

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

Факультет Прикладной информатики

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

**Образовательная программа Языковые модели и искусственный
интеллект**

КУРСОВОЙ ПРОЕКТ

Тема: «Разработка модуля обратной связи для обучающей платформы»

Обучающийся: Сафонова Людмила Марковна, К3161

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.....	8
2.1 Суть проблемы, которая была поставлена передо мной.....	8
2.2 Как я решала поставленную задачу.....	8
2.3 Анализ своей работы.....	12
3 Взаимодействие с командой и оценка работы руководителя.....	14
3.1 Взаимодействие с командой.....	14
3.2 Взаимодействие с руководителем проекта.....	14
3.3 Оценка работы руководителя.....	14
ЗАКЛЮЧЕНИЕ.....	16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	17
ПРИЛОЖЕНИЕ.....	18

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:

- 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 разработке модуля обратной связи для образовательной платформы, который станет удобным инструментом для обеспечения быстрого и структурированного взаимодействия между студентами, преподавателями и техническими сотрудниками. Модуль предназначен для студентов, которые при помощи этого приложения смогут задавать вопросы по теоретическим материалам, лабораторным работам, а также по работе образовательной платформы.

Проект предусматривает реализацию формы обратной связи, которая позволит пользователям указывать тему сообщения для упрощения классификации запросов, вводить текст сообщения, в деталях описывающий проблему или вопрос, и прикреплять файлы, подтверждающие или иллюстрирующие запрос.

По технической части проект включает в себя:

- 1) разработку backend-части на базе платформы FastAPI для обработки запросов, взаимодействия с базой данных и присвоения статуса сообщениям,
- 2) реализацию frontend-части для создания пользовательского интерфейса с помощью фреймворков React и Astro,
- 3) развертывание модуля на сервере в облаке, чтобы он был доступен всем пользователям,
- 4) тестирование.

1.2 Процессы работы над всем проектом

Работа над проектом была разделена на следующие роли участников команды: backend, frontend, devops и тестирование. За backend отвечал Плешнев Арсений, frontend - Колесников Игорь и Нгуен Динь Нам, тестирование - Юров Кирилл и Перова Максин, devops - Сафонова Людмила.

Модульная структура проекта позволила вести работу в параллельном режиме. Одновременно выполнялись работы над backend, frontend и devops частями.

В рамках задачи по frontend был разработан пользовательский web-интерфейс с полями для ввода темы сообщения, текста сообщения, прикрепления файлов и кнопкой отправки сообщения на сервер.

В рамках задачи по backend была выполнена разработка структуры базы данных для сохранения сообщений, полученных от модуля frontend. Каждой отправленной заявке в базе данных присваивается уникальный идентификатор и сохраняется информация: категория запроса, текст сообщения, прикрепленный файл, время отправки сообщения. В дальнейшем преподаватели и технические сотрудники получают доступ к содержанию заявки. Также после отправки формы студент получает информацию о статусе сообщения: сообщение принято или возникла ошибка.

Второй задачей по backend была реализация и настройка API-эндпоинтов, необходимых для сохранения информации в базе данных и возврата статусов обработки отправленных сообщений. Документация по API была создана с помощью Swagger. Этот инструмент позволил упростить последующую работу с API и тестирование проекта.

Для запуска проекта в рабочую эксплуатацию и обеспечения доступа пользователей был использован подход devops.

В рамках этого подхода была выполнена контейнеризация приложения с использованием технологии Docker. Были созданы отдельные docker-контейнеры для всех частей проекта: backend-модуля, frontend-модуля и базы данных, после чего с помощью инструмента docker-compose была настроена работа всех контейнеров одновременно. Это позволило упростить запуск приложения, протестировать взаимодействие между компонентами и гарантировать их стабильную совместную работу.

Следующим этапом модуль обратной связи был развернут на сервере в облаке YandexCloud на виртуальной машине на базе Ubuntu. Все компоненты

приложения были собраны через `docker-compose` и запущены. Далее виртуальной машине был присвоен постоянный IP-адрес, через который осуществляется доступ пользователей к приложению. После запуска модуля обратной связи была проведена проверка доступности приложения из Интернета.

В процессе разработки велось постоянное тестирование частей `backend` и `frontend` с оперативным исправлением ошибок. После публикации приложения было проведено финальное тестирование, чтобы убедиться в работоспособности системы.

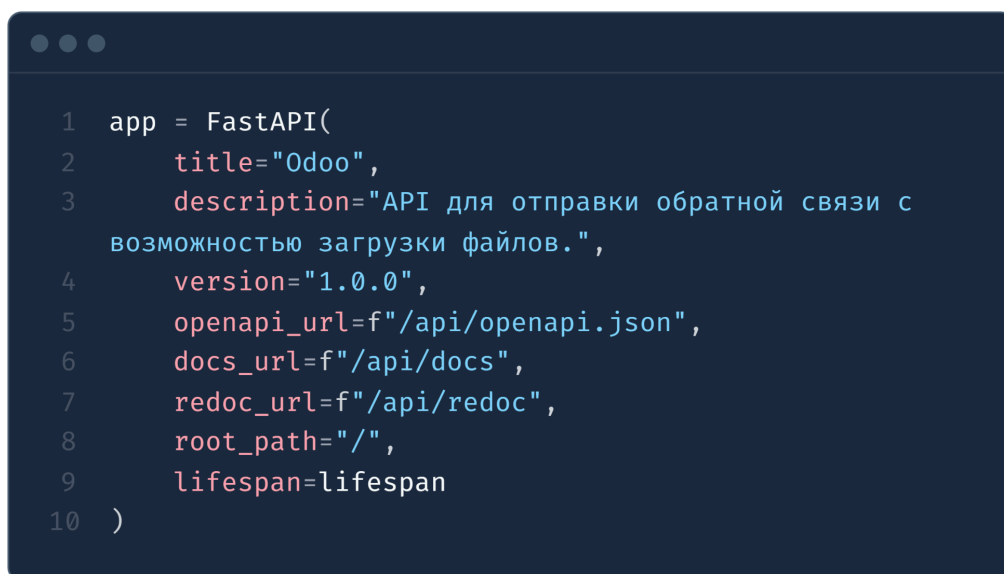
2 Личная задача

2.1 Суть проблемы, которая была поставлена передо мной

Проблемы, поставленные передо мной, включали работу со следующими областями: backend и devops. В рамках работы над backend мне нужно было настроить автоматическое описание API с использованием инструмента Swagger при запуске кода Python FastAPI, а также обеспечить понятный и грамотный вывод данных при работе с этой технологией. Что касается работы над devops - она включала в себя контейнеризацию всех частей проекта, размещение и запуск приложения на сервере, а также проверку корректности работы модуля обратной связи после запуска.

2.2 Как я решала поставленную задачу

В ходе выполнения задачи, связанной с backend, я узнала основы работы с технологией FastAPI [1], организовала вывод документации API с помощью Swagger (код конфигурации и вывод документации из Swagger в браузере представлены на рисунках 1 и 2) и ознакомилась с настройками интерфейса этого инструмента [2]. Проверяла при помощи Swagger, работоспособность всех эндпоинтов, изменила визуальную тему и настроила валидацию данных (проверку соответствия входных данных указанным типам).

A screenshot of a code editor with a dark background. It shows Python code for initializing a FastAPI application. The code is numbered from 1 to 10. It includes parameters for title, description, version, openapi_url, docs_url, redoc_url, root_path, and lifespan.

```
1 app = FastAPI(  
2     title="Odo",  
3     description="API для отправки обратной связи с  
возможностью загрузки файлов.",  
4     version="1.0.0",  
5     openapi_url=f"/api/openapi.json",  
6     docs_url=f"/api/docs",  
7     redoc_url=f"/api/redoc",  
8     root_path="/",  
9     lifespan=lifespan  
10 )
```

Рисунок 1 – Код для подключения Swagger в FastAPI

Feedback

POST /api/feedback Отправить обратную связь

Эндпоинт для отправки обратной связи с возможностью загрузки файла. Категория и сообщение обязательны для заполнения. Сообщение не должно превышать 280 символов.

Parameters
Try it out

No parameters

Request body required
multipart/form-data

category * required

Категория обратной связи

string

message * required

Сообщение обратной связи

string

file

Файл для загрузки

string (binary)

Responses

Code	Description	Links
200	<div>Успешно отправлено!</div> <div>Media type</div> <div>application/json</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <pre>{ "detail": "Успешно отправлено!" }</pre>	No links
400	<div>Некорректные данные</div> <div>Media type</div> <div>application/json</div>	No links

Рисунок 2 – Представление документации в Swagger

Задачи, связанные с devops, были следующие: контейнеризация и размещение приложения на сервер в облаке.

В рамках первой задачи я создала контейнеры для каждой из частей проекта “Модуль обратной связи”: backend, frontend, база данных. Я ознакомилась с технологией Docker [3], ее особенностям при применении с FastAPI [4] и создала Dockerfile (примеры файлов представлены на рисунках 3 и 4) для каждой из областей проекта, затем собрала образы контейнеров, а после и сами экземпляры контейнеров.

```
FROM python:3.13

WORKDIR /app

COPY ./backend/requirements.txt .

RUN pip install --no-cache-dir --upgrade -r requirements.txt

COPY ./backend /app

EXPOSE 8000
```

Рисунок 3 – Backend.dockerfile

```
FROM node:lts AS build
WORKDIR /app

COPY ./frontend/package*.json ./
RUN npm install --verbose

COPY ./frontend .
RUN npm run build

FROM nginx:alpine AS runtime
COPY ./docker/nginx/nginx.conf /etc/nginx/nginx.conf
COPY --from=build /app/dist /usr/share/nginx/html
CMD ["nginx", "-g", "daemon off;"]
```

Рисунок 4 – Frontend.dockerfile

Когда все контейнеры были собраны, я создала файл docker-compose [5] (код представлен на рисунках 5, 6), в котором были прописаны связи между частями проекта и организован запуск каждого из контейнеров.

```

1   version: '3.9'
2   services:
3     frontend:
4       container_name: frontend
5       build:
6         context: .
7         dockerfile: docker/frontend/frontend.dockerfile
8       restart: unless-stopped
9       ports:
10        - 80:80
11        - 443:443
12       depends_on:
13        - backend
14
15     backend:
16       container_name: backend
17       build:
18         context: .
19         dockerfile: docker/backend/backend.dockerfile
20       env_file: ".env"
21       environment:
22        - PYTHONUNBUFFERED=1
23        - ENV=production
24       restart: unless-stopped
25       command: uvicorn main:app --host 0.0.0.0 --port 8000
26       ports:
27        - 8000:8000
28       depends_on:
29        - db

```

Рисунок 5 – Docker-compose (часть 1)

```

30    db:
31      image: postgres:latest
32      ports:
33        - 5432:5432
34      container_name: db
35      env_file: ".env"
36      environment:
37        POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
38        POSTGRES_USER: ${POSTGRES_USER}
39        POSTGRES_DB: ${POSTGRES_DATABASE}
40
41    volumes:
42      pgdata:
43        driver: local

```

Рисунок 6 – Docker-compose (часть 2)

После создания docker-compose, было проверено взаимодействие между контейнерами (запросы от API к базе данных). Таким образом с помощью docker-compose была создана заготовка для запуска всего приложения.

Задача с размещением приложения на сервер была решена в несколько этапов. Сначала я подготовила виртуальную машину с дистрибутивом Ubuntu

к деплою приложения, а именно: установила docker, docker-compose и проверила наличие системы управления версиями git [6]. Далее весь код проекта был загружен в репозиторий на Github. Репозиторий был клонирован командой `git clone` на виртуальную машину в облаке. Командой `docker-compose up -d` были собраны и запущены все контейнеры, т.е. само приложение. Серверу был назначен постоянный IP-адрес, по которому и осуществляется доступ к приложению (на момент написания отчета по курсовому проекту адрес больше не доступен). Схема выгрузки приложения наглядно представлена на рисунке 7.

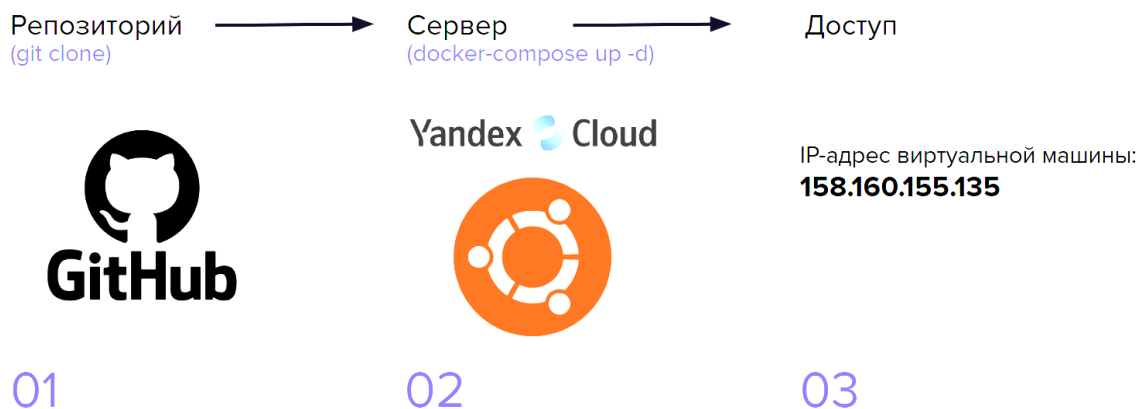


Рисунок 7 – Схема выгрузки приложения

2.3 Анализ своей работы

При работе над проектом у меня получилось разобраться с инструментом подготовки документации Swagger, с технологией контейнеризации Docker и процессом публикации приложений в облачной инфраструктуре YandexCloud.

Основные трудности вызвала моя неопытность в использовании Docker и FastAPI. Мне потребовалось много времени на изучение документации и дополнительных материалов. При запуске приложения на сервере база данных не была связана с другим частями проекта, так как я неправильно настроила порт для подключения к базе данных. Из-за этого пришлось пересобирать контейнеры, что замедлило процесс разработки.

Мне удавалось работать планомерно, так как я привыкла к дисциплинированной работе, хотя иногда у меня возникали сложности из-за освоения большого объема информации за короткий период, а также необходимости совмещения выполнения проекта с другими учебными задачами. Эти проблемы я смогла решить благодаря тайм-менеджменту, четкой постановке приоритетов и регулярной коммуникации с командой и руководителем проекта.

Несмотря на недостаточный опыт, я научилась работать с инструментом Swagger для создания документации API, более уверенно использовать Docker и docker-compose для контейнеризации и публикации приложения.

В ходе выполнения своих задач я поняла, как происходит процесс размещения приложения на сервер и с какими проблемами он может быть сопряжен.

В результате выполнения проекта я не только приобрела технические навыки, но и научилась работать в команде, выстраивать деловую коммуникацию и быстро находить решения возникающих проблем. Этот опыт помог мне лучше понять, как организован процесс разработки приложений, и осознать важность четкого распределения обязанностей в команде.

3 Взаимодействие с командой и оценка работы руководителя

3.1 Взаимодействие с командой

Взаимодействие с командой происходило посредством чатов и совещаний. На совещаниях обсуждалась последовательность выполнения задач участников проекта, что и в каком виде каждый ожидает получить от других для выполнения своей работы. Те участники, с кем я должна была выполнять смежные задачи, выполняли все в срок, легко выходили на контакт, подсказывали, если видели в моей части ошибки или огрехи, которые они могли бы помочь исправить. В общении была взаимопомощь, поддержка и уважение.

3.2 Взаимодействие с руководителем проекта

Взаимодействие с руководителем проекта, Жуковым Вадимом, проходило эффективно и строилось на взаимном уважении. Вадим проводил совещания по промежуточным результатам каждого из членов команды, на которых он вносил правки и предложения по улучшению проделанной работы. Если у меня возникали проблемы по задаче, я могла обратиться к нему за помощью и получала подробное объяснение по предмету своего вопроса. Вадим всегда отвечал быстро и четко, давал большое количество источников и материалов, которые могут пригодиться при работе. Нам не нужно было гадать над тем, что от каждого требовалось, потому что все было грамотно и подробно расписано в доступной для понимания форме.

3.3 Оценка работы руководителя

Руководитель проекта, на мой взгляд, успешно справился со своей задачей. Вадим Жуков с умом распределял задачи и время, которое может потребоваться на их выполнение каждому из участников проекта, ориентируясь на его уровень знаний, отслеживал сроки и качество работы команды. Не было такой ситуации, когда бы я не знала, что должна делать в данный момент.

Положительными чертами своего руководителя могу назвать: компетентность, умение находить общий язык с командой, ответственность,

честность. Вадим не пустил проект на самотек, всегда присутствовал и держал руку на пульсе. Он следил за фактом выполнения задач и быстро находил решения возникающим проблемам.

Из отрицательных моментов могу только отметить, что мой руководитель проекта иногда был несколько мягче, чем того требует его должность.

Своему руководителю проекта я могу дать исключительно положительную оценку. Благодаря ему я разобралась с новыми для себя областями знаний: работа с API и FastAPI, а также технологией Docker и инструментом docker-compose. Приобретенные знания помогут мне в дальнейшем образовании и могут стать фундаментом моей профессиональной деятельности в будущем. Я благодарна Вадиму Жукову за его проделанную работу.

ЗАКЛЮЧЕНИЕ

Проект «Разработка модуля обратной связи для образовательной платформы» был успешно завершен, а его цель – создание инструмента для эффективной коммуникации, с помощью которого студенты смогут задавать вопросы по учебным материалам, лабораторным работам, а также по использованию платформы в целом – достигнута.

Во время работы над проектом были выполнены все поставленные задачи: разработана база данных, создан пользовательский интерфейс, обеспечена валидация вводимых данных, настроен API на FastAPI с помощью Swagger, проведено тестирование, была произведена контейнеризация приложения, его развертка и запуск на сервере в облаке YandexCloud с дальнейшим сопровождением и обеспечением доступа по постоянному IP-адресу.

Мой вклад в достижение цели включал настройку Swagger, организацию devops-процесса по контейнеризации всех компонентов проекта и публикации приложения на виртуальной машине в облаке, а также устранение ошибок, возникающих после запуска модуля обратной связи. Моя работа была сосредоточена на задачах backend и devops, что позволило обеспечить функционирование приложения в общем доступе.

Цель проекта была достигнута благодаря четкому распределению обязанностей между членами команды, эффективной коммуникации, помощи со стороны руководителя и слаженной работе, несмотря на возникающие проблемы с задачами и ограничениями по времени.

Модуль обратной связи был успешно создан и запущен. Этот инструмент, улучшающий качество учебного процесса студентов, станет важной частью образовательной платформы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 FastAPI [Электронный ресурс]. – URL: <https://fastapi.tiangolo.com> (дата обращения: 15.11.2024).
- 2 Documentation: Swagger UI Configuration [Электронный ресурс]. – URL: <https://swagger.io/docs/open-source-tools/swagger-ui/usage/configuration/> (дата обращения: 16.11.2024).
- 3 Docker Desktop [Электронный ресурс]. – URL: <https://docs.docker.com/desktop/> (дата обращения: 28.11.2024).
- 4 FastAPI in Containers - Docker [Электронный ресурс]. – URL: <https://fastapi.tiangolo.com/deployment/docker/> (дата обращения: 28.11.2024).
- 5 Docker Compose [Электронный ресурс]. – URL: <https://docs.docker.com/compose/> (дата обращения: 30.11.2024).
- 6 Git - git Documentation [Электронный ресурс]. – URL: <https://git-scm.com/docs/git> (дата обращения: 6.12.2024).

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.