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**

К У Р С О В О Й П Р О Е К Т

Тема: «Прототип платформы для адаптации текстов русскоязычных учебных материалов для иностранных студентов «EduAdapt»»

Обучающийся: Сармусокова Амина Сухробовна, К3162

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы проекта можно рассматривать с двух сторон: разработчиков самого сайта и его пользователей. Для начала обратимся к необходимости создания сайта с точки зрения его пользователей.

Говоря о том, почему проведение данной работы является нужным в наше время, важно упомянуть, что эффективная адаптация иностранных студентов имеет важное значение для того, чтобы обеспечить комфортное и гладкое вхождение в образовательный процесс. Глобализация и межкультурное взаимодействие становятся главными ценностями любого университета, и потому они сталкиваются с необходимостью адаптировать свои учебные материалы и способы их подачи для студентов из других стран. Эта необходимость становится особенно актуальной в условиях растущего количества студентов, которые выбирают учиться за границей. Кроме того, адаптация учебных материалов поможет наладить коммуникацию между студентами и преподавателями и улучшить понимание предметов учащимися.

Нетрудно заметить, что адаптация учебных материалов для иностранных студентов представляет собой сложный и занимающий много времени процесс, требующий учета многих аспектов, таких как уровень языковой компетенции отдельно взятого учащегося, его культурный опыт, а также поиск сотрудника, работа которого была бы ориентирована на данную задачу. Таким образом, в условиях активного технологического развития создание сайта, который будет решать обозначенную ранее проблему, является наиболее рациональным решением, так как он будет облегчать прохождение учебной программы иностранными студентами, не добавляя дополнительной нагрузки сотрудникам университета и выполняя свою задачу в соответствии с индивидуальными особенностями знания языка каждого учащегося.

Рассматривая актуальность проекта со стороны его разработчиков, то есть студентов, можно понять, что он представляет из себя уникальную возможность для учащихся развить множество навыков, которые будут

полезны как в учебе, так и в будущей профессиональной деятельности. Эти навыки включают в себя не только знания конкретных языков программирования и решения определенных задач создания какого-либо цифрового продукта, но и умение слаженно работать в команде для достижения общей цели.

Таким образом, результат работы полезен и разработчикам и пользователям продукта. Иностранные студенты получают возможность глубже понимать предмет, улучшать знание языка и налаживать коммуникацию с преподавателями, а создатели сайта – профессиональные навыки и опыт работы в команде.

Главной целью проекта было создать прототип платформы для упрощения образовательных материалов для иностранных студентов, обеспечивая удобный доступ к адаптированным текстам и интерактивным инструментам.

Основными задачами для достижения данной цели были:

1) реализовать модуль регистрации (логин, пароль, роль) и авторизации (логин и пароль), используя язык программирования Python и Django;

2) реализовать модуль словаря (преподаватель добавляет слово и комментарий, студент просматривает слова) с помощью языка программирования Python и библиотеки Django;

3) разработать API для упрощения текста;

4) проанализировать и сделать отчет об UX-исследовании на основе трех смежных решений;

5) создать макеты страниц сайта в Figma;

6) на основе созданных макетов разработать интерфейсы;

7) соединить интерфейсы с API серверной части (CORS);

8) посмотреть и проанализировать готовые решения (библиотеки Python) для упрощения текста и выбрать подходящий для дальнейшей работы;

9) протестировать API для регистрации и авторизации;
10) протестировать API для работы со словарем;
11) протестировать API для упрощения текста;
12) провести финальное тестирование и отладку.

1 Суть проекта и процессы работы над ним

1.1 Описание сути проекта

Проект направлен на то, чтобы создать прототип веб-приложения, которое упрощает для иностранных студентов материалы для обучения, обеспечивает доступ к облегченным текстам и интерактивным инструментам, таким как словарь незнакомых слов. Функция упрощения текста реализуется с помощью алгоритмов обработки естественного языка (NLP), позволяя автоматически создавать адаптированные версии сложных учебных материалов.

На этом сайте доступны разные опции для иностранных студентов и их преподавателей. Студенты могут получать упрощенные версии того или иного текста, а также добавлять незнакомые слова, которые они встретили, в свой собственный словарь. Преподаватели, в свою очередь, также могут добавлять слова, которые они посчитали сложными для учащихся, вместе с комментариями-пояснениями к ним, чтобы студенты могли их просматривать. Использование простого языка, сформированного алгоритмами, добавленные пояснения и дополнительные примечания – все это помогает упрощать понимание текста и облегчать процесс обучения в университете.

1.2 Процессы реализации проекта

Реализацию проекта можно разделить на несколько условных частей:

1) подготовка лидером группы материалов для обучения разработчиков;
2) обсуждение формата поддержки связи с командой, обязанностей и дедлайнов;
3) разработка серверной части;
4) создание дизайнов интерфейса;
5) написание алгоритмов упрощения текста;
6) разработка интерфейса;
7) тестирование и отладка.

После формирования группы лидер команды провел с ней первый звонок, на котором познакомился с ее членами, оценил навыки и уровень компетентности каждого участника и распределил задания. Также он согласовал основной способ коммуникации – общий чат и личные сообщения в мессенджере Telegram и еженедельные общие звонки для обсуждения прогресса и трудностей в каждой части разработки.

Тимлид создал Google-папку, в которой были собраны все необходимые материалы для начала работы: обучающие ролики и статьи для выполнения каждой задачи, таск-трекер и подробное описание обязанностей участников проекта. Подготовленное техническое задание, представленное в приложении А, было выложено на платформе “Odoo”, разработанной специально для того, чтобы реализовывать командные проекты. После знакомства и получения необходимых знаний можно было приступать к работе.

Разработка серверной части состояла из трех модулей: модуль авторизации, словаря и упрощения текста. Основными языками программирования были выбраны Python за его простоту и высокую производительность при работе с данными и Django благодаря встроенным возможностям для работы с пользователями, такими как модель User и система аутентификации.

Для создания системы регистрации и авторизации пользователей был использован язык программирования Django. Был реализован функционал регистрации, который позволяет пользователю создать аккаунт, указав логин, пароль и роль (преподаватель или студент). Для авторизации использовалась стандартная форма, которая проверяла введенные данные на корректность. Использование Django позволило ускорить процесс разработки за счет использования готовых компонентов и встроенной безопасности (например, хэширования паролей).

В рамках модуля словаря была реализована возможность для преподавателя добавлять новые слова с комментариями, которые студенты могли просматривать. Django в данном случае упростил взаимодействие с

базой данных SQLite и позволил создать модель для слов и комментариев. Таким образом, преподаватель может добавлять новые записи через админ-панель Django, а студенты – через интерфейс приложения, где отображаются все добавленные слова.

Для упрощения текста был разработан API на Python, который принимает на вход текст и возвращает его в упрощенной форме. В качестве алгоритмов для упрощения использовались базовые методы обработки естественного языка (NLP), такие как удаление сложных конструкций и замена их на более простые с использованием синонимов.

Разработка дизайна интерфейса началась с исследования трех существующих решений, на основе которого был сделан вывод о том, как должен выглядеть прототип. В онлайн-сервисе для прототипирования Figma были отрисованы страницы регистрации, авторизации, навигации, упрощения текста, а также модуля словаря. Таким образом, основными решениями в дизайне стали минимальный набор цветов в оформлении, отсутствие всплывающих окон, выделение ключевых элементов размером и цветом.

На основе созданных макетов был разработан интерфейс веб-приложения с использованием JavaScript-фреймворка Vue.js, так как он позволил обеспечить интеграцию с API и работу прототипа веб-приложения. Были внедрены модули для регистрации и авторизации пользователей, а также реализовано взаимодействие с API для обработки данных учебных материалов.

По разработанным в серверной части модулям и готовому прототипу были составлены списки кейсов и проведено ручное тестирование для проверки корректности работы продукта.

2 Личная задача

2.1 Описание задачи и процесс ее выполнения

Передо мной была поставлена задача по разработке алгоритма упрощения текста на основе методов NLP, что является направлением в машинном обучении, посвященным обработке естественного языка. Требовалось проанализировать готовые решения из библиотек Python, выбрать подходящие и написать алгоритм, который будет на вход принимать текст и возвращать его адаптированную версию, то есть ту, которую иностранным студентам будет легче понять.

Был пройден курс на платформе Stepik [1], где было подробно рассказано про основные методы NLP и существующие библиотеки Python, а также прочитаны некоторые статьи для более глубокого понимания концепции.

Говоря про анализ библиотек, использовалась встроенная библиотека NLTK, которая и была рекомендована авторами курса, но в процессе работы понадобилось дополнительно скачивать и другие, такие как SPARQLWrapper, pymorphy2, googletrans и так далее.

Работа с текстом состояла из двух основных частей: первичной и вторичной обработки. Первичную обработку текста в NLP называют этапом, на котором выполняются начальные шаги для преобразования необработанного текстового материала в форму, пригодную для дальнейшего анализа и обработки. Этот процесс включает несколько важных операций, таких как деление текста на предложения, а их – на слова (токенизация) [2]. Деление текста на предложения – первый шаг в обработке, направленный на выделение отдельных единиц текста, которые логически и грамматически завершены. Токенизация, в свою очередь, включает в себя деление текста на слова и знаки препинания. Этот процесс позволяет выделить отдельные элементы, с которыми можно работать на следующих этапах обработки. В работе токенизация проводилась с помощью решений библиотеки NLTK send_tokenize и word_tokenize [3]. Первичная обработка также включает в

себя очистку текста от стоп-слов – слов, которые имеют минимальную смысловую нагрузку, но могут путать тех, кто не является носителем русского языка. С помощью той же бибилотеки NLTK слова обрабатывались и удалялись из текста, если находились в списке русских стоп-слов. Полученный код первичной обработки представлен на рисунке 1.

```
# Токенизация на предложения
sentences = sent_tokenize(text, language='russian')

for element in sentences:
  for word in word_tokenize(element):
    # Токенизация на слова
      if word.isalpha():
        if word in stop_words and word != 'не':
          word = ''
```

Рисунок 1 – Код первичной обработки текста

После завершения первичной обработки следовала вторичная, направленная на конкретную задачу упрощения текста. С помощью функций библиотеки Pymorphy2 у каждого слова извлекались основные характеристики, такие как часть речи, род, падеж, число и т.д., затем они ставились в начальную форму [4]. Итоговая функция, выполняющая морфологический анализ слова, показана на рисунке 2.

```
def decline_word(word1, word2):
    # Инициализация морфологического анализатора
    morph = pymorphy2.MorphAnalyzer()

    # Анализируем первое слово
    parsed_word1 = morph.parse(word1)[0]  # Берем первый вариант разбора

    # Получаем падеж и число из первого слова
    case_ = parsed_word1.tag.case  # Падеж
    number_ = parsed_word1.tag.number  # Число

    # Анализируем второе слово
    parsed_word2 = morph.parse(word2)[0]
    if case_ and number_:
        return parsed_word2.inflect({number_, case_}).word
    return False
```

Рисунок 2 – Функция морфологического анализа слова

Далее с помощью SPARQLWrapper была написана функция get_lexical_level(), которая обращалась к русскому словарю и извлекала оттуда уровень слова в виде ‘B1’, ‘B2’ и т.п. [5]. Если уровень слова был ниже и равен B1, то его оставляли в начальном виде, а если нет, то проводилось преобразование. Код функции представлен на рисунке 3.

```
def get_lexical_level(word):
    # Формируем URL для запроса на сайт
    url = f"https://en.openrussian.org/ru/{word.lower()}"
    # Отправляем запрос на страницу
    response = requests.get(url)
    # Проверяем, что запрос прошел успешно
    if response.status_code != 200:
        return False
     # Создаем объект BeautifulSoup для парсинга HTML-страницы
    soup = BeautifulSoup(response.text, 'html.parser')
    # Ищем все элементы с классом 'tags', которые содержат ссылки с уровнями
    level_tags = soup.find_all('div', class_='tags')
    # Пытаемся найти уровень лексики в каждом теге <div class="tags">
    levels = []
    for tag in level_tags:
        link = tag.find('a')
        if link:
            levels.append(link.text.strip())
    # Если нашли хотя бы один уровень лексики, возвращаем его
    if levels:
        return levels
    else:
        return False
```

Рисунок 3 – Код функции изъятия уровня лексики слова

Преобразование заключалось в том, чтобы подобрать синоним и определение слова. Синоним брался из библиотеки ru_synonyms и ставился в ту же форму, что и начальное слово по ранее извлеченным характеристикам рода, падежа и т.д с помощью готовых функций из Pymorphy2. Это нужно было делать для того, чтобы текст на выходе получался читаемым, грамматически корректным и понятным. Для нахождения определения слова была написана функция get_definition(), которая посылала запрос к большой базе слов и их значений “WikiData” и брала оттуда определение. Ее код представлен на рисунке 4.

```
def get_definition(word):
    word =  morph.parse(word)[0].normal_form
    Q_id = wbgetentities(word)
    # Укажите URL SPARQL-эндпоинта
    sparql = SPARQLWrapper("https://query.wikidata.org/sparql")
    # Определите ваш SPARQL-запрос
    if Q_id is not None:
        query = f'''
        SELECT ?description WHERE {{
          wd:{Q_id} rdfs:label ?label .
          OPTIONAL {{ wd:{Q_id} schema:description ?description . }}
          FILTER (lang(?label) = "ru" && lang(?description) = "en")
        }}
        '''
    # Установите запрос и формат ответа
        sparql.setQuery(query)
        sparql.setReturnFormat(JSON)
    # Выполните запрос и получите результаты
        results = sparql.query().convert()
    # Извлечение описания
        descriptions = []
        for result in results["results"]["bindings"]:
            try:
                if "description" in result:
                    descriptions.append(result["description"]["value"])
                return descriptions[0]
            except:
              return False
    return False
```

Рисунок 4 – Функция нахождения определения слова

Если оно было найдено, то ставилось сразу за словом в скобки, иначе – нет. Так как определения на сайте были на английском языке, дополнительно была написана функция перевода translator() на основе Python-библиотеки Googletrans [6].

После всех замен и нахождений значений, слова собирались обратно в текст вместе со знаками препинания, которые присутствовали в исходном варианте. Отдельный блок кода был посвящен тому, чтобы наладить количество пробелов до и после знаков препинания, так как при тестировании выяснилось, что запятые и точки стояли от слов слишком далеко.

2.2 Анализ проведенной работы

В ходе реализации основное техническое задание, поставленное лидером команды передо мной, было выполнено. Получилось грамотно

использовать функции обработки текста из различных библиотек для получения желаемого результата. Слова в адаптированном тексте стоят в нужной форме и имеют определения, которых обычно так не хватает при встрече иностранными студентами новых незнакомых слов в учебных материалах.

Стоит отметить и то, что реализованный алгоритм ищет синонимы и определения не для всех слов, а только для тех, что входят в лексику высокого уровня, начиная с B2. То есть удалось построить функцию так, чтобы она тратила ресурсы и время на обработку текста только там, где это действительно необходимо.

Если говорить про трудности, которые появлялись в процессе выполнения задачи, их было не так много. В основном сложности возникали только в самом начале работы, когда было неясно, с чего стоит начать разработку алгоритма и какие библиотеки для этого выбрать. Также небольшие трудности возникали в тех блоках кода, где надо было обращаться к сторонним сайтам для извлечения, например, уровня лексики и определения слова, так как синтаксис обращений SPARQL тогда еще не был знаком и трудно было понять, правильно формируется запрос или нет.

Отдельно хочется отметить и работу с отдельными частями речи, так как там тоже возникало немало сложностей и многое не удалось сделать. В алгоритме работа в основном проводилась с подлежащими, так как к ним легче всего подобрать синонимы и определения, однако к другим частям речи найти подходящую базу определений и синонимов, похожую на “WikiData” не удалось.

К тому же, алгоритм не учитывает всего контекста и выбирает первый попавшийся синоним и определение, которое он встретил на сайте, к которому обращался. Таким образом, функция может искажать смысл первоначального текста при встрече омонимов и не выполнять свою первоначальную задачу.

Следовательно, не удалось создать алгоритм так, чтобы он учитывал все контекстуальные особенности текста, так как в данной сфере не хватило опыта и знаний, а также не было подходящего по мощности сервера, который бы смог учитывать также и данный аспект без увеличения времени выполнения общей функции.

Итак, в процессе разработки алгоритма упрощения текста, используя различные методы NLP, было приобретено несколько ключевых знаний и навыков, которые являются основными для работы с текстами на естественном языке и разработки подобных алгоритмов.

На первом этапе работы над алгоритмом были изучены основные понятия и методы обработки естественного языка, такие как токенизация, лемматизация, определение части речи и анализ синтаксической структуры. Работа с этими методами позволяет понять, как можно разбирать и структурировать текст для дальнейших операций.

В рамках алгоритма также был получен навык использования SPARQLWrapper для извлечения информации из открытых баз данных, таких как Wikidata. Данный навык полезен тем, что дает представление о том, как внедрять в свой проект готовые решения, извлекать необходимую информацию из открытых ресурсов в случае отсутствия библиотек с нужными функциями.

Также немаловажно отметить и тот факт, что проект научил искать и использовать библиотеки, подходящие для решения поставленных задач. Изучение документации – неотъемлемая часть работы с любой библиотекой. Были получены навыки разбора документации для понимания, какие функции и методы предоставляются для решения конкретных задач. В процессе работы с такими библиотеками, как NLTK или pymorphy2, нужно было не только находить нужные функции для токенизации, лемматизации, анализа синтаксиса или работы с частями речи, но и понимать, какие параметры необходимо передавать в функцию для корректной работы.

Вся работа проводилась планомерно и была готова к назначенному сроку. Никаких трудностей с организацией времени или прогрессом по задачам не появилось, так как руководитель проекта был постоянно на связи, вовремя отвечал на все вопросы и помогал в случае их возникновения. Немаловажную роль здесь сыграло и то, что язык программирования Python был хорошо знаком, следовательно не было трудностей с пониманием кода и изучением новой темы при доступе к многочисленным онлайн-ресурсам.

3 Взаимодействие с командой и руководителем проекта

3.1 Взаимодействие с командой

На протяжении всего периода реализации проекта общение с командой и руководителем происходило в мессенджере Telegram. Так как у каждого участника была своя задача, которая никак не затрагивала работу других, необходимости в постоянной связи с участниками проекта не было, соответственно и взаимодействия с самой командой было немного, но достаточно для работы.

3.2 Взаимодействие с руководителем и оценка его работы

С руководителем же, напротив, связь была постоянная. Было много моментов, которые надо было уточнять в процессе выполнения работы. Много поддержки и советов было получено и во время подготовки к защите. Конечно, были и те вещи, с которыми к нему нужно было обращаться с просьбой о помощи, и тут так же ответ никогда не занимал больше одного дня. Благодаря такой отзывчивости, участие в проекте действительно принесло не только ценный опыт, но и много положительных эмоций.

Руководитель отвечал на все вопросы, поддерживал постоянную связь, рационально относительно времени и ресурсов организовывал процесс работы и общался с участниками проекта наравне, поэтому не преувеличением будет сказать, что свою работу он выполнял идеально.

ЗАКЛЮЧЕНИЕ

Разработка проекта «EduAdapt» привела к созданию прототипа платформы, которая адаптирует учебные материалы на русском языке для иностранных учащихся. Главной задачей инициативы было создание прототипа веб-приложения с возможностями упрощения текстов, добавления слов в словарь и системы авторизации пользователей. В ходе реализации проекта удалось достичь заметных результатов, несмотря на то что некоторые функции и требуют доработки и улучшения.

На данный момент разработан действующий прототип платформы, который демонстрирует заявленные возможности. Серверная часть продукта, а именно модули регистрации, авторизации и словаря, была создана и корректно интегрирована. Также были разработаны дизайны интерфейса, которые в дальнейшем вместе с модулями сервера были соединены в единый прототип. Алгоритм обработки текста, хотя и требует дальнейшей доработки в связи с недостаточностью опыта в разработке и ресурсов, соответствует техническому заданию и обеспечивает ту необходимую функциональность, которая требуется для демонстрации системы. Таким образом, цели проекта были успешно достигнуты, а поставленные задачи выполнены в полном объеме. Проект можно считать успешно реализованным в соответствии с поставленной задачей создания прототипа.

Мой вклад в достижение цели был немал, так как моя задача предполагала осуществление главной функции веб-приложения – адаптация текста. Платформа «EduAdapt» направлена на упрощение понимания учебных материалов иностранцами, поэтому алгоритм, реализованный мной, удовлетворяет главному назначению платформы. Считаю, что выполненная мною работа соответствует техническому заданию и удовлетворяет тем задачам, которые были поставлены.

Итак, платформа "EduAdapt" обладает значительным потенциалом для своего будущего роста и может стать ценным ресурсом для иностранных учащихся, которые изучают материалы на русском языке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в обработку текста на Python (NLP) [Электронный ресурс]. – 2023. URL: https://stepik.org/course/182099 (дата обращения: 20.11.2024).

2. NLP: что это такое и как она работает [Электронный ресурс]. – 2023. URL: https://skillbox.ru/media/code/nlp-chto-eto-takoe-i-kak-ona-rabotaet/ (дата обращения: 25.11.2024).

3. Анализ текстовых данных с помощью NLTK и Python [Электронный ресурс]. – 2023. URL: https://habr.com/ru/companies/otus/articles/774498/ (дата обращения: 01.11.2024).

4. Руководство пользователя [Электронный ресурс]. – 2020. URL: https://pymorphy2.readthedocs.io/en/stable/user/guide.html (дата обращения: 30.11.2024).

5. SPARQL Endpoint interface to Python [Электронный ресурс]. – 2019. URL: https://sparqlwrapper.readthedocs.io/en/latest/main.html (дата обращения: 04.11.2024).

6. Googletrans: Free and Unlimited Google translate API for Python [Электронный ресурс]. – 2020. URL: https://py-googletrans.readthedocs.io/en/latest/ (дата обращения: 02.11.2024).

ПРИЛОЖЕНИЕ А
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

- **Название проекта**
  Прототип платформы для адаптации текстов русскоязычных учебных материалов для иностранных студентов «EduAdapt»
- **Цель (назначение)**
  Создать прототип платформы для упрощения образовательных материалов для иностранных студентов, обеспечивая удобный доступ к адаптированным текстам и интерактивным инструментам.
- **Сроки выполнения**

  Начало 15 ноября 2024 г. Окончание 20 декабря 2024 г.
- **Руководитель проекта**
  Урнежус Филипс, студент К4240, факультет ИКТ, аналитик Центра карьеры обучающихся ИТМО.
- **Термины и сокращения**
  API — Интерфейс программирования приложений, используется для связи между серверной частью (бэкенд) и интерфейсом (фронтенд).
  MVC — Архитектурный паттерн «Модель-Представление-Контроллер», помогает разделить логику приложения на три части для упрощения разработки и поддержки.
  NLP — Обработка естественного языка, набор технологий для работы с текстами, включая упрощение и анализ сложности текста.
  JWT — Токен доступа, используемый для авторизации и поддержания сессии пользователей.
  SQLite — Легковесная база данных, используемая для хранения данных в приложении.
- **Технические требования**

Технические:

- Серверная часть построена на Django с использованием Python.
- База данных SQLite для хранения учебных материалов и пользовательских данных.
- REST API для взаимодействия клиентской и серверной частей с использованием Django REST Framework.
- Поддержка кроссбраузерности.

Программные:

- Реализация регистрации, авторизации и функционала с использованием JWT.
- Безопасная валидация данных на стороне сервера для предотвращения SQL-инъекций и XSS-атак.

- Разработка адаптивного интерфейса на Vue.js, совместимого с различными устройствами.

Эргономические:

- Удобное размещение интерфейсных элементов для обеспечения интуитивного взаимодействия пользователей.
- Отдельные интерфейсы для преподавателей и студентов с четким разделением функционала.

- **Содержание работы**

№	Этапы проекта	Сроки выполнения этапов	Ответственный за этап	Вид представления результатов этапа
1	Планирование проекта	15.11.2024 - 18.11.2024	Урнежус Ф.	Готовое техническое задание для дальнейшей работы
2	Разработка серверной части	21.11.2024 - 06.12.2024	Бондар И.А.	Готовый исходный код серверной части прототипа, согласно задачам
3	Разработка интерфейса	25.11.2024 - 12.12.2024	Просветова В. Д.	Разработанные интерфейсы (Регистрация и авторизация; Главная страница с навигацией к разделам сайта; страница с функциями упрощения текста, модуль словаря (студент и преподаватель)); соединение интерфейсов с серверной частью проекта.
4	Разработка алгоритма упрощения текста	18.11.2024 - 10.12.2024	Сармусокова А. С.	Реализованный метод упрощения текста с помощью

				NLP алгоритмов. согласно задаче
5	Тестирование и отладка системы	25.11.2024 - 15.12.2024	Цзян Ю.	Проведенное тестирование на основе составленному списку кейсов по прототипу платформы.
6	Проведение UX-исследования и создание визуальных прототипов	18.11.2024 - 02.12.2024	Грачева С. А.	Анализ и отчет об UX-исследовании на основе трех смежных решений; создание макетов страниц в Figma (Регистрация и авторизация; Главная страница с навигацией к разделам сайта; страница с функциями упрощения текста, модуль словаря (студент и преподаватель)
7	Подготовка документации и защита проекта	15.12.2024 - 18.12.2024	Бондар И. А.	Сдача отчета, предоставление доклада с презентацией и демонстрация работы прототипа платформы.

7* Сделать отдельный пункт и расписать задачи для каждого члена команды.

Бондар И. А.

- **Создать API для регистрации и авторизации**
    - Реализовать модуль регистрации (логин, пароль, роль) и авторизации (логин и пароль) используя ЯП Python и Django.
- **Разработать API для работы со словарем**
    - Реализовать модуль словаря (преподаватель добавляет слово и комментарий, студент просматривает слова) с помощью ЯП Python и библиотекой Django.
- **Разработать API для упрощения текста**

- Необходимо разработать API на языке Python, которое будет выполнять следующую функциональность:
- Принимать текстовые данные: API должно принимать текстовые данные в формате JSON через HTTP-запрос (например, POST-запрос).
- Обрабатывать текст: Полученный текст должен быть обработан с использованием встроенного NLP-алгоритма. Алгоритм может выполнять такие действия, как анализ, модификация текста, извлечение сущностей, токенизация, лемматизация.
- Возвращать результат: API возвращает результат обработки текста в формате JSON. Результат должен содержать как исходный текст, так и обработанный (если применимо), либо дополнительные данные, полученные в результате обработки.

Просветова В. Д.

- **Реализовать интерфейс для сайта**
  - На основе созданных макетов разработать интерфейсы (Регистрация и авторизация; Главная страница с навигацией к разделам сайта; страница с функциями упрощения текста, модуль словаря (студент и преподаватель)).
- **Интегрировать интерфейс с API**
  - Соединить интерфейсы с API серверной части (CORS).

Сармусокова А. С.

- **Исследовать и разработать функции упрощения текста**
  - Посмотреть и проанализировать готовые решения (библиотеки Python) для упрощения текста и выбрать подходящий для дальнейшей работы. Реализовать метод упрощения в Google Colab.

Грачева С. А.

- **Провести UX-исследование для улучшения удобства интерфейса**
  - Проанализировать и сделать отчет об UX-исследовании на основе трех смежных решений.
- **Создать макеты страниц**
  - Создать макеты страниц в Figma (Регистрация и авторизация; Главная страница с навигацией к разделам сайта; страница с функциями упрощения текста, модуль словаря (студент и преподаватель)).

Цзян Ю.

- **Протестировать API для регистрации и авторизации**
  - Составить список кейсов и провести ручное тестирование по разработанному модулю.
- **Протестировать API для работы со словарем**
  - Составить список кейсов и провести ручное тестирование по разработанному модулю.
- **Протестировать API для упрощения текста**

    - Составить список кейсов и провести ручное тестирование по разработанному модулю.
- **Провести финальное тестирование и отладку**
    - Составить список кейсов и провести ручное тестирование по разработанному прототипу платформы.

• Основные результаты работы и формы их представления

Готовый прототип платформы EduAdapt представляет собой веб-приложение для адаптации и изучения русскоязычных учебных материалов. Функционал платформы включает модули для упрощения текстов и работы со словарем, обеспечивая удобство и эффективность образовательного процесса.
Функция упрощения текста использует алгоритмы обработки естественного языка (NLP), позволяя автоматически создавать адаптированные версии сложных учебных материалов. Это упрощает понимание текста за счёт использования простого языка, пояснений и дополнительных примечаний.
Словарь представляет собой инструмент, который позволяет преподавателям добавлять новые слова и комментарии, а студентам — просматривать эти записи в адаптированном формате. Он включает интегрированные подсказки, помогающие пользователю понять значение новых слов в контексте учебного материала.