.	2
Разработка WFD диаграммы использования продукта	Соболев Е. В.	3
Разработка BP-модели использования продукта	Тимофеев И. Д.	3
Создание интерфейса/примера взаимодействия бота и пользователя	Косторной Д. В.	5
Создание структуры базы данных	Соболев Е. В.	2
Составление тех. задания	Тимофеев И. Д.	2
Разработка плана механизма анализа документов	Косторной Д. В.	2
Выбор базы данных для оптимальной работы и тестовая связь со скриптами	Соболев Е. В.	8
Реализация скелета бота	Косторной Д. В.	7
Проработка пользовательских сценариев	Косторной Д. В.	10
Реализация механизма извлечения и разбиения текста из файлов	Тимофеев И. Д.	3
Реализация стандарта общения сервера и клиента, разработка схемы запроса	Соболев Е. В.	7
Разработка модуля загрузки и обработки документов	Косторной Д. В.	5
Реализация базового функционала сравнения документов	Тимофеев И. Д.	12

Продолжение таблицы 2

Внедрение базы данных в работу бота и тестирование её работы	Соболев Е. В.	11
Реализация системы создания отчётов	Косторной Д. В.	5
Реализация детализированного анализа расхождений	Тимофеев И. Д.	13
Реализация сценария с обычным вопросом по документу	Косторной Д. В.	7
Тестирование базового сравнения	Соболев Е. В.	7
Фиксы и доработка скрипта бота	Косторной Д. В.	6
Доработка алгоритма сравнения (доп. категории)	Тимофеев И. Д.	8
Оптимизация БД	Соболев Е. В.	10
Доработка алгоритма ответа на вопрос по содержанию	Косторной Д. В.	5
Оптимизация сравнения	Тимофеев И. Д.	4
Проведение тестирования на mock-сервере	Соболев Е. В.	6
Хостинг финальной версии бота	Косторной Д. В.	4
Разработка отчёта и презентации для защиты	Тимофеев И. Д.	7

ПРИЛОЖЕНИЕ Е

(обязательное)

Листинг обработчиков бота

Листинг небольшой части обработчиков событий в боте:

```
@router.callback_query(F.data == 'analyze_docs')
async def choose_category(callback: types.CallbackQuery, state: FSMContext):
    await callback.message.edit_text(
        "Выберите раздел для сравнения:",
        reply_markup=KeyboardBuilder.categories_kb()
    )
    await state.set_state(ComparisonStates.choosing_category)
@router.callback_query(F.data == 'eng_options')
async def choose_category(callback: types.CallbackQuery, state: FSMContext):
    await callback.message.edit_text(
        "Выберите конкретный раздел инженерных решений:",
        reply_markup=KeyboardBuilder.eng_options_kb()
    )
    await state.set_state(ComparisonStates.choosing_category)
@router.callback_query(F.data == 'start')
async def back_to_start(callback: types.CallbackQuery, state: FSMContext):
    await state.clear()
    try:
        await callback.message.delete()
    except:
        pass
    await callback.message.answer(
        "Привет! Добро пожаловать в бота.", reply_markup=KeyboardBuilder.start_kb())
```

ПРИЛОЖЕНИЕ Ж

(обязательное)

Примеры отчётов в разных форматах

Примеры отчётов в разных форматах представлены на рисунках 8 – 10.

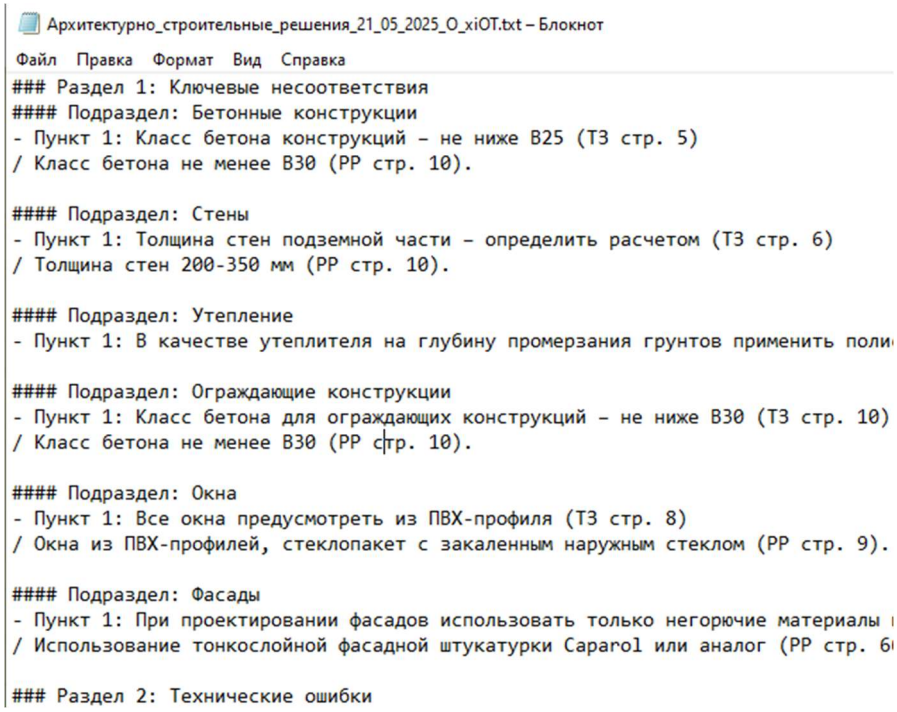
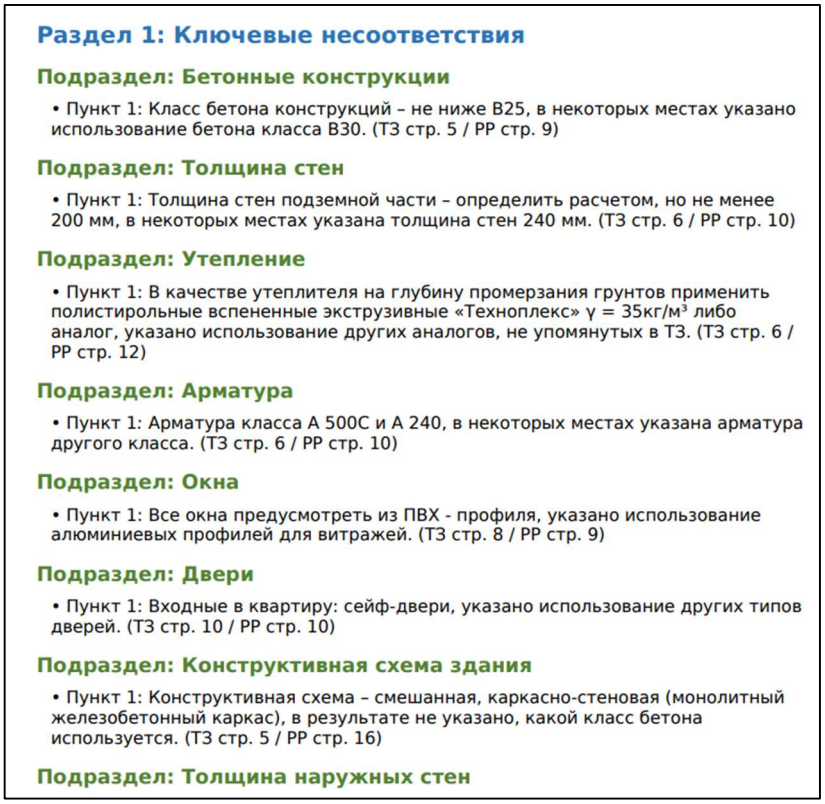


Рисунок 8 – Отчёт в txt формате

	A	B	C	
1	Раздел	Подраздел	Тип проблемы	Описание
2	Раздел 1: Ключевые несоответствия	Подраздел: Арматура	Несоответствие	Пункт 1: Несоответствие в арматуре: Арматура класса A500C и A240, в неко
3	Раздел 1: Ключевые несоответствия	Подраздел: Бетонные конструкции	Несоответствие	Пункт 1: Несоответствие в классе бетона для стен и перекрытий: Класс бет
4	Раздел 1: Ключевые несоответствия	Подраздел: Бетонные конструкции	Несоответствие	Пункт 2: Несоответствие в классе бетона для несущих конструкций: Класс б
5	Раздел 1: Ключевые несоответствия	Подраздел: Стены	Несоответствие	Пункт 1: Несоответствие в толщине стен: Толщина стен подземной части до
6	Раздел 1: Ключевые несоответствия	Подраздел: Стены	Несоответствие	Пункт 2: Несоответствие в толщине стен: Толщина стен подземной части до
7	Раздел 1: Ключевые несоответствия	Подраздел: Утеплитель	Несоответствие	Пункт 1: Несоответствие в материалах утеплителя: Указано использование
8	Раздел 1: Ключевые несоответствия	Подраздел: Утеплитель	Несоответствие	Пункт 2: Несоответствие в использовании материалов для утепления: В кач
9	Раздел 1: Ключевые несоответствия	Подраздел: Фасады	Несоответствие	Пункт 1: Несоответствие в системе фасадов: Указано, что при проектирова
10	Раздел 1: Ключевые несоответствия	Подраздел: Фасады	Несоответствие	Пункт 2: Несоответствие в использовании огнестойких материалов: Все при
11	Раздел 2: Технические ошибки	Общее	Ошибка	Несоответствие в использовании аналогов: Применение аналогов допуска
12	Раздел 3: Нарушения нормативных требований	Общее	Нарушение	Несоответствие в использовании огнестойких материалов: Все применяем

Рисунок 9 – Отчёт в xls формате



Раздел 1: Ключевые несоответствия

Подраздел: Бетонные конструкции

• Пункт 1: Класс бетона конструкций – не ниже В25, в некоторых местах указано использование бетона класса В30. (ТЗ стр. 5 / РР стр. 9)

Подраздел: Толщина стен

• Пункт 1: Толщина стен подземной части – определить расчетом, но не менее 200 мм, в некоторых местах указана толщина стен 240 мм. (ТЗ стр. 6 / РР стр. 10)

Подраздел: Утепление

• Пункт 1: В качестве утеплителя на глубину промерзания грунтов применить полистирольные вспененные экструзивные «Технoплекс» γ = 35кг/м³ либо аналог, указано использование других аналогов, не упомянутых в ТЗ. (ТЗ стр. 6 / РР стр. 12)

Подраздел: Арматура

• Пункт 1: Арматура класса А 500С и А 240, в некоторых местах указана арматура другого класса. (ТЗ стр. 6 / РР стр. 10)

Подраздел: Окна

• Пункт 1: Все окна предусмотреть из ПВХ - профиля, указано использование алюминиевых профилей для витражей. (ТЗ стр. 8 / РР стр. 9)

Подраздел: Двери

• Пункт 1: Входные в квартиру: сейф-двери, указано использование других типов дверей. (ТЗ стр. 10 / РР стр. 10)

Подраздел: Конструктивная схема здания

• Пункт 1: Конструктивная схема – смешанная, каркасно-стеновая (монолитный железобетонный каркас), в результате не указано, какой класс бетона используется. (ТЗ стр. 5 / РР стр. 16)

Подраздел: Толщина наружных стен

Рисунок 10 – Отчёт в pdf формате