

«Национальный исследовательский университет
«Высшая школа экономики»

Лицей

Индивидуальная выпускная работа

Отчёт по IT-проекту LyceumDocBot

Выполнил Воловиков Дмитрий Сергеевич

Москва 2026

1. Введение

LyseumDocBot — веб-приложение, которое помогает быстро находить ответы в официальных документах лицея. Пользователь задаёт вопрос в формате чата или выполняет поиск по ключевой фразе, а система находит релевантные фрагменты документов и показывает результат вместе со списком источников.

Проект появился из практической потребности: нужные правила и регламенты часто находятся в длинных PDF-файлах. Даже если документ можно открыть и поискать по нему вручную, на это уходит время, а формулировки в документах не всегда совпадают со словами, которыми обычно задают вопрос. В итоге ученики и преподаватели тратят время на ручной поиск, а одни и те же вопросы повторяются.

2. Проблемное поле и предпроектное исследование

Проблема, которую решает мой проект — быстро и без ошибок получить ответ на вопрос по внутренним правилам и процедурам лицея. Информация о пересдачах, сроках и требованиях к итоговым работам, правилах пропусков, оформлении заявлений и других регламентах распределена по нескольким документам и разделам. При этом формулировки часто канцелярские, а нужный пункт может находиться в большом PDF или на отдельной странице сайта. В результате пользователь тратит время на ручной поиск, может пропустить важные детали или неверно интерпретировать правило. Для преподавателей и кураторов это дополнительно превращается в повторяющиеся вопросы, на которые приходится отвечать «вручную».

2.1 Предпроектное исследование

Чтобы убедиться, что проблема действительно актуальна, я провёл небольшое предпроектное исследование: короткие интервью и опрос через онлайн-форму.

Качественное исследование (интервью)

Я поговорил с двумя участниками образовательного процесса. Оба отметили, что недавно сталкивались с ситуацией, когда нужно было быстро найти правило или пункт регламента, и приходилось вручную листать несколько PDF или страницы сайта.

Ключевые выводы по интервью:

- важна скорость получения ответа (не тратить время на поиск по нескольким документам);
- критично видеть источник — ссылку на конкретный фрагмент документа;
- полезно, когда система честно пишет, что подтверждения в документах нет, и предлагает уточнить вопрос.

Количественное исследование (опрос)

Я провёл опрос среди 56 человек через онлайн-форму. Большинство респондентов поддержали идею сервиса и подтвердили, что периодически испытывают трудности с поиском информации в официальных документах.

По свободным комментариям чаще всего упоминались вопросы про пересдачи, требования к проектным/итоговым работам, пропуски и оформление справок/заявлений.

Таким образом, результаты исследования подтверждают, что задача поиска по документам актуальна, а формат «вопрос → ответ со ссылкой на источник» воспринимается пользователями как полезный.

3. Целевая аудитория

Целевая аудитория проекта:

- Учащиеся 8–11 классов — быстро получать справку по правилам и требованиям без чтения длинных документов.
- Преподаватели и кураторы — оперативно сверяться с регламентами и давать корректные ответы.
- Администрация (вторичная аудитория) — снижение нагрузки на сотрудников за счёт автоматизации ответов на частые вопросы и централизованного управления базой документов.

4. Функциональные требования

Основные функциональные требования (FR), реализованные в проекте:

- FR1. Аккаунт пользователя: регистрация, вход/выход, профиль, смена пароля.
- FR2. Чат-запрос: пользователь задаёт вопрос в интерфейсе чата.
- FR3. Ответ с источниками: система выдаёт ответ и список фрагментов-источников ($S_1 \dots S_k$).
- FR4. Просмотр источников: переход к документу и просмотр извлечённого текста.
- FR5. Поиск по документам: поиск по ключевой фразе с выдачей релевантных фрагментов.
- FR6. История запросов: просмотр прошлых запросов и повторная обработка (новая версия ответа).
- FR7. Экспорт: сохранение ответа и источников в PDF.

- FR8. Загрузка документов (admin): загрузка файла и запуск обработки/индексации, статусы indexing/review/error.
- FR9. Модерация документов (admin): публикация/отклонение и удаление.
- FR10. Управление пользователями (admin): роли user/admin, блокировка/разблокировка, сброс пароля администратором.

5. Архитектура и стек технологий

Проект реализован как full-stack веб-приложение.

Backend: Python + FastAPI, REST-эндпоинты, работа с БД.

Frontend: Next.js (React) + TypeScript.

Хранилище: PostgreSQL в продакшене; SQLite используется в CI/тестах.

Экспорт: генерация PDF на сервере (reportlab).

Развёртывание: Docker / docker-compose.

Качество сборки: GitHub Actions (тесты и проверки).

6. Этапы работы над проектом

Этапы описаны по фактическому ходу разработки. Даты можно уточнить по истории коммитов.

Этап	Содержание работ	Сроки
1	Проектирование: требования (FR), сценарии, схема данных, базовая архитектура.	01.12.2025-10.12.2025
2	Backend: авторизация/роли, работа с документами и историей, экспорт PDF.	11.12.2025-20.12.2025
3	Поиск/ответы: индексация документов, выдача источников; интеграция с интерфейсом.	21.12.2025-08.01.2026
4	Frontend: чат, поиск, история, админ-разделы.	09.01.2026-11.01.2026
5	CI/деплой: GitHub Actions, Docker-сборка, исправление тестов и линтера.	12.01.2026-13.01.2026

7. Краткий анализ аналогов

Поиск по сайту и ручной просмотр PDF (Ctrl+F) требует заранее знать, где искать, и работает только по буквальному совпадению. В результате пользователь может пропустить нужное место из-за разных формулировок.

Универсальные ассистенты могут отвечать быстро, но не гарантируют соответствие именно внутренним правилам лица, потому что не опираются на конкретный корпус документов и могут допускать галлюцинации.

FAQ-боты на заранее подготовленных ответах требуют ручного обновления и обычно покрывают ограниченный набор сценариев.

LuciumDocBot ориентирован на работу с локальным корпусом документов и показывает источники, по которым сформирован ответ.

8. Рефлексия

8.1. Проблемы и способы их решения

- Самое неприятное в разработке было «свести» формат источников между backend и frontend. Сначала в ответе и в панели источников данные лежали по-разному, из-за этого то пропадали поля, то ломалась верстка. В итоге я зафиксировал один понятный формат (id источника, документ, фрагмент текста, служебные поля) и привёл к нему оба слоя.

- С RAG оказалось сложнее, чем я ожидал. На старте retrieval часто выдавал «почти одинаковые» куски или нерелевантные фрагменты — из-за грязного текста после извлечения из PDF, неудачного разбиения на чанки и отсутствия нормального дебага. Я добавил очистку, пересмотрел chunking (размер/перекрывание) и сделал debug-скрипт, который показывает топ-кандидатов со скорями.
- Ещё одна боль — «привязка» ответа к источникам. LLM легко пишет убедительно, но может «съехать» от документов. Поэтому я сделал правило: в ответе должны быть ссылки вида [S#], а если подтверждения нет — честное «не знаю». Плюс — простая проверка корректности ссылок и сохранение источников вместе с ответом.
- CI оказался сюрпризом: локально всё работало, а в GitHub Actions тесты падали из-за отсутствующих зависимостей (например, reportlab для PDF). Пришлось пройтись по импортам, дописать зависимости и проверять сборку в максимально «чистом» окружении.
- Ещё одна боль — линтер на фронте. В какой-то момент `next lint` запускался интерактивно и просил выбрать конфигурацию ESLint. Я добавил готовый конфиг и настроил workflow так, чтобы линт работал одинаково и без вопросов.

8.2. Приобретённые навыки и их применение

- Я почувствовал, что действительно научился собирать продукт: FastAPI на бэкенде + Next.js на фронте, авторизация, экраны чата/поиска/источников, админка и интеграция всего этого в одну систему.

- С БД стало проще, когда я начал думать от сценариев: какие действия делает пользователь — такие сущности и нужны (пользователи, документы и их статусы, история запросов, версии ответов, связи с источниками). Это помогло держать логику понятной и не раздувать схему.
- Самый полезный опыт — именно RAG-инженерия. Я научился разбивать систему по слоям: извлечение текста → чанкинг → retrieval → ранжирование → генерация с цитатами. И понял, что качество чаще всего «падает» не в LLM, а раньше — в тексте и поиске.
- Я научился делать ответы проверяемыми: фиксировать источники S1..Sk, требовать ссылки [S#] в тексте и честно отвечать «не знаю», если подтверждения нет. Это повышает доверие к продукту и делает поведение предсказуемым.

8.3. Что можно было бы сделать по-другому

- Если бы начинал заново, я бы с самого начала завёл небольшой «эталонный» набор: 20–30 типичных вопросов и ожидаемые источники/фрагменты. Тогда можно было бы автоматически проверять качество retrieval и не гадать, стало лучше или хуже после очередного изменения чанкинга или индекса.
- Я бы раньше построил наблюдаемость для RAG: логировать кандидатов (текст, метаданные, scores), сохранять промпт/контекст и хранить отладочную трассу по каждому запросу. При наличии такого трейсинга многие баги находятся в разы быстрее.
- Я бы больше времени вложил в подготовку корпуса: чистку текста, устранение дублей, сохранение структуры (заголовки/разделы/страницы). На практике именно качество текста и метаданных сильнее всего влияет на то, насколько «умно» выглядит система.

- Я бы раньше выделил retrieval в отдельный модуль с конфигом и флагами (BM25/вектора/гибрид, top-k, переранжирование), чтобы проще было крутить параметры и не трогать остальной код. Тогда настройка качества шла бы быстрее и аккуратнее.
- И параллельно я бы раньше закрепит прод-настройки: строгие переменные окружения, отключение лишних эндпоинтов в проде и отдельный режим для демо/отладки. Сейчас часть этого пришлось доделывать ближе к концу.

8.4. Дальнейшее развитие проекта

- Дальше я хочу улучшить поиск: добавить гибридный retrieval, переранжирование и простые метрики на наборе контрольных вопросов — чтобы было видно, что качество реально растёт, а не «кажется».
- В просмотре источников можно сделать удобнее переход к месту в документе: точнее подсветку, якоря, быстрые переключатели между источниками.
- Ещё полезно добавить статистику и мониторинг: какие вопросы задают чаще всего, где система отвечает «не знаю», какие документы чаще падают при обработке.
- И последнее — версии документов. Хочу хранить разные редакции и переиндексировать прозрачно, чтобы всегда было понятно, по какой версии документа дан ответ.

9. Заключение

В результате работы создан веб-сервис, который решает задачу быстрого доступа к информации из официальных документов лица. Продукт поддерживает ключевые сценарии: аккаунт пользователя, чат, поиск, просмотр источников, историю, экспорт, а также администрирование документов и пользователей.