

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»
(Университет ИТМО)**

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Языковые модели и искусственный интеллект**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

КУРСОВОЙ ПРОЕКТ

Тема: «Разработка модуля обратной связи для обучающей платформы»

Обучающийся: Плешнев Арсений Вадимович

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Суть проекта и процессы работы над ним.....	5
1.1 Суть проекта.....	5
1.2 Процессы работы над всем проектом.....	5
2 Личная задача.....	6
2.1 Суть проблемы, которая была поставлена передо мной.....	6
2.2 Как я решал поставленную задачу.....	6
2.3 Анализ своей работы.....	6
3 Взаимодействие с командой и оценка работы руководителя.....	7
3.1 Взаимодействие с командой.....	7
3.2 Взаимодействие с руководителем проекта.....	7
3.3 Оценка работы руководителя.....	7
ЗАКЛЮЧЕНИЕ.....	8
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	9
ПРИЛОЖЕНИЕ.....	10

ВВЕДЕНИЕ

В современных образовательных системах нередко отсутствует эффективная система обратной связи, что значительно снижает их функциональность. Во время использования образовательной платформы у студентов могут возникать вопросы, связанные с изучением теоретических материалов, выполнением лабораторных работ и использованием самой платформы. При отсутствии обратной связи вопросы решаются с большими задержками, что снижает качество образовательного процесса, негативно влияет на мотивацию студентов и мешает достижению поставленных учебных целей.

Для решения этой проблемы реализуется проект, целью которого является обеспечение эффективной коммуникации студента и образовательной платформы посредством модуля обратной связи, с помощью которого студенты смогут задавать вопросы по материалам, лабораторным работам, а также по использованию платформы в целом с возможностью указать тему сообщения, ввести само сообщение и прикрепить файл.

Для реализации модуля обратной связи для обучающей платформы необходимо (задачи проекта):

- 1) разработать функциональность для сохранения отправленных сообщений в базу данных с темой сообщения, текстом, прикрепленным файлом и временем отправки,
- 2) разработать пользовательский интерфейс с возможностью указать тему сообщения, написать текст и прикрепить файл,
- 3) обеспечить валидацию вводимых данных на backend и frontend,
- 4) настроить API для обработки запросов с использованием FastAPI и Swagger,
- 5) контейнеризовать приложение с использованием Docker,
- 6) создать файл docker-compose, который будет координировать работу всех компонентов приложения для дальнейшего размещения модуля обратной связи на сервере,

- 7) обеспечить доступ к приложению по постоянному IP-адресу,
- 8) выполнить тестирование частей frontend и backend,
- 9) организовать сопровождение и мониторинг приложения после его запуска, устранение возникающих ошибок.

1 Суть проекта и процессы работы над ним

1.1 Суть проекта

Суть проекта заключается в том, чтобы разработать и развернуть простое веб-приложение, позволяющее пользователям образовательной платформы задать вопрос по теме или решить техническую трудность. Чтобы пользователю не приходилось описывать свою проблему словами (что часто может быть довольно неудобно), необходимо предусмотреть опцию загрузки файлов (например, скриншота проблемы или задания).

Весь проект можно разделить на четыре основных шага:

- 1) Разработка Backend-части на языке Python с использованием библиотеки FastAPI;
- 2) Разработка Frontend-части (было решено сделать два варианта, с использованием фреймворков React и Astro);
- 3) Развёртывание модуля на сервере;
- 4) Тестирование и исправление багов, если таковые будут найдены.

В начале работы для лучшего понимания была составлена следующая схема:

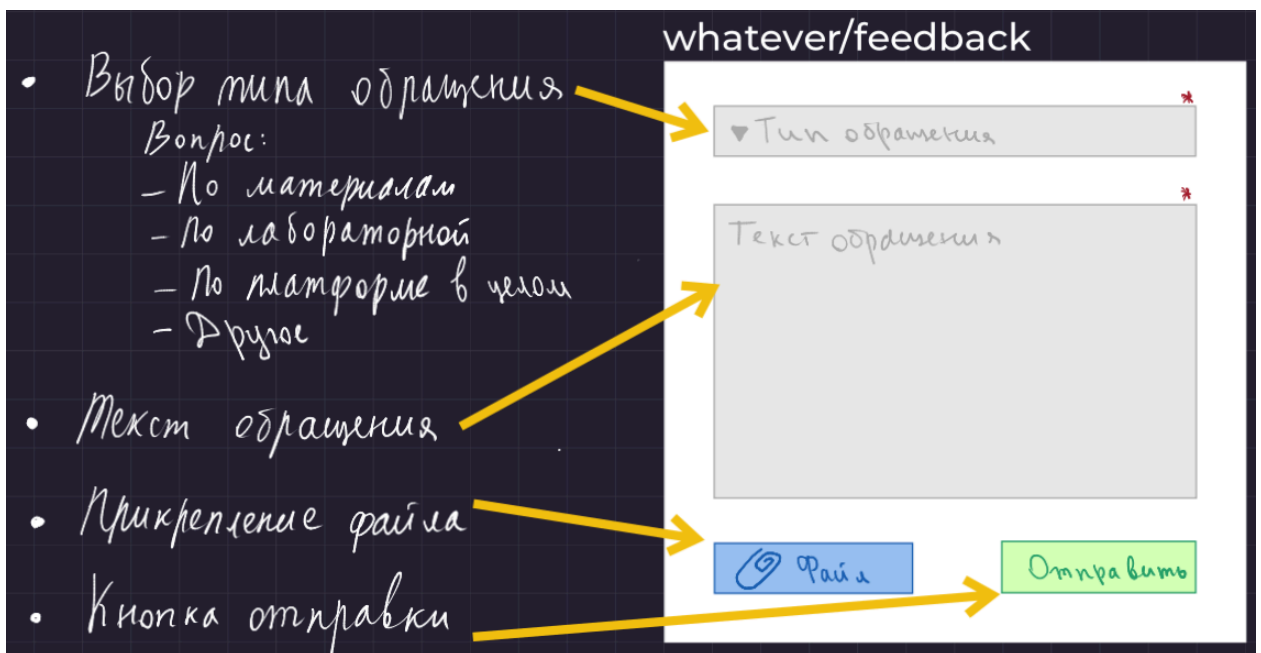


Рисунок 1 - эскиз будущего модуля

1.2 Процессы работы над всем проектом

Весь проект был разделен на четыре направления, обозначенные выше. Задачи по каждому из них были разделены между участниками команды следующим образом:

Backend	Плешнев Арсений
Frontend	Колесников Игорь и Нгуен Динь Нам
Тестирование	Юров Кирилл и Перова Максин
Devops	Сафонова Людмила

Таблица 1 - Распределение ролей в проекте

Слаженная работа и грамотное управление позволили обеспечить параллельную разработку (Backend + Frontend) и тестирование, также этому способствовали модульная структура и четкое разделение ролей. Работа проводилась с использованием системы контроля версий git и хостинг-сервиса GitHub, что позволяло получать оперативную обратную связь от руководителя и своевременно ознакамливаться с работой коллег.

В рамках задачи по разработке frontend был создан пользовательский веб-интерфейс, который включает поля для ввода темы и текста сообщения, возможность прикрепления файлов и кнопку для отправки данных на сервер.

Что касается backend, была разработана база данных для хранения сообщений, поступающих от frontend. Каждому запросу присваивается уникальный идентификатор, а также фиксируются категория запроса, текст сообщения, прикрепленные файлы и время отправки. Преподаватели и технические специалисты могут получить доступ к содержимому заявок (хотя в рамках проекта не было реализовано интерфейса, такая возможность есть при прямом обращении к базе данных), а студенты после отправки формы получают уведомление о статусе сообщения: успешно ли оно отправлено или возникла ошибка, требующая действий.

В backend-части также были реализованы API-эндпоинты, необходимые для сохранения данных в базе и возврата статусов обработки

сообщений. Документация по API была подготовлена с использованием Swagger, что значительно упростило дальнейшую работу с API и тестирование.

Для развертывания проекта в рабочей среде и обеспечения доступа пользователей был применен подход devops. В рамках этого подхода приложение было контейнеризировано с помощью Docker, и для всех его компонентов (backend, frontend и базы данных) были созданы отдельные docker-контейнеры. С помощью docker-compose была организована синхронная работа всех контейнеров, что упростило запуск приложения.

Следующим этапом модуль обратной связи был развернут на сервере в облаке YandexCloud на виртуальной машине с Ubuntu. Все компоненты приложения были собраны и запущены через docker-compose. Виртуальной машине был присвоен постоянный IP-адрес, обеспечивающий доступ пользователей к приложению. После запуска модуля была проведена проверка его доступности из Интернета.

В процессе разработки проводилось регулярное тестирование как backend, так и frontend с оперативным исправлением ошибок. После публикации приложения было выполнено финальное тестирование для подтверждения его работоспособности.

2 Личная задача

Как было указано в Таблице 1, я занимался Backend-частью приложения. Моя задача заключалась в том, чтобы создать API-эндпоинты (пути, по которым Frontend-часть будет обмениваться информацией с Backend-частью и базой данных), настроить валидацию полученной информации, в случае успеха - создать новую запись в базе данных и вернуть код успешного добавления записи, в случае нарушения заранее установленных стандартов сообщения - вернуть код ошибки.

2.1 Суть проблемы, которая была поставлена передо мной



Рисунок 2 - логика работы Backend-a

Как видно из Рисунка 2, основой Backend-части должна являться база данных. Также необходимо проверить условия, такие как проверка на пустые поля и проверка длины сообщения, что позволяет сделать стандартный синтаксис любого языка программирования.

2.2 Как я решал поставленную задачу

В основу разработки лёг Python с библиотекой FastAPI. Выбор был продиктован несколькими условиями: знание языка (на первом курсе моей программы изучается только Python), хотя бы относительное понимание работы библиотеки (у меня оно имелось).

В качестве СУБД был выбран PostgreSQL - абсолютно универсальный стандарт реляционной базы данных, к тому же с нативной библиотекой Python - Psycopg, значительно упрощающей работу.

Для лучшего понимания на этапе разработки базы данных схема (изображенная на Рисунке 1) была доработана, отражены соответствия полей формы полям в базе данных.

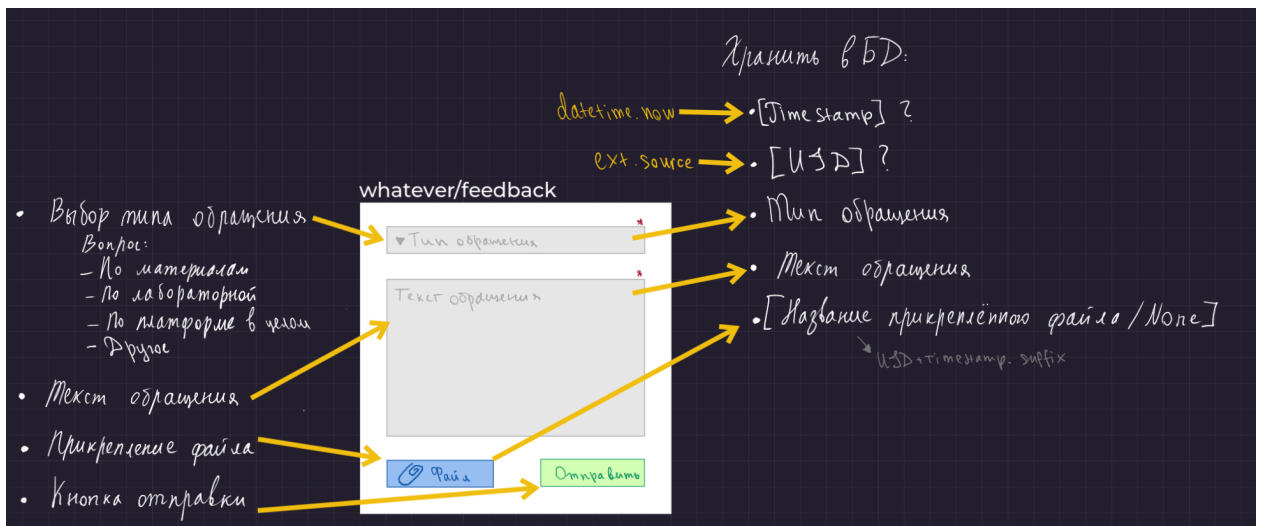


Рисунок 3 - Соответствие полей формы полям в базе данных

Далее были созданы ER и IDEF1X диаграммы:

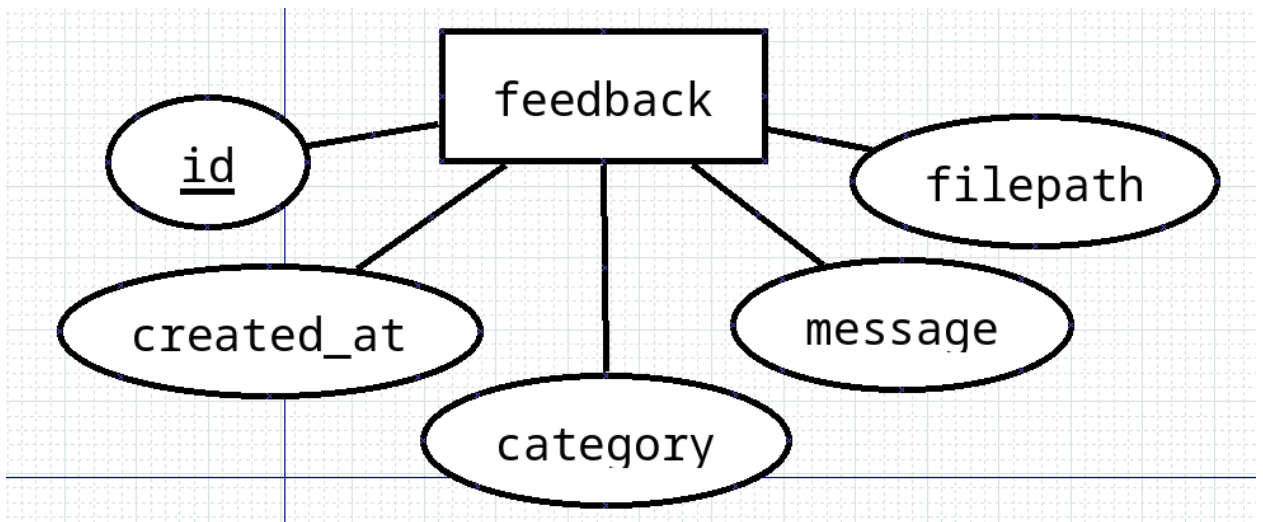


Рисунок 4 - ER-диаграмма



feedback	
id 	SERIAL
category	VARCHAR(255) NN
message	TEXT NN
file_path	VARCHAR(255)
created_at 	TIMESTAMP

Рисунок 5 - IDEF1X диаграмма

Далее было необходимо запрограммировать создание этой таблицы через само приложение при первом запуске в случае, если она еще не была создана:

```

1  # Функция для создания таблицы feedback
2  def create_feedback_table():
3      conn = get_db_connection()
4      with conn.cursor() as cur:
5          cur.execute("""
6              CREATE TABLE IF NOT EXISTS feedback (
7                  id SERIAL PRIMARY KEY,
8                  category VARCHAR(255) NOT NULL,
9                  message TEXT NOT NULL,
10                 file_path VARCHAR(255),
11                 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
12             )
13             """)
14         conn.commit()

```

Рисунок 6 - Функция создания таблицы feedback

Теперь у приложения есть, куда записывать данные, но оно еще не имеет самого эндпоинта для их получения. Было решено сделать один эндпоинт для упрощения архитектуры:

```
1 @router.post("/feedback")
2 async def submit_and_upload_feedback(
3     category: Annotated[str, Form(description="Категория обратной связи")],
4     message: Annotated[str, Form(description="Сообщение обратной связи")],
5     file: Annotated[Union[UploadFile, None], File(description="Файл для загрузки")] = None
6 ):
7     # Проверка на пустые поля
8     if not category or not message:
9         raise HTTPException(
10             status_code=status.HTTP_400_BAD_REQUEST,
11             detail="Категория и сообщение обязательны для заполнения."
12         )
13
14     # Проверка длины сообщения
15     if len(message) > 280:
16         raise HTTPException(
17             status_code=status.HTTP_400_BAD_REQUEST,
18             detail="Сообщение не должно превышать 280 символов."
19         )
20
21     # Сохраняем локально загруженный файл
22     ...
23     # Записываем в базу данных
24     ...
25     return {"detail": "Успешно отправлено!"}
```

Рисунок 7 - Endpoint, валидация данных

В функции эндпоинта проводится простая валидация: не допускаются пустые сообщения и сообщения с количеством символов более 280. После успешного прохождения валидации сообщение необходимо записать в базу данных, реализовано это так же при помощи библиотеки Psycopg:

```

1  # Записываем в базу данных
2  try:
3      conn = get_db_connection()
4      with conn.cursor() as cur:
5          cur.execute("BEGIN;")
6          cur.execute("""
7              INSERT INTO feedback (category, message, file_path)
8              VALUES (%s, %s, %s)
9              """, (category, message, file_path))
10         conn.commit()
11 except Exception as e:
12     conn.rollback()
13     raise HTTPException(
14         status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
15         detail=f"Ошибка при сохранении данных в базу: {str(e)}"
16     )

```

Рисунок 8 - Запись сообщения в базу данных

Представленные функции исчерпывающе описывают разработанный мною функционал. Код, не представленный на рисунках, в основном был необходим для удобной Swagger-документации (чем занималась Людмила).

2.3 Анализ своей работы

При работе над проектом у меня получилось углубить свои знания по таким инструментам и технологиям, как ООП, библиотеки FastAPI и Psycopg. Практически с нуля удалось изучить систему управления базами данных PostgreSQL, на что ушло достаточно много времени.

Также стоит отметить, что мне очень пригодились знания, полученные на парах по предмету “Базы данных”, я активно их применял при разработке базы данных нашего проекта. Также очень пригодились знания ОС GNU/Linux, ведь приложение было рассчитано для запуска на Ubuntu - дистрибутиве GNU/Linux.

Благодаря тайм-менеджменту и регулярным созвонам, мне удавалось выполнять задачи качественно и в установленный руководителем срок.

В ходе выполнения задач я понял, что мне не хватает знаний в области объектно-ориентированного программирования, из-за этого многие функции использованных библиотек требовали более тщательного изучения и были менее интуитивно понятны. К счастью, на втором курсе у нас будет отдельный курс “Объектно-ориентированное программирование”, где я погружусь в эту парадигму программирования.

В результате выполнения проекта я не только приобрёл технические навыки, но и научился работать в команде, что незаменимо при реальной разработке и трудоустройстве.

3 Взаимодействие с командой и оценка работы руководителя

3.1 Взаимодействие с командой

Работа в команде проходила слаженно, с первого дня проекта у нас был организован Телеграмм - чат, где мы обсуждали текущие задачи и делились опытом.

Коммуникация с одним из членом команды была незначительно затруднена языковым барьером, но мы успешно его преодолевали, старались помогать и объяснять нашему иностранному коллеге всё максимально доходчиво.

Я благодарен своей команде за слаженный труд и комфортную работу, которые и позволили выполнить все цели проекта в установленный срок, а также достойно выступить и защитить проект перед экспертами.

3.2 Взаимодействие с руководителем проекта

Вадим Жуков проявил себя как ответственный и чуткий руководитель. Он организовал еженедельные звонки в удобное всем время через удобный канал связи – Телеграм. На этих встречах мы делились своими успехами, задавали вопросы, если что-то не получалось, и получали исчерпывающие ответы.

3.3 Оценка работы руководителя

Получая подробные объяснения и чёткие наставления, я чувствовал, что Вадим уже много лет занимается веб-разработкой, а главное - очень любит эту работу, поэтому разбирается в ней настолько хорошо.

Учитывая во внимание совокупность вышеперечисленных факторов, я не могу оценить работу Вадима Жукова ниже чем на 10/10.

ЗАКЛЮЧЕНИЕ

Проект по разработке модуля обратной связи для образовательной платформы успешно завершен, и его главная цель (создание эффективного инструмента для общения между студентами и преподавателями) достигнута. Теперь студенты имеют возможность задавать вопросы по учебным материалам, лабораторным работам и общему опыту использованию платформы.

В процессе работы над проектом были выполнены все запланированные задачи: создана база данных, разработан пользовательский интерфейс, реализована валидация вводимых данных, настроен API с использованием FastAPI и Swagger, проведено тестирование, а также осуществлена контейнеризация приложения. Модуль был развернут и запущен на сервере в облаке YandexCloud, обеспечив доступ через постоянный IP-адрес.

Мой вклад в проект заключался в разработке базы данных и бэкенд-части приложения, с чем я справился в полном объеме.

Модуль обратной связи был успешно внедрен и запущен, и теперь он станет важным инструментом для повышения качества учебного процесса на платформе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Документация [FastAPI](#)
- 2) Статьи на портале [geeksforgeeks](#)
- 3) Видеоуроки по FastAPI на портале [freecodecamp](#)
- 4) Документация [PostgreSQL](#)
- 5) Материалы по PostgreSQL на портале [freecodecamp](#)

ПРИЛОЖЕНИЕ

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. Название проекта:

Разработка модуля обратной связи для обучающей платформы

2. Цель (назначение):

Разработать модуль обратной связи для обучающей платформы

3. Сроки выполнения:

Начало - 2024-11-01, Конец - 2024-12-20

4. Исполнитель проекта (руководитель проекта):

Жуков Вадим Витальевич

5. Термины и сокращения:

API - Application Programming Interface (интерфейс прикладного программирования)

JSON - JavaScript Object Notation Swagger/OpenAPI - Инструменты для документирования и описания API

СУБД - Система управления базами данных

QA - Quality Assurance (обеспечение качества)

YAML - Yet Another Markup Language (формат разметки)

Бэклог - Очередь задач Деплой - Процесс переноса кода из среды разработки на рабочий сервер

Прод/продакшн - Завершающий этап разработки после сборки, тестирования и развёртывания программы на рабочем сервере

Технологический стек - Набор инструментов, необходимых для разработки продукта

Фреймворк - Готовая структура и набор инструментов, на основе которых ведётся разработка на разных языках программирования

6. Технические требования (технические, дидактические, программные, эргономические, экологические и др.)

Техническое требование	Язык разработки	СУБД	Потребители
Эндпоинт для отправки данных формы (Backend)	Python, FastAPI	Нет	Разработчики
Чек-лист и тест-кейсы на разработанный функционал (QA)	Русский	Нет	Разработчики
Руководство пользователя по использованию приложения (Общее/QA)	Русский	Нет	Все пользователи
Поля ввода контактных данных и сообщения (Frontend)	JavaScript/TypeScript	Нет	Все пользователи
Обработка и сохранение данных формы (Backend)	Python, FastAPI	Локально/PostgreSQL	Разработчики
Контейнеризация приложения с помощью Docker (DevOps)	YAML, docker, docker-compose	Нет	Разработчики
Деплой приложения в облако (DevOps)	YAML, docker, docker-compose	Нет	Разработчики

Валидация данных формы перед отправкой на сервер (Frontend)	JavaScript/TypeScript	Нет	Все пользователи
Swagger/OpenAPI документация для API (Backend)	Python, FastAPI	Нет	Разработчики

Функциональные и нефункциональные требования*

7. Содержание работы:

Таблица 1

Этапы работы
Анализ требований
Настройка среды разработки
Разработка
Тестирование
Внедрение и поддержка

Таблица 2

Название задачи	Описание	Technical Specification	Этап	Ответственный
Верстка формы обратной связи	Сверстать форму обратной связи. Можно использовать готовые компоненты Mantine UI или	Разработка модуля обратной связи для обучающей платформы	Разработка	Колесников Игорь Евгеньевич

	разработать свои, используя HTML, CSS, JS			
Изучить FastAPI и развернуть стартовое приложение	Изучить фреймворк FastAPI, ознакомиться с документацией и создать стартовое приложение https://fastapi.tiangolo.com/	Разработка модуля обратной связи для обучающей платформы	Настройка среды разработки	Плешнев Арсений Вадимович
Изучить работу с формами (FormData)	Изучить документацию по работе с HTML-формами и объектом FormData: создание, заполнение, отправка данных на сервер, обработка вложений и ключевых методов https://doka.guide/js/deal-with-forms/	Разработка модуля обратной связи для обучающей платформы	Настройка среды разработки	Нгуен Динь Нам

Написать чек-лист и тест-кейсы	Составить чек-лист по разработанному функционалу и по нему написать 2-3 тест-кейса	Разработка модуля обратной связи для обучающей платформы	Тестирование	Юров Кирилл Игоревич
Настроить валидацию данных на клиенте	Настроить валидацию данных на клиенте, проверку на пустые поля ввода и т.д. Можно использовать готовые методы обработки формы https://mantine.dev/form/use-form/	Разработка модуля обратной связи для обучающей платформы	Разработка	Колесников Игорь Евгеньевич
Настроить генерацию документации и API (Swagger/OpenAPI)	Подключить Swagger по инструкции. Доступ к Swagger должен осуществляться по url /api/docs.	Разработка модуля обратной связи для обучающей платформы	Настройка среды разработки	Сафонова Людмила Марковна
Настроить	Написать	Разработка	Разработка	Сафонова

контейнеризацию приложения (бэкенд)	Dockerfile для сборки образа бэкенда, а также настроить docker-compose для управления контейнером и взаимодействия с другими сервисами	модуля обратной связи для обучающей платформы		Людмила Марковна
Настроить контейнеризацию приложения (фронтенд)	Создать докерфайл для деплоя статического приложения https://docs.as-tro.build/en/recipes/docker/#static	Разработка модуля обратной связи для обучающей платформы	Разработка	Колесников Игорь Евгеньевич
Настроить отправку данных на бэкенд через API	Необходимо отправлять данные формы по соответствующему эндпоинту на сервере. Данные формы должны передавать в теле запроса или через FormData в	Разработка модуля обратной связи для обучающей платформы	Разработка	Колесников Игорь Евгеньевич

	случае отправки файлов.			
Настроить сохранение данных в базе данных или локально	Необходимо сохранять данные отправленной формы локально или в базе данных PostgreSQL. В случае с PostgreSQL необходимо создать таблицу для сохранения данных формы.	Разработка модуля обратной связи для обучающей платформы	Разработка	Плешнев Арсений Вадимович
Отправить файл (картинку) на бэкенд	Необходимо соединить фронтенд с бэкендом и отправить файл на сервер. Получить положительный ответ от сервера.	Разработка модуля обратной связи для обучающей платформы	Разработка	Нгуен Динь Нам
Перенести код из среды разработки	Перенести код на облачный	Разработка модуля обратной	Внедрение и поддержка	Сафонова Людмила Марковна

на сервер	сервер и запустить приложение с помощью docker-compose	связи для обучающей платформы		
Подготовить руководство пользователя	Составить инструкцию о том, как пользоваться приложением	Разработка модуля обратной связи для обучающей платформы	Внедрение и поддержка	Юров Кирилл Игоревич
Протестировать интеграцию фронтенда и бэкенда	Проверить корректность работы всего приложения, включая отправку данных с фронтенда, обработку их на бэкенде и отображение результата на клиенте	Разработка модуля обратной связи для обучающей платформы	Тестирование	Юров Кирилл Игоревич
Сверстать поле для загрузки файлов в форме	Сверстать поле ввода для загрузки файлов. Например, https://ui.mantine.dev/category/dropzones/ или https://mantine.dev/core/file	Разработка модуля обратной связи для обучающей платформы	Разработка	Нгуен Динь Нам

	<input type="text"/> Убедиться, что файл можно загрузить. Вывести результат в консоль браузера			
Создать новый проект Astro	Изучить фреймворк Astro, ознакомиться с документацией и создать стартовое приложение https://docs.astro.build/ru/install-and-setup/	Разработка модуля обратной связи для обучающей платформы	Настройка среды разработки	Колесников Игорь Евгеньевич
Создать поля для ввода текста и контактных данных	Создать поля для ввода текста и контактных данных пользователя. Поле ввода Mantine UI	Разработка модуля обратной связи для обучающей платформы	Разработка	Колесников Игорь Евгеньевич
Создать эндпоинты для отправки	Написать один эндпоинт с	Разработка модуля обратной	Разработка	Плешнев Арсений Вадимович

и обработки данных формы (FastAPI)	методом POST, который будет принимать данные формы (как тело запроса) с клиента	связи для обучающей платформы		
------------------------------------	---	-------------------------------	--	--

8. Основные результаты работы:

Рабочее веб-приложение с формой обратной связи. Подготовлена документация к разработанному функционалу, включая API, а также руководство пользователя. Составлены тест-кейсы для проверки ключевых функций. Приложение упаковано в Docker-контейнеры, настроено управление через Docker Compose и выполнен деплой в облачную среду.