## 2. Реализация отдельных компонентов проекта

### 2.1. Разработка локального сервера

Разработкой локального сервера занимался я. Как было написано ранее, для разработки была выбрана библиотека Flask. Она была выбрана из-за своей простоты и удобства использования. Она полностью подходила нам по своему функционалу. В ходе изучения этой библиотеки возникла проблема с версиями, в файле со всеми зависимостями в проекту requirements.txt была указана не самая новая версия этой библиотеки. Было решено изменить requirements.txt и обновить до новейшей версии, в последствии версия была обновлена до модифицированной, поддерживающей асинхронность.

Была получена ссылка на рабочий репозиторий, в котором была прописана файловая структура для сервера. Было принято решение не изменять структуру проекта. В файловой структуре была проблема с настройкой зависимостей, поскольку файл server.py находится в корне, а работа с сервером происходит на уровень ниже. Было изучено взаимодействие с файлом \_\_init\_\_.py, который является специальным файлом в Python, используемым для обозначения директорий как пакетов. Этот файл выполняет несколько функций: инициализирует пакет, управляет взаимодействием с ним, а также позволяет создавать пространство имён для модулей. В ходе работы с \_\_init\_\_.py были устранены все возникавшие проблемы, и была начата работа с сервером.

Была создана html страница для главной страницы сайта, чтобы было видно, что сервер работает. В качестве html страницы был взят документ из прошлого независимого проекта.

Она помещена в templates/  
Для нее был написал код для перехода на страницу /summarize. Переход осуществляется в виде кнопки со скриптом

Была инициализирована страница /summarize, на которую и должен отправляться запрос.  
 Были получены все функции от нашей команды.

Для понимания работы с сервером необходимо было изучить модуль “youtube\_transcript\_api”, который в свою очередь и принимает субтитры с видео.

Возникла проблема с этим модулем: по умолчанию этот модуль ищет в видео английские субтитры, а если субтитров нет вообще, то выдается ошибка. Было принято решение установить стандартный приоритет языков, по которым ищутся субтитры и была написана функция, которая показывает отсутствие субтитров. Таким образом, я немного залез в выполнение задачи Дмитрия.

Все функции были импортированы и началась работа с их взаимодействием.

Требовалось понять логику взаимодействия сервера с запросами. Как должен работать сервер: сервер получает в качестве запроса ссылку на видео -> сервер выводит json-ответ с транскрипцией видео.

Для того, чтобы корректно вызвать выполнение функции получения субтитров требуется корректное получение ID видео. Для этого существует функция, которая получает строку и получает из нее id. В качестве нее была написана пустая функция, которая в последствии была заменена на корректную Дмитрием в рамках своей задачи. Затем выполняется функция, которая по ID получает субтитры. При отладке этой функции возникла проблема: бесконечное ожидание ответа. Вскоре была предпринята попытка запустить на сервере VPN. В итоге получение ответа стало возможным. Таким образом, возникла глобальная проблема: VPN на сервере. Было принято решение, что на сервере должен стоять VPN, поскольку иначе дальнейшая разработка не представлялась возможной.

После получения функцией текста требуется отправить ответ сервера на запрос.  
 Для отладки сервера была написана программа, которая создает запросы. Она имеет вид: (введите ссылку: … - субтитры)

Таким образом начался процесс отладки. Были протестированы несколько типов видео.  
– короткие,

– длинные,

– очень длинные,

– без субтитров,

– с большим количеством субтитров,

– русские,

– иностранные.

Все тесты пройдены, везде при тестировании был получен корректный ответ.

Все шаги выполнения сопровождались коммитами, которые требовались от нашего тимлида. Все коммиты были оформлены по госту, благодаря чему команде будет с ними проще работать.

Был переписан Readme файл. В нашем проекте Readme файл нужен для того, чтобы каждый член нашей команды смог правильно запустить и протестировать проект на своей машине. В нем был указан путь для запуска сервера и запуска отладки. Были прописаны следующие пункты: активация виртуального окружения, установка зависимостей из requirements.txt, запуск сервера, запуск отладки. Были написаны инструкции для win и unix-систем, так как команды у разных операционных систем отличаются.

### 2.2. Получение субтитров

Получением субтитров из видео занимался Дмитрий. Для получения субтитров использовался модуль “youtube\_transcript\_api”. Проблемы с использованием этого модуля и их решение были описаны выше. Требовалось написать функцию, которая анализирует ссылку и извлекает из нее уникальный ID видео.   
 Пустая функция, написанная ранее была изменена на рабочую. Для реализации этой функции использовалась библиотека «re», которая позволяет использовать шаблоны. Таким образом, была написана функция извлечения уникального ID из любого типа YouTube URL адресов.   
 Далее для получения субтитров использовалась библиотека “youtube\_transcript\_api”, с помощью которой, имея уникальный ID видео, можно получить субтитры видеоролика. Возникла проблема с приоритетом языков, ее решение описывалось выше.  
 После получения субтитров ответ преобразовывается в необходимый формат, чтобы сервер корректно мог отправить обратно ответ на запрос.   
 Для функции получения субтитров была добавлена асинхронность. Таким образом, была сделана работающая функция для получения транскрипции видеоролика и можно было начинать выполнение следующих задач.

### 2.3. Работа с запросами на сервер

Работой с запросами на сервер в нашем проекте занималась Алиса. Требовалось написать программу, которая отправляет запросы на запущенный сервер и получает ответ. В качестве запроса должна подаваться ссылка на видео, а в качестве ответа сервер должен предоставлять транскрипцию видеоролика.   
 Все поставленные задачи были выполнены, запросы корректно отправлялись на сервер и принимались с корректной информацией.

### 2.4. Telegram-бот

Созданием телеграмм бота занималась Ксения. Требовалось написать телеграмм бота с помощью библиотеки Aiogram. Стояла задача продумать функционал бота и создать интерфейс для удобной работы с ним. Были добавлены такие функции как смена языка и запуск анализа видео.  
 Все поставленные задачи были достигнуты, телеграмм бот был написан и протестирован. Интерфейсом команда осталась довольна.

### 2.5. Анализ видео с использованием нейросети

Анализом видео и работой с нейросетью занималась Анастасия. Стояла задача с использованием Api нейросети от Яндекса получить корректный анализ видео с помощью полученных субтитров. Стояла задача определиться с температурой запроса и конфигурацией запроса.   
 На созвоне команда решила, что температура ответа должна быть 0, что означает, что нейросеть использует только полученную информацию и не добавляет ничего от себя. Это гарантирует точность анализа видео и минимизирует отдаление от изначального смысла видеоматериала.  
 Был написан запрос, состоящих из множества параметров. Запрос включал в себя правильное оформление вывода, чтобы в телеграмме ответ выглядел естественно. Полученный ответ направляется к боту и сразу без обработки к пользователю.   
 Все поставленные подзадачи были выполнены в установленный срок.