МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ИНТЕЛЛЕКТУАЛЬНЫЙ БОТ-ПОМОЩНИК УЧИТЕЛЯ

Студент 26.05.26 (подпись, дата) Тельнова Александра Дмитриевна (ФИО полностью)

Руководитель 26.05.2026 (подпись, дата) Митрохина Н.Ю. (фамилия, инициалы)

Нормоконтролёр 26.05.2026 (подпись, дата) Бычков А.С. (фамилия, инициалы)

Рецензент нач. УИТ (должность, место работы) 27.05.2026 (подпись, дата) Севастьянов Д.К. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 № 14)

Заведующий кафедрой _____ (подпись) М.А. Митрохин (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 04 июня 2026 № 5)

Секретарь ГЭК _____ (подпись) Е.И. Гурин (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ М.А. Митрохин
«20» сентября 2024 г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Тельнова Александра Дмитриевна гр. 24ВВИм1 факультета ВТ

2.Тема работы «Интеллектуальный бот-помощник учителя»

Тема утверждена приказом ПГУ № 920/о от "20" сентября 2024 г.

3. Руководитель магистерской диссертации к.т.н., доцент Митрохина Н.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать интеллектуального Telegram-бота-помощника учителя, предназначенного для автоматизации части рутинных и интеллектуальных задач педагогической деятельности.

Разрабатываемая система должна обеспечивать работу с учебными данными: просмотр расписания учителя, просмотр оценок учащихся, расчёт среднего и средневзвешенного балла, учёт посещаемости, формирование отчётов об успеваемости и отправку отчётов родителям или законным представителям учащихся.

Система также должна включать интеллектуальные функции: генерацию плана урока, генерацию письменной работы, анализ письменных работ учащихся по фотографиям, сравнение решения ученика с эталонным решением и формирование рекомендуемой отметки.

В качестве исходных данных используются сведения об учебном процессе, хранящиеся в реляционной базе данных, а также текстовые параметры, вводимые пользователем в Telegram-боте, документы с учебными материалами и изображения письменных работ.

5. Перечень подлежащих разработке вопросов

1. Обзор существующих образовательных решений, их функциональных возможностей, достоинств и ограничений.
2. Анализ методов и алгоритмов автоматизации педагогической деятельности.
3. Определение требований к интеллектуальному боту-помощнику учителя, его функциональным характеристикам, входным и выходным данным.
4. Проектирование архитектуры системы, базы данных, сервисных модулей и предобученных моделей искусственного интеллекта.
5. Разработка базы данных, подготовка учебных данных и реализация базового функционала системы.
6. Реализация интеллектуального функционала системы с использованием предобученных моделей искусственного интеллекта.
7. Проверка работы реализованной системы на наборе учебных данных и анализ полученных результатов.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	15%	20.09.2024 – 23.11.2024	
2	Анализ существующих решений, методов и алгоритмов	15%	24.11.2024 – 24.03.2025	
3	Проектирование системы и выбор средств разработки	15%	25.03.2025 – 28.05.2025	
4	Разработка базы данных и базового функционала системы	20%	01.09.2025 – 31.12.2025	
5	Разработка интеллектуального функционала системы	20%	01.01.2026 – 31.03.2026	
6	Проверка работы системы и анализ результатов	5%	01.04.2026 – 20.04.2026	
7	Оформление пояснительной записки	10%	21.04.2026 – 21.05.2026	

Дата выдачи задания "20" сентября 2024 г.

Руководитель магистерской диссертации _____ 20.09.2024 Митрохина Н.Ю.
(подпись, дата)

Задание к исполнению принял студент _____ 20.09.2024 Тельнова А.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 01.06.2026 Фионова Л.Р.

(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 130 листов, 1 рисунок, 2 таблицы, 37 источников, 3 приложения.

TELEGRAM-БОТ, ИНТЕЛЛЕКТУАЛЬНЫЙ ПОМОЩНИК, БОЛЬШИЕ ЯЗЫКОВЫЕ МОДЕЛИ, VISION-МОДЕЛЬ, PYTHON, SQLITE, OLLAMA

Целью магистерской диссертации является исследование возможностей применения предобученных нейросетевых моделей в образовательной сфере, а также разработка интеллектуального Telegram-бота-помощника учителя для автоматизации отдельных рутинных, методических и аналитических задач педагогической деятельности.

В работе рассмотрена предметная область, связанная с применением искусственного интеллекта в образовании. Проведён анализ особенностей педагогической деятельности, существующих образовательных решений, а также методов и алгоритмов, которые могут применяться для автоматизации работы учителя.

В ходе выполнения работы были определены требования к интеллектуальному боту-помощнику учителя, его функциональным характеристикам, входным и выходным данным. Разработана архитектура системы, основанная на взаимодействии Telegram-бота, реляционной базы данных, сервисных модулей и предобученных моделей искусственного интеллекта.

В результате была реализована система, обеспечивающая работу с учебными данными учителя и учащихся, взаимодействие с законными представителями учащихся. Также реализованы функции, позволяющие педагогу упростить подготовку к урокам и проверку письменных работ.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5

ВВЕДЕНИЕ 7

1 Анализ предметной области и постановка задачи исследования 11

1.1 Особенности педагогической деятельности 11

1.2 Анализ существующих решений, применяемых в образовательной практике 12

1.2.1 Сервис для изучения иностранных языков Duolingo 13

1.2.2 Нейросетевой помощник для учителя Eduaide.ai 14

1.3 Анализ методов и алгоритмов 16

1.3.1 Методы для расчета итоговой оценки учащегося 16

1.3.2 Методы формирования сводки по оценкам учащегося 18

1.3.3 Методы и алгоритмы для автоматизированной проверки письменных работ учащихся 23

1.3.4 Методы и алгоритмы для автоматизированного составления учебных материалов 31

1.4 Постановка задачи исследования 37

2 Проектирование системы интеллектуального бота-помощника учителя 41

2.1 Функциональное назначение 41

2.2 Требования к системе 42

2.2.1 Общие требования к системе 42

2.2.2 Требования к функциональным характеристикам 43

2.2.3 Требования к организации входных и выходных данных 45

2.3 Выбор методов и алгоритмов для реализации системы 47

2.4 Проектирование архитектуры системы 52

3 Реализация интеллектуального бота-помощника учителя 55

3.1 Реализация базы данных и подготовка исходных данных 55

3.1.1 Формирование и генерация исходных данных 55
3.1.2 Реализация базы данных SQLite 56
3.2 Разработка Telegram-бота и организация взаимодействия модулей 57
3.2.1 Создание Telegram-бота 57
3.2.2 Организация взаимодействия Telegram-бота с базой данных и сервисными модулями 58
3.3 Реализация базового функционала системы 61
3.3.1 Просмотр расписания учителя 61
3.3.2 Просмотр оценок ученика 62
3.3.3 Расчёт среднего, средневзвешенного балла и посещаемости 63
3.3.4 Формирование и отправка отчётов 65
3.4 Реализация интеллектуального функционала системы 67
3.4.1 Генерация плана урока с использованием LLM 67
3.4.2 Генерация письменной работы с использованием LLM 69
3.4.3 Анализ письменных работ учащихся на основе vision-модели 72
3.4.4 Механизм оценки качества решения и формирования рекомендуемой отметки 74
ЗАКЛЮЧЕНИЕ 78
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 79
ПРИЛОЖЕНИЕ А – Листинг программы 83
ПРИЛОЖЕНИЕ Б – Презентация 120
ПРИЛОЖЕНИЕ В – Схемы 1218

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ИИ – искусственный интеллект

ФИО – фамилия, имя, отчество

API (Application Programming Interface) – программный интерфейс приложения

BiLSTM (Bidirectional Long Short-term Memory) – двунаправленная долговременная кратковременная память

CNN (Convolutional Neural Network) – свёрточная нейронная сеть

COVID-19 (coronavirus disease 2019) – коронавирусное заболевание 2019 года

CRNN (Convolutional Recurrent Neural Networks) – сверточная рекуррентная нейронная сеть

CSV (Comma-Separated Values) – значения, разделённые запятыми

CTC (Connectionist Temporal Classification) – коннекционистская временная классификация

DOCX (Document XML) – формат файлов для текстовых документов, основанный на стандарте Office Open XML

GPT (Generative Pre-trained Transformer) – генеративный предобученный трансформер

HTR (Handwritten Text Recognition) – распознавание рукописного текста

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

JPEG (Joint Photographic Experts Group) – растровый графический формат изображений и фотографий с высокой степенью сжатия

JSON (JavaScript Object Notation) – JavaScript объектная нотация

LLM (Large Language Model) – большая языковая модель

LSTM (Long Short-Term Memory) – долгая краткосрочная память

NLP (Natural Language Processing) – обработка естественного языка

OCR (Optical Character Recognition) – оптическое распознавание символов

PDF (Portable Document Format) – переносимый формат документов

RAG (Retrieval Augmented Generation) – генерация с дополненной выборкой

RGB (Red, Green, Blue) – красный, зелёный, синий – аддитивная цветовая модель

RNN (Recurrent Neural Network) – рекуррентная нейронная сеть

SMTP (Simple Mail Transfer Protocol) – простой протокол передачи почты

SQL (Structured Query Language) – язык структурированных запросов

TXT (Text File Document) – стандартное расширение для текстовых файлов

ВВЕДЕНИЕ

Искусственный интеллект в настоящее время активно внедряется в различные области общественной и профессиональной деятельности, в том числе в сферу образования. Его применение создает дополнительные возможности для повышения качества обучения, индивидуализации образовательного процесса, а также для совершенствования взаимодействия между учащимися, педагогами и цифровыми образовательными платформами.

Следует отметить, что в условиях современного образования технологии искусственного интеллекта приобретают всё большую значимость. Переход образовательных организаций по всему миру к дистанционному формату обучения в период пандемии COVID-19 выявил ряд ограничений традиционной образовательной модели. В частности, стали более заметны проблемы недостаточной персонализации обучения, нехватки обратной связи и высокой нагрузки на педагогов. В связи с этим возникла необходимость в разработке и внедрении новых технологических решений, способных сделать образовательный процесс более гибким, адаптивным и эффективным.

Искусственный интеллект может рассматриваться как инструмент поддержки педагогической деятельности. Его использование позволяет преподавателям уделять больше внимания творческой, методической и воспитательной составляющим учебного процесса, частично передавая автоматизированным системам выполнение повторяющихся задач. К таким задачам относятся проверка и анализ письменных работ, организация занятий, обработка учебных данных и выставление отметок.

Образовательные чат-боты способны взаимодействовать с пользователями, обеспечивая ответы на вопросы, предоставляя информацию, помогая в выполнении заданий или в подготовке к урокам и занятиям.

Чат-боты имеют множество преимуществ перед использованием иных ресурсов и, в частности, программных приложений: боты легко установить, не используя память устройства, ссылки на бот легче распространить, его проще создать и использовать.

Актуальность темы магистерской диссертации обусловлена необходимостью разработки программных средств, позволяющих автоматизировать отдельные виды педагогической деятельности и объединить в одной системе работу с учебными данными, учебными материалами и интеллектуальную помощь в учебном процессе.

Объектом исследования является профессиональная деятельность педагога, включающая административно-организационные процессы, реализуемые в рамках должностных обязанностей.

Предметом исследования являются методы и программные средства автоматизации административно-организационной деятельности педагога.

Целью данной магистерской диссертации является разработка и исследование интеллектуального-бота помощника учителя, обеспечивающего автоматизацию базовых рутинных и интеллектуальных задач педагога и повышение качества обучения.

В соответствии с поставленной целью были определены следующие восемь задач:

1) Анализ особенностей взаимодействия человека с интеллектуальными системами,

2) Анализ особенностей педагогической деятельности, определение задач, которые могут быть частично автоматизированы с помощью интеллектуальной системы,

3) Обзор существующих решений, использующих технологии искусственного интеллекта, определение их достоинств и ограничений,

4) Анализ методов расчёта успеваемости, формирования отчётности, генерации учебных материалов и анализа письменных работ учащихся,

5) Проектирование архитектуры Telegram-бота-помощника учителя и структуры реляционной базы данных для хранения учебных данных,

6) Разработка методов по автоматизации рутинных задач учителя: просмотр расписания на неделю, просмотр отметок учеников, расчет среднего и средневзвешенного балла, учёт посещаемости, формирование и отправку отчётов об успеваемости,

7) Разработка методов по автоматизации интеллектуальных задач учителя: генерацию плана урока, генерацию письменной работы и анализ письменных работ учащихся на основе предобученных моделей искусственного интеллекта,

8) Тестирование и анализ эффективности разработанной системы на демонстрационном наборе учебных данных.

Методы исследования

Для решения поставленных задач использовались методы анализа предметной области, методы сбора и анализа данных, проектирования реляционных баз данных, программирования на языке Python, разработки Telegram-ботов, обработки естественного языка, интеллектуальной обработки изображений и тестирования программного обеспечения.

Актуальность (новизна) исследования состоит в разработке и исследовании подхода к интеграции предобученных ИИ-моделей в систему поддержки деятельности учителя, включающей структурирование входных запросов, постобработку результатов, алгоритмы сравнения решений и механизм автоматизированного оценивания.

Практическая и научная значимость работы заключается в разработке интеллектуального Telegram-бота, который может использоваться педагогом для оперативного доступа к учебным данным, анализа успеваемости учащихся, формирования отчётов для родителей, подготовки планов уроков, генерации письменных работ и предварительной проверки решений учащихся. Использование Telegram-интерфейса обеспечивает доступность

системы. Применение локальной базы данных и предобученных моделей позволяет адаптировать решение под задачи образовательной организации.

Границы исследования: основной акцент ставится на разработке и исследовании интеллектуального Telegram-бота, обеспечивающего автоматизацию рутинной работы с учебными данными, а также поддержку более сложных методических и аналитических задач с использованием предобученных моделей искусственного интеллекта. В рамках данного исследования не рассматриваются задачи интеграции разрабатываемой системы с внешними образовательными платформами и электронными журналами, а также обучение собственных нейросетевых моделей.

1 Анализ предметной области и постановка задачи исследования

1.1 Особенности педагогической деятельности

Работа школьного учителя в современных условиях относится к числу наиболее ответственных направлений профессиональной деятельности, поскольку она напрямую связана с организацией учебного процесса, развитием учащихся и обеспечением качества образования. Вместе с тем следует отметить, что данная профессия сопровождается значительным количеством организационных, методических и эмоциональных нагрузок.

С каждым годом круг обязанностей педагога расширяется. Помимо проведения уроков, учитель выполняет задачи, связанные с внеурочной деятельностью, ведением документации, классным руководством и взаимодействием с участниками образовательного процесса. Таким образом, увеличение объёма таких задач может отрицательно сказываться не только на качестве образовательного процесса, но и на состоянии самих педагогов.

Искусственный интеллект активно внедряется в различные сферы общественной и профессиональной деятельности, включая образование. Его применение открывает новые возможности для повышения качества обучения, индивидуализации образовательного процесса и совершенствования взаимодействия между учащимися, педагогами и цифровыми образовательными платформами.

Значительная часть рабочего времени учителя затрачивается на проверку тетрадей и контрольных работ, подготовку планов уроков, а также выполнение внеурочной и административной работы. Подобная нагрузка является одной из причин профессионального выгорания педагогов и снижает привлекательность преподавательской деятельности для квалифицированных специалистов.

Использование искусственного интеллекта, а также машинного обучения может позволить преподавателям сделать процесс обучения более

увлекательным за счет учета интересов, сильных и слабых сторон каждого обучающегося, подготовки занимательных учебных планов и домашних зданий.

Кроме того, интеллектуальные боты-помощники экономят время преподавателям, выполняя рутинные задачи. Например, ИИ может взять на себя создание подробных отчетов по успеваемости, доступ к которым могут получить, родители или опекуны, а также сами учащиеся, проверку сочинений, эссе, тестов, контрольных работ.

Отсутствие необходимости решать рутинные задачи может повлиять на подбор и сохранение педагогических кадров. У педагогов с помощью интеллектуальных помощников появляется больше времени на работу и общение с учениками, планирование занятий, повышение квалификации.

1.2 Анализ существующих решений, применяемых в образовательной практике

Технологии ИИ уже начинают менять учебный процесс, усиливая его эффективность. Многие образовательные платформы и приложения развивают, внедряют и используют ИИ для организации обучения пользователей, создания новых форм учебного взаимодействия и стратегий оценивания обучения, предоставления обратной связи. ИИ способствует созданию персонализированной траектории обучения в процессе проектирования онлайн-курсов. Обучающиеся и педагоги получают доступ к различным ресурсам и базам данных, причем нейросети анализируют образовательные метаданные по запросам, помогают в решении различных профессиональных задач.

Далее будет рассмотрено несколько программных продуктов, использующих технологии ИИ, которые позволяют пользователям упростить процесс обучения и преподавания.

1.2.1 Сервис для изучения иностранных языков Duolingo

Duolingo – бесплатная платформа для изучения языков с помощью весёлых коротких уроков. Тренирует говорить, читать, понимать на слух и писать, помогает изучать грамматику и пополнять словарный запас. В приложении Duolingo можно изучать более 40 разных языков.

Алгоритмы приложения составляют индивидуальные программы для каждого пользователя в зависимости от его уровня. Для этого разработчики используют систему Birdbrain 2 на основе искусственного интеллекта.

Система Birdbrain 2 ежедневно анализирует почти миллиард упражнений, которые выполняют пользователи. Это помогает найти закономерности между уровнем владения языком и результатами, которые показывает пользователь.

Приложение составляет индивидуальные программы обучения для каждого пользователя. Система на старте анализирует, как обучающийся выполняет задания. Например, если у пользователя средний уровень, то система предложит упражнения соответствующей сложности.

Birdbrain 2 оценивает скорость и правильность выполнения заданий. Затем проводит анализ и предлагает подходящие по уровню упражнения. Система определяет уровень пользователя и на его основе формирует учебную программу.

Помимо Birdbrain 2 образовательная платформа Duolingo использует преимущества GPT-4, новейшей технологии от OpenAI, чтобы сделать обучение ещё более эффективным. Платформа Duolingo предлагает своим пользователям подписку Duolingo Max, которая предоставляет учащимся доступ к двум новым функциям и упражнениям: «Объясни мой ответ» и «Ролевая игра».

Функция «Объяснить мой ответ» позволяет учащимся узнать больше о своём ответе на уроке независимо от того, был ли он правильным или неправильным. ИИ объяснит учащимся в чате почему их ответ был верным

или неверным, а также приведет примеры или даст дополнительные пояснения.

Функция «Ролевая игра» позволяет учащимся практиковать навыки общения в реальном мире с персонажами из приложения. Учащиеся могут обсудить планы на отпуск с Лин, заказать кофе в кафе в Париже, пойти за мебелью с Эдди или пригласить друга в поход. После взаимодействия учащиеся получают обратную связь на основе искусственного интеллекта о точности и сложности их ответов, а также советы для будущих разговоров.

Сервис для изучения иностранных языков Duolingo предоставляет пользователям уникальные функции и упражнения на базе ИИ, помогающие сделать процесс обучения более простым и эффективным. Однако у платформы есть несколько недостатков, а именно: взаимодействие с интеллектуальным помощником доступно далеко не всем пользователям. Подписка Duolingo Max достаточно дорогостоящая – $30 в месяц (или $168 в год). Кроме этого, единственные курсы, на которых можно использовать новые функции – это испанский и французский для англоговорящих пользователей на устройствах с iOS. На момент запуска Duolingo Max доступен в США, Великобритании, Ирландии, Канаде, Австралии и Новой Зеландии.

1.2.2 Нейросетевой помощник для учителя Eduaide.ai

Eduaide.ai – это платформа, которая предлагает учителям рабочее пространство на базе ИИ, где они могут создавать учебные материалы, автоматизировать административные задачи и получать целевую обратную связь.

Платформа Eduaide предоставляет пользователям широкий набор инструментов, основанных на технологиях искусственного интеллекта. В состав сервиса входит более 150 функций, ориентированных на поддержку педагогической деятельности. Они могут использоваться для оценки знаний учащихся, создания учебного контента, планирования уроков, подготовки

интерактивных заданий, внедрения игровых элементов в обучение, взаимодействия с родителями, управления образовательными ресурсами и решения других профессиональных задач учителя.

Платформа была разработана с учётом особенностей реального учебного процесса. Её основное назначение заключается в снижении временных затрат педагога на выполнение административных и методических задач, что позволяет уделять больше внимания непосредственной организации обучения.

К основным преимуществам платформы можно отнести следующие функции на базе искусственного интеллекта:

– создание плана урока. Eduaide.ai предоставляет возможность формировать долгосрочные учебные планы, поддерживающие необходимый уровень сложности материала и способствующие постепенному формированию и закреплению знаний учащихся,

– создание индивидуальных систем оценивания. Данная функция позволяет педагогу оценивать уровень усвоения учебного материала с помощью индивидуально сформированных заданий,

– диаграмма привязки. Позволяет создавать наглядные конспекты уроков для закрепления ключевых концепций,

– краткое содержание книги. Инструмент выделяет ключевые темы и контекст, чтобы облегчить ученикам понимание смысла книги,

– создание нового контента: генерация текста, изображений, аудио,

– бот для обратной связи. Генерирует конкретные, своевременные и конструктивные отчеты и отзывы о работах учащихся,

– встроенный редактор, позволяющий в режиме реального времени вносить изменения в текст. В отличие от ChatGPT, ответ которого требует переформатирования и редактирования с помощью других инструментов, Eduaide.ai позволяет быстро скорректировать полученный текст сразу после генерации ответа,

– детальный контроль над результатами, созданными с помощью искусственного интеллекта, позволяет преподавателям настраивать ответы в соответствии с их потребностями в классе.

К недостаткам Eduaide.ai относится высокая стоимость месячной/годовой подписки – $6 в месяц и $50 в год. В бесплатной версии Eduaide.ai функционал сильно ограничен, а также накладывается ограничение в 15 сеансов генераций ответов от ИИ. Кроме этого, сайт и функционал Eduaide.ai недоступен для пользователей из России.

1.3 Анализ методов и алгоритмов

1.3.1 Методы для расчета итоговой оценки учащегося

Формирование итоговой оценки за четверть или учебный год осуществляется на основе совокупности отметок, полученных учащимся в течение соответствующего периода. При этом четвертные оценки имеют существенное значение, поскольку именно они во многом определяют итоговый результат обучения. Порядок выставления итоговых отметок в образовательных организациях регулируется нормативными документами, однако конкретные способы подсчёта могут различаться. В связи с этим для учащихся важно иметь возможность отслеживать текущую успеваемость и понимать, каким образом может быть сформирована итоговая оценка.

Является целесообразным упростить и ускорить процесс расчёта итогового балла для педагогов. Для решения данной задачи может применяться интеллектуальный помощник, позволяющий автоматически рассчитывать результаты по различным видам учебной деятельности. Такой инструмент может использоваться при обработке оценок за контрольные работы, домашние задания, тесты и другие формы проверки знаний. Применение подобных систем делает процесс подведения итогов более удобным и снижает объём ручных вычислений.

Итоговый результат складывается из баллов и отметок, полученных учащимся за разные виды заданий, контрольных мероприятий и экзаменов. В

зависимости от правил конкретной образовательной организации оценка может формироваться на основе текущих результатов за четверть либо с учётом итоговых проверочных работ. Кроме того, на процесс выставления отметки могут влиять результаты тестирования, активность ученика на уроках, выполнение домашних заданий, участие в конкурсах и другие показатели учебной деятельности.

Для подсчета итоговой оценки выделяют два основных метода: простое среднее арифметическое и средневзвешенная оценка.

1.3.1.1 Простое среднее арифметическое

Итоговая оценка рассчитывается путем вычисления среднего арифметического всех полученных отметок за определенный период (четверть, полугодие, год). Этот метод предполагает равнозначность всех результатов.

Формула выглядит следующим образом:

$$O = \frac{\sum_{i=1}^{n} x_i}{n},$$

где O – итоговая оценка,

x_i – отдельные отметки ученика,

n – количество оценок.

1.3.1.2 Средневзвешенная оценка

Средний балл за четверть рассчитывается с учетом всех оценок: контрольных, тестов, проектов и устных ответов. В разных школах могут быть свои системы подсчета, но обычно учитывается вес каждой работы.

Коэффициенты, присвоенные различным предметам, существенно влияют на итоговую оценку. Например, если экзамен по математике оценивается с более высоким коэффициентом, его результат будет иметь больший вес при расчете итоговой оценки. Оценивается вклад каждого типа

задания в общую оценку отдельно, затем суммируются взвешенные показатели.

Формула для расчета:

$$O = w_1 * o_1 + w_2 * o_2 + \cdots + w_n * o_n,$$

где w_i – вес отдельного вида работы,

o_i – оценка за этот вид работы.

1.3.2 Методы формирования сводки по оценкам учащегося

Сводная отчетность по оценкам ученика в школе представляет собой обобщенный документ, содержащий подробную информацию обо всех академических успехах школьника за определенный период времени. Этот отчет объединяет различные виды оценочной информации, такие как итоговые отметки по предметам, промежуточные и итоговые контрольные работы, тесты, участие в олимпиадах и конкурсах, посещаемость уроков и поведение ученика.

Периодическое формирование такой отчетности позволяет предоставить ученикам, их законным представителям и администрации объективную картину уровня знаний и достижений; определить сильные стороны и области, нуждающихся в улучшении; осуществлять мониторинг прогресса ученика от одного периода к другому. На основе сводной отчетности образовательные организации могут формировать базы данных для принятия решений администрацией школы касательно индивидуальных образовательных траекторий ученика.

Сводная отчетность может включать как количественные оценки, средние баллы, так и их распределение, и дополнительные сведения о классе, в котором обучается ученик:

– оценки за проверочные и контрольные мероприятия,

– средний балл ученика за выбранный период, отражающий общий уровень успеваемости за определенный период,

– коэффициент успеваемости – отношение количества успешно сданных экзаменов к общему числу испытаний,

– итоговые оценки по основным дисциплинам,

– процент отличников и неуспевающих – индикатор поляризации класса или группы,

– абсолютная успеваемость – доля учащихся, не имеющих неудовлетворительных оценок,

– качественная успеваемость – доля учащихся, успевающих на «4» и «5»,

– комментарии учителя по поводу особенностей восприятия материала ребенком.

Критически важно также учитывать контекстные показатели – факторы, влияющие на образовательные результаты, но не являющиеся прямым отражением успеваемости:

– информация о пропущенных занятиях и причинах отсутствия,

– социально-демографические факторы – они могут объяснять до 30% вариативности академических результатов,

– психологический климат в классе – влияет на успеваемость через уровень стресса и восприятие образовательной среды,

– доступность образовательных ресурсов – разрыв в оснащенности может создавать неравные условия для демонстрации реальных знаний.

Такой отчет является важным инструментом коммуникации между школой, родителями и учениками, помогающим выявить проблемы и поддержать развитие каждой личности.

Сводка об успеваемости учеников может быть представлена в таких форматах, как текстовый отчет, тепловая карта успеваемости, диаграмма и таблица прогресса.

1.3.2.1 Текстовый отчет

Текстовый отчёт об успеваемости ученика – это документ, отражающий успехи учащегося в освоении образовательной программы. Такой отчёт составляется преподавателем на основании результатов контрольных работ, тестов, экзаменов, текущих оценок и участия учащегося в учебной деятельности.

Текстовый отчет об успеваемости содержит основные элементы:

– общие сведения. ФИО ученика, класс, образовательная организация,

– уровень успеваемости. Оценка общей успешности освоения учебного материала (средний балл, процент усвоенных предметов),

– оценки по предметам. Итоги промежуточной аттестации и итоговые оценки по каждой учебной дисциплине,

– динамика достижений. Отражение изменений уровня успеваемости от четверти к четверти или от года к году. Может включать комментарии педагогов относительно прогресса или регресса ученика,

– посещаемость занятий. Информация о пропусках уроков, причинах отсутствия и посещениях факультативных мероприятий,

– дополнительные достижения. Учет участия ученика в олимпиадах, конкурсах, спортивных мероприятиях, проектной работе и другой внеклассной активности,

– рекомендации и комментарии учителя. Советы преподавателей родителям и самому ученику по улучшению учебных показателей, рекомендации по дополнительному обучению, занятиям, развитию индивидуальных способностей,

– заключение и выводы. Общая оценка успехов и перспектив развития ребёнка, предложение дальнейших действий родителей и администрации школы.

1.3.2.2 Тепловая карта успеваемости

Тепловая карта успеваемости представляет собой способ визуального отображения учебных результатов учащегося. С её помощью в графической форме могут быть представлены различные показатели: текущие оценки, посещаемость занятий, уровень освоения учебного материала и другие параметры, характеризующие образовательные достижения обучающегося.

Для построения тепловой карты используются цветовые обозначения разной интенсивности. Каждый цвет соответствует определённому уровню успеваемости. Как правило, более «тёплые» оттенки указывают на наличие затруднений, тогда как «холодные» оттенки отражают положительную динамику и устойчивые учебные результаты.

Для интерпретации тепловой карты могут быть использованы следующие цветовые категории:

– зеленый – учащийся успешно справляется с заданиями, имеет положительные оценки по предметам, демонстрирует высокий уровень подготовки, проявляет активность на уроках и участвует во внеурочной деятельности,

– желтый – наблюдается средний уровень учебных достижений, при котором возможны отдельные трудности или пробелы в знаниях,

– оранжевый/красный – выявляются существенные проблемы с успеваемостью, которые могут быть связаны с низкой мотивацией, пропусками занятий или недостаточным вниманием к учебному процессу.

Применение тепловых карт позволяет учителю получить наглядное представление о сильных и слабых сторонах каждого учащегося. На основе такой визуализации педагог может скорректировать образовательный процесс и своевременно оказать дополнительную помощь. Для родителей данный формат также является полезным, поскольку он позволяет увидеть общую картину учебного прогресса ребёнка и при необходимости обсудить возникшие сложности с учителем.

1.3.2.3 Диаграмма и таблица прогресса

Диаграммы и таблицы прогресса успеваемости учащегося представляют собой средства анализа и наглядного отображения учебных данных. Их применение позволяет отслеживать изменения в образовательных результатах, выявлять динамику достижений, определять сильные и слабые стороны обучающегося, а также обозначать направления для дальнейшей педагогической поддержки.

Диаграмма прогресса – это графическое представление изменения оценок, баллов или уровня освоения учебного материала за определённый период времени, например, за четверть, семестр или учебный год.

К основным видам диаграмм, применяемых в отчётах об успеваемости, относятся:

– линейная диаграмма. Данный тип диаграммы отражает изменение показателей успеваемости учащегося на протяжении нескольких временных периодов. Горизонтальная ось, как правило, показывает месяцы, четверти или семестры, а вертикальная — уровень успеваемости в баллах или процентах. Такая форма представления данных позволяет быстро определить положительную или отрицательную динамику учебных результатов,

– столбчатая диаграмма. Она используется для отображения отдельных значений успеваемости по предметам или учебным периодам. Каждый столбец соответствует определённому предмету, четверти или полугодию, а его высота зависит от среднего балла либо итогового результата учащегося,

– круговая диаграмма. Этот вид диаграммы может применяться для сравнения относительных долей успешности по различным предметам в рамках одного выбранного периода.

Таблица прогресса успеваемости представляет собой упорядоченный набор сведений об учебных результатах, оформленный в табличной форме.

Она включает несколько колонок, каждая из которых отражает отдельную сторону учебной деятельности учащегося. Следует отметить, что таблицы позволяют рассматривать каждую категорию оценивания отдельно, поэтому они удобны для выявления конкретных проблем и зон роста.

В таблицу могут включаться следующие колонки:

– учебный период: четверть, полугодие, месяц,

– учебные предметы: математика, русский язык, история и другие дисциплины,

– итоговые показатели: средний балл, итоговая оценка, процент правильно выполненных заданий,

– дополнительные критерии: посещаемость, активность на занятиях, выполнение домашних заданий.

Каждая строка таблицы соответствует определённому временному интервалу или учебному циклу. В ней фиксируются фактические данные по отдельным предметам и показателям успеваемости.

Таким образом, диаграммы и таблицы являются удобным способом представления учебных данных. Они облегчают восприятие информации, позволяют быстро оценить прогресс или снижение результатов учащегося, а также дают учителям и родителям возможность более детально рассмотреть показатели по каждому предмету. Использование таких инструментов помогает своевременно выявлять возможные трудности и принимать меры для поддержки обучающегося.

1.3.3 Методы и алгоритмы для автоматизированной проверки письменных работ учащихся

Автоматизированная проверка письменных работ учащихся является сложной задачей, так как она включает не только распознавание рукописного текста, но и дальнейший анализ содержания работы. В отличие от проверки тестовых заданий, где достаточно сопоставить выбранный вариант ответа с правильным, письменная работа может содержать условие, промежуточные

вычисления, преобразования, пояснения и итоговый ответ. Поэтому система автоматизированной проверки должна уметь не только извлекать текстовую информацию из изображения, но и определять структуру решения, выделять отдельные задания, сравнивать решение учащегося с эталоном и формировать вывод о корректности выполнения.

Оптическое распознавание символов (Optical Character Recognition, OCR) представляет собой направление компьютерного зрения и искусственного интеллекта, связанное с преобразованием изображения печатного, машинописного или рукописного текста в машиночитаемый текстовый формат. Источником данных для OCR могут быть отсканированные документы, фотографии текстовых работ или документы в формате PDF. В результате работы OCR-системы изображение преобразуется в текст, который далее может быть сохранён, отредактирован, проанализирован или использован в других программных модулях.

Распознавание рукописного текста (Handwritten Text Recognition, HTR) является более сложной задачей по сравнению с распознаванием печатного текста. Это связано с индивидуальными особенностями почерка, различным наклоном букв, неодинаковым размером символов, качеством фотографии, наличием исправлений, зачеркиваний и неравномерного освещения. Поэтому задача автоматизированной проверки письменной работы не ограничивается простым переводом изображения в текст, а требует применения дополнительных методов анализа.

В зависимости от способа обработки исходных данных выделяют два основных вида распознавания рукописного текста:

– оффлайн-распознавание – обработка уже готового изображения текста, полученного с помощью сканера или камеры,

– онлайн-распознавание – обработка текста, вводимого с помощью графического планшета или сенсорного экрана, где дополнительно фиксируется не только форма букв, но и динамика движения руки.

В рамках автоматизированной проверки школьных письменных работ наиболее актуальным является оффлайн-распознавание, так как учитель может отправить фотографию уже выполненной работы. Для решения такой задачи используются классические OCR-методы, нейросетевые методы распознавания текста, мультимодальные модели, а также алгоритмы последующего анализа и сравнения решений.

1.3.3.1 Классические OCR-методы распознавания рукописного текста

Классические методы OCR основаны на предварительной обработке изображения, выделении символов или фрагментов текста и их последующем сопоставлении с известными образцами. Общая схема работы таких методов включает несколько этапов: улучшение качества изображения, бинаризацию, удаление шума, сегментацию текста, выделение признаков и классификацию символов. Подобный подход используется во многих OCR-системах, так как позволяет последовательно перейти от изображения документа к набору распознанных символов. Например, в архитектуре OCR-движка Tesseract выделяются этапы анализа страницы, поиска строк, выделения признаков и классификации символов.

Одним из простейших классических подходов является шаблонный метод. Его суть заключается в том, что каждый распознаваемый символ сравнивается с набором заранее подготовленных шаблонов. Наиболее подходящим считается тот шаблон, который имеет наименьшее отличие от анализируемого изображения. Данный метод может показывать достаточно хорошие результаты при распознавании печатного текста или символов, написанных в ограниченном числе вариантов. Однако для рукописного текста он имеет существенные ограничения. Почерк разных людей может сильно отличаться, поэтому один и тот же символ может иметь множество вариантов написания. Из-за этого системе необходимо хранить большое количество шаблонов, а точность распознавания всё равно может снижаться

при изменении наклона, толщины линии, размера символов или качества изображения.

Другим классическим подходом является структурный метод. В этом случае символ рассматривается не как набор пикселей, а как совокупность структурных признаков. К таким признакам могут относиться концы линий, пересечения, изгибы, петли, углы, расположение элементов относительно друг друга. После выделения признаков формируется описание символа, которое затем сравнивается с эталонными структурами. Достоинством такого подхода является большая устойчивость к некоторым изменениям начертания символов. Однако структурные методы также зависят от качества предварительной обработки изображения и могут давать ошибки при наличии исправлений, неразборчивого почерка или сложной записи математических выражений.

Классические OCR-методы имеют понятную логику работы и могут применяться для обработки документов с достаточно стабильной структурой. Однако при анализе школьных письменных работ их возможностей часто недостаточно. Это связано с тем, что необходимо распознать не только отдельные символы, но и смысловую структуру решения: номер задания, условие, последовательность вычислений и итоговый ответ.

1.3.3.2 Нейросетевые методы распознавания рукописного текста

С развитием методов машинного обучения для распознавания рукописного текста стали активно применяться нейронные сети. В отличие от шаблонных и структурных методов, нейросетевые алгоритмы не требуют ручного описания всех признаков символов. Модель обучается на большом количестве примеров и самостоятельно выделяет закономерности, необходимые для распознавания.

Одним из широко применяемых подходов к распознаванию рукописного текста является использование сверточных нейронных сетей (Convolutional Neural Network, CNN). На вход такой сети подаётся

изображение, после чего из него выделяются локальные визуальные признаки: линии, углы, контуры, отдельные фрагменты символов и их сочетания. Следует отметить, что данный подход хорошо подходит для анализа изображений, поскольку позволяет учитывать пространственное расположение элементов. Вместе с тем при распознавании текста важно не только определить признаки отдельных символов, но и сохранить порядок их следования.

Для обработки последовательностей используются рекуррентные нейронные сети (Recurrent Neural Network, RNN), а также их модификации, например, LSTM и BiLSTM. Такие сети позволяют учитывать расположение элементов в строке и анализировать зависимость текущего символа от предыдущих. В связи с этим в задачах распознавания рукописного текста часто применяются комбинированные архитектуры, объединяющие сверточные и рекуррентные слои.

Одним из примеров такой архитектуры является CRNN (Convolutional Recurrent Neural Network). В ней сверточная часть используется для извлечения визуальных признаков из изображения строки, а рекуррентная часть отвечает за обработку полученной последовательности признаков и формирование итоговой последовательности символов. Для обучения подобных моделей часто применяется метод Connectionist Temporal Classification (CTC). Его использование позволяет обучать модель на последовательностях без необходимости заранее задавать точное соответствие между отдельным символом и конкретным участком изображения. Это особенно важно при работе с рукописным текстом, поскольку границы между символами не всегда можно определить однозначно.

Нейросетевые методы обладают рядом преимуществ. Они более устойчивы к различиям в почерке, могут обрабатывать изображения разного качества и распознавать сложные последовательности символов. Однако следует учитывать, что такие методы требуют обучающих наборов данных,

вычислительных ресурсов и предварительной настройки модели. Кроме того, результат распознавания может содержать неточности, поэтому для практического использования необходима постобработка: исправление ошибок, проверка структуры результата, сопоставление с предметной областью и анализ корректности решения.

Таким образом, для проверки письменных работ учащихся нейросетевое распознавание является значимым, но не единственным этапом обработки. Даже при корректном распознавании текста необходимо определить, является ли решение верным, на каком этапе возникла ошибка и возможно ли начисление частичного балла. Поэтому после этапа распознавания требуется применение дополнительных алгоритмов анализа.

1.3.3.3 Мультимодальные модели для анализа изображений письменных работ

Отдельным направлением развития технологий распознавания и анализа документов являются мультимодальные модели. Их особенность заключается в способности обрабатывать разные типы данных, прежде всего текст и изображения, объединяя подходы компьютерного зрения и обработки естественного языка. В отличие от классических OCR-систем, такие модели не ограничиваются извлечением отдельных символов с изображения. Они могут анализировать структуру документа, выделять значимые фрагменты, определять взаимосвязи между элементами и представлять результат в заранее заданном формате.

В задачах анализа письменных работ мультимодальная модель получает на вход изображение работы и текстовую инструкцию, уточняющую, какие элементы должны быть распознаны. Например, модель может выделить номер задания, условие, последовательность шагов решения, итоговый ответ, а также фрагменты, распознанные с низкой степенью уверенности. Такой подход особенно полезен при обработке учебных

материалов, поскольку позволяет сохранить не только текстовое содержание, но и логическую структуру решения.

Преимуществом мультимодальных моделей является то, что они позволяют перейти от простого распознавания символов к анализу содержания изображения. Модель может воспринимать письменную работу как цельный документ и возвращать результат в структурированном виде, например, в формате JSON. Это облегчает дальнейшую программную обработку, так как результат можно передать в модуль сравнения решений, модуль оценивания или модуль формирования отчёта.

В то же время мультимодальные модели имеют ограничения. Точность их работы зависит от качества фотографии, освещения, читаемости почерка, наличия исправлений и сложности математической записи. Кроме того, модель может неверно распознать отдельные символы или пропустить часть решения. Поэтому результат работы такой модели не должен использоваться без дополнительной проверки.

Для системы, в которой требуется анализировать фотографии письменных работ учащихся, целесообразно сочетать мультимодальное распознавание с алгоритмами постобработки, нормализации и сравнения результата с эталоном. Мультимодальная модель выполняет этап извлечения структурированной информации из изображения, а дальнейшая проверка осуществляется алгоритмическими методами. Такой способ позволяет не обучать собственную OCR-модель с нуля, но при этом получить данные, пригодные для анализа решения.

1.3.3.4 Алгоритмы сравнения решения учащегося с эталонным решением

После этапа распознавания письменной работы возникает необходимость проверить корректность представленного решения. При этом простого сопоставления итоговых ответов недостаточно. Учащийся может получить неверный результат, сохранив при этом частично правильную

логику рассуждений, либо, наоборот, случайно прийти к верному ответу при наличии ошибок в промежуточных преобразованиях. В связи с этим автоматизированная проверка должна учитывать не только конечный ответ, но и последовательность выполненных шагов.

Первым этапом такого анализа является нормализация распознанного текста. Она позволяет привести ответы и промежуточные действия к единому формату. Например, в ходе нормализации могут удаляться лишние пробелы, унифицироваться обозначения переменных, заменяться похожие символы, а также приводиться к одному виду знак минуса. Такая обработка необходима для того, чтобы записи, одинаковые по смыслу, не рассматривались системой как различные только из-за особенностей оформления или неточностей распознавания.

Далее выполняется сравнение итогового ответа учащегося с эталонным вариантом. Если после нормализации ответы совпадают, решение может быть признано полностью верным. В случае несовпадения система должна перейти к анализу промежуточных шагов. Для этого сопоставляется последовательность действий, выполненных учащимся, с последовательностью действий, представленной в эталонном решении.

В процессе анализа хода решения важно учитывать частичную правильность выполнения задания. Если начальные этапы совпадают с эталоном, но ошибка возникает на более позднем шаге, учащемуся может быть начислена часть баллов. Такой подход ближе к традиционной педагогической проверке, поскольку учитель оценивает не только конечный результат, но и ход рассуждений. Кроме того, анализ промежуточных шагов позволяет определить характер ошибки: ошибку в знаке, арифметическую ошибку, ошибку переноса или преобразования, ошибку на позднем этапе решения либо логическое нарушение последовательности действий.

Результатом алгоритмического анализа становится определение статуса выполнения задания: полностью верно, частично верно или неверно. На основе количества правильно выполненных заданий и набранных баллов

может быть рассчитана рекомендуемая отметка. При этом результат автоматизированной проверки следует рассматривать как вспомогательный инструмент. Окончательное решение о выставлении оценки остаётся за учителем, поскольку качество распознавания и интерпретации решения может зависеть от особенностей конкретной письменной работы.

1.3.4 Методы и алгоритмы для автоматизированного составления учебных материалов

Автоматизированное формирование учебных материалов представляет собой одно из значимых направлений применения искусственного интеллекта в образовательной сфере. К таким материалам относятся планы уроков, конспекты занятий, проверочные и письменные работы, задания для самостоятельного выполнения, тестовые материалы, критерии оценивания, а также пояснения, предназначенные для использования педагогом. Следует отметить, что подготовка подобных материалов требует значительных временных ресурсов, поскольку учителю необходимо учитывать тему занятия, возраст и уровень подготовки учащихся, сложность изучаемого материала, структуру урока, планируемые образовательные результаты и соответствие содержания учебной программе.

В отличие от обычного поиска информации, автоматизированное составление учебных материалов связано с созданием нового текстового содержания на основе заранее заданных параметров. Пользователь формулирует запрос, указывает необходимые характеристики будущего материала и дополнительные требования к результату. После этого система формирует ответ в соответствии с определённой структурой. В контексте вышесказанного для решения подобных задач в современных интеллектуальных системах применяются методы обработки естественного языка, большие языковые модели, промт-инжиниринг, генерация с опорой на внешний материал, а также алгоритмы проверки, валидации и постобработки полученного результата.

Большие языковые модели позволяют генерировать связный текст на естественном языке, так как обучаются на больших корпусах текстовых данных и способны выявлять закономерности в употреблении слов, терминов и смысловых конструкций. В образовательной сфере такие модели могут использоваться для подготовки заданий, объяснений, планов уроков и учебных сценариев. При этом результат работы модели должен рассматриваться как вспомогательный материал, так как сгенерированный текст может требовать проверки, корректировки и адаптации со стороны педагога. Большие языковые модели обладают потенциалом для образовательных задач, но их использование требует контроля качества, так как возможны фактические ошибки, неточные формулировки и несоответствие учебным целям.

1.3.4.1 Обработка естественного языка

Обработка естественного языка (Natural Language Processing, NLP) представляет собой область искусственного интеллекта, направленную на анализ, понимание и генерацию текстов на языке человека. В задачах автоматизированного составления учебных материалов методы NLP используются для обработки входных запросов, анализа исходного текста, выделения ключевых понятий и формирования нового текстового содержания.

Технология обработки естественного языка для классификации текста подразделяется на несколько этапов: токенизация, удаление незначимых слов, нормализация, лемматизация, стемминг, выделение ключевых слов и анализ структуры текста. Применение данных методов позволяет привести исходный материал к более удобному для обработки виду и снизить влияние элементов, не несущих существенной смысловой нагрузки. Например, при загрузке учебного материала система может предварительно удалить лишние пробелы, номера страниц, колонтитулы и фрагменты, не относящиеся непосредственно к содержанию темы.

При автоматизированном составлении учебных материалов методы обработки естественного языка могут использоваться в нескольких направлениях:

– выделение темы, ключевых понятий и терминов из учебного текста,

– очистка и сокращение исходного материала,

– определение фрагментов, наиболее связанных с заданной темой,

– подготовка текста к передаче в языковую модель,

– преобразование результата генерации в структурированный формат.

Применение методов NLP особенно важно в случаях, когда учебный материал поступает в систему в виде файла. Такой документ может содержать таблицы, заголовки, номера страниц, служебные строки и элементы оформления. Перед использованием текста в качестве контекста его необходимо привести к более чистому и компактному виду. Таким образом, подобная предварительная обработка позволяет уменьшить объём нерелевантной информации и повысить качество последующей генерации.

1.3.4.2 Большие языковые модели и промт-инжиниринг

Большие языковые модели (Large Language Models, LLM) относятся к числу наиболее востребованных современных инструментов автоматизированной генерации текста. Они позволяют формировать связные ответы, пояснения, инструкции, учебные задания и другие материалы на естественном языке. В отличие от классических алгоритмов, основанных на заранее заданных шаблонах, LLM создают результат с учётом текстового запроса и контекста, предоставленного пользователем.

Следует отметить, что для получения корректного и управляемого результата важное значение имеет промт-инжиниринг. Промт представляет собой текстовую инструкцию, с помощью которой задаются условия генерации и требования к ответу модели. От качества такой инструкции

зависят структура, полнота и практическая ценность сформированного материала. При составлении учебных материалов в промпте целесообразно указывать назначение модели, предмет, класс, тему, тип материала, требования к стилю, структуре и ограничения. Например, при генерации плана урока в запрос могут быть включены длительность занятия, основные этапы урока, необходимость сформировать цель, задачи, планируемые результаты, перечень оборудования и формы работы.

Промт-инжиниринг позволяет управлять поведением языковой модели без дополнительного обучения. Это особенно важно при разработке прикладных систем, где требуется получить результат в заранее определённом формате. Для образовательных задач промт должен быть сформулирован так, чтобы модель не только сгенерировала текст, но и соблюдала педагогический стиль, не использовала случайные или неуместные материалы, придерживалась темы и учитывала возраст учащихся.

Большие языковые модели могут применяться для генерации:

– плана урока по предмету, классу и теме,
– письменной работы с выбранным типом заданий,
– тестовых заданий с вариантами ответа,
– заданий с развернутым ответом,
– пояснений и ответов для учителя,
– критериев оценивания.

Достоинством данного подхода является гибкость. Один и тот же программный механизм может использоваться для генерации разных типов учебных материалов, если изменяется промт и набор входных параметров. Недостатком является необходимость контроля результата. Языковая модель может сформировать некорректные, неполные или слишком общие ответы. Поэтому в практических системах результат генерации должен проходить программную проверку и, при необходимости, корректировку.

1.3.4.3 Генерация с опорой на учебный материал

Одним из важных подходов к автоматизированному составлению учебных материалов является генерация с опорой на внешний источник. В этом случае языковая модель использует не только собственные знания, полученные при обучении, но и дополнительный контекст, предоставленный пользователем. Таким контекстом может быть фрагмент учебника, параграф, методический материал, конспект или иной текст, относящийся к теме урока.

Подобный подход близок к идее retrieval-augmented generation (RAG), при которой генерация ответа дополняется информацией из внешних источников. Использование внешнего контекста позволяет повысить предметную точность ответа и лучше связать результат с конкретной учебной темой.

В образовательных задачах генерация с опорой на материал имеет несколько преимуществ. Во-первых, она позволяет формировать план урока или письменную работу на основе конкретного параграфа, а не только на основе общей информации по теме. Во-вторых, такой подход уменьшает вероятность появления содержания, не относящегося к изучаемому материалу. В-третьих, учитель получает результат, связанный с уже используемым учебником или методическим источником.

Для реализации такого подхода необходимо выполнить несколько этапов:

- получить файл с учебным материалом от пользователя,
- извлечь текст из документа,
- очистить текст от лишних элементов,
- при необходимости сократить текст до допустимого объёма,
- включить подготовленный фрагмент в промт,
- передать промт языковой модели,
- проверить и структурировать полученный результат.

При этом важно учитывать ограничения. Если исходный материал содержит сканы без текстового слоя, искажения, таблицы сложной структуры или фрагменты без контекста, качество генерации может снижаться. Кроме того, слишком большой объём текста может перегружать модель и приводить к менее точному ответу. Поэтому в практических системах необходимо предусмотреть ограничение длины контекста и предварительную очистку текста.

1.3.4.4 Структурированная генерация и постобработка результата

Для практического применения в программной системе недостаточно получить от языковой модели произвольный текст. Результат генерации должен быть представлен в таком виде, чтобы его можно было автоматически обработать, проверить и вывести пользователю. Поэтому одним из важных методов является структурированная генерация, при которой модель получает требование вернуть ответ в заранее заданном формате.

Наиболее удобным форматом для передачи структурированных данных является JSON. В таком формате можно задать поля, необходимые для конкретного типа учебного материала. Например, для плана урока могут быть предусмотрены поля: цель, задачи, планируемые результаты, оборудование, формы работы, этапы урока и длительность каждого этапа. Для письменной работы могут быть предусмотрены название, инструкция, список заданий, варианты ответов, правильные ответы, пояснения и критерии оценивания.

Структурированная генерация позволяет связать языковую модель с программной логикой системы. Вместо того чтобы обрабатывать произвольный текст, программа получает набор полей, которые можно проверить и преобразовать в сообщение для пользователя. Однако даже при наличии инструкции модель может вернуть неполный, некорректный или частично нарушенный JSON. Поэтому после генерации необходима постобработка результата.

Постобработка особенно важна в образовательных системах, так как результат должен быть понятным, методически корректным и пригодным для использования педагогом. Например, при генерации плана урока необходимо проверить, что сумма длительностей этапов соответствует общей продолжительности урока, а оборудование и формы работы выглядят реалистично. При генерации письменной работы необходимо проверить, что количество заданий соответствует запросу пользователя, а для тестовых заданий присутствуют варианты ответа.

1.4 Постановка задачи исследования

На основе проведённого анализа предметной области и существующих решений можно сделать вывод о необходимости разработки интеллектуальной системы, ориентированной на поддержку деятельности учителя.

Бот-помощник должен забирать на себя большую часть рутинных задач учителя, чтобы обеспечить разгрузку педагога и позволить ему сфокусироваться на более важных задачах, или задачах, требующих особого внимания. Интеллектуальный бот-помощник позволит педагогам в любое время, в любом месте просматривать нужную информацию и выполнять рабочие обязанности. Делегирование части задач боту-помощнику позволит учителям больше времени уделять ученикам, улучшать качество обучения, персонализировать учебный процесс.

Разрабатываемая система должна представлять собой Telegram-бота, так как данный формат взаимодействия обеспечивает доступность, простоту использования и возможность работы с системой без установки отдельного программного обеспечения. Такая система должна иметь преимущества перед использованием иных ресурсов и обеспечивать автоматизацию повторяющихся операций, связанных с работой с учебными данными, а также предоставлять интеллектуальные функции для подготовки учебных материалов и анализа письменных работ учащихся. Пользователь должен

иметь возможность обращаться к функциям бота через команды и проходить пошаговые сценарии ввода данных.

Функционал разрабатываемой системы целесообразно разделить на две составляющие: автоматизация рутинных задач, выполняемых учителем, и автоматизация интеллектуальной деятельности.

Сводная таблица рутинной деятельности учителя и соответствующего функционала бота-помощника учителя:

Таблица 1 – Список задач, относящихся к рутинной деятельности учителя

Деятельность учителя	Функционал бота-помощника
Просмотр расписания на неделю в рукописных заметках	Выдача расписания на неделю в электронном формате
Просмотр отметок учеников в журнале класса	Выдача всех отметок конкретного ученика класса в электронном формате
Расчет среднего и средневзвешенного балла по предмету вручную	Автоматический расчет среднего и средневзвешенного балла по всем предметам
Расчет итоговой оценки вручную	Автоматическое прогнозирование итоговой оценки
Подсчёт количества пропусков учащегося	Автоматический подсчет количества пропусков учащегося
Формирование сводки по успеваемости ученика	Автоматическая генерация отчета об успеваемости ученика
Отправка сводки по успеваемости родителю или законному представителю по электронной почте вручную	Автоматическая отправка отчета об успеваемости родителю или законному представителю по электронной почте

К базовому функционалу относятся функции, связанные с хранением, извлечением, обработкой и представлением учебных данных. Данные

функции не должны требовать генерации нового содержания с помощью искусственного интеллекта, и должны позволять автоматизировать рутинные задачи педагога.

Сводная таблица деятельности учителя и соответствующего интеллектуального функционала бота-помощника учителя:

Таблица 2 – Интеллектуальная деятельность учителя и соответствующий функционал бота-помощника

Деятельность учителя	Функционал бота-помощника
Подбор материала и составление плана урока	Генерация плана урока по заданным параметрам/заданному материалу и выдача его пользователю в электронном формате
Подбор материала и составление письменных работ вручную	Генерация письменных работ по заданной теме/загруженному материалу и параметрам и выдача его пользователю в электронном формате
Проверка письменных работ учеников вручную	Анализ письменных работ и выдача резюме по работе в электронном формате
Анализ ошибок ученика в письменной работе и расчет оценки за письменную работу	Автоматическое сравнение решения учащегося с эталонным решением, выявление частично верных решений и типовых ошибок, формирование рекомендуемой отметки за письменную работу

Интеллектуальный функционал системы основывается на применении предобученных моделей искусственного интеллекта, а также алгоритмов постобработки полученных результатов. Данные функции предназначены

для автоматизации методических и аналитических задач, выполнение которых связано с генерацией текста, обработкой изображений или интерпретацией решений учащихся.

Следует отметить, что при разработке интеллектуального бота-помощника учителя необходимо учитывать не только набор реализуемых функций, но и общие требования к качеству работы системы. В данном контексте особое значение имеют функциональность, сохранность данных, удобство использования и доступность.

К основным аспектам, которые необходимо учитывать при разработке интеллектуального бота-помощника, относятся:

– функциональность. Бот должен предоставлять пользователю необходимые инструменты для ввода, обработки и просмотра учебной информации,

– сохранность данных. Система должна обеспечивать корректное хранение введённых данных и их последующее отображение пользователю,

– удобство использования. Взаимодействие с ботом должно быть простым, понятным и не требующим длительного освоения. Система должна упрощать процесс ввода и получения информации, а не создавать дополнительную нагрузку для пользователя,

– доступность. Бот должен быть доступен для использования независимо от операционной системы мобильного устройства или персонального компьютера, времени и местонахождения пользователя.

2 Проектирование системы интеллектуального бота-помощника учителя

2.1 Функциональное назначение

Разрабатываемая система должна представлять собой интеллектуального бота-помощника учителя, предназначенного для для поддержки педагогической деятельности и автоматизации части рутинных, задач, связанных с организацией учебного процесса, анализом успеваемости и подготовкой учебных материалов.

Использование системы должно способствовать снижению временных затрат и нагрузки на педагога с целью оптимизации трудовой деятельности. Система должна помогать учителю быстрее получать сведения об учащихся, анализировать их успеваемость, формировать отчётные материалы, подготавливать задания и выполнять предварительный анализ письменных работ.

Функциональное назначение системы заключается в поддержке двух групп задач. Первая группа связана с автоматизацией рутинной деятельности учителя: просмотром расписания, просмотром оценок учащихся, расчётом показателей успеваемости, учётом посещаемости, формированием отчётов и передачей информации об успеваемости родителям или законным представителям. Вторая группа связана с поддержкой методических и аналитических задач: генерацией учебных материалов и анализом письменных решений учащихся с использованием предобученных моделей искусственного интеллекта.

Система должна быть ориентирована на использование в условиях, когда педагогу требуется быстро получить или сформировать учебную информацию без обращения к нескольким разрозненным инструментам. В рамках одного пользовательского сценария учитель должен иметь возможность задать необходимые параметры, получить результат обработки и использовать его в дальнейшей педагогической работе.

2.2 Требования к системе

2.2.1 Общие требования к системе

Пользователь должен взаимодействовать с системой посредством приложения Telegram. Разрабатываемая система должна быть реализована в виде чат-бота, принимающего команды, текстовые сообщения, документы и фотографии. Такой способ взаимодействия должен обеспечить пользователю доступ к функциям системы с персонального компьютера или мобильного устройства, на котором установлен Telegram.

Чат-бот должен запрашивать у пользователя сведения, необходимые для выполнения выбранного функционального сценария, и последовательно собирать их в процессе диалога. После получения требуемых данных они передаются в соответствующие модули серверной части, где выполняется обработка, после чего результат возвращается пользователю в виде сообщения или файла.

Следует отметить, что система должна поддерживать сценарный режим работы. Для функций, предполагающих поэтапный ввод нескольких параметров, необходимо предусмотреть сохранение промежуточного состояния диалога и переход между отдельными шагами взаимодействия. Такой подход позволяет корректно обрабатывать сложные пользовательские запросы, которые не могут быть выполнены на основе одного сообщения.

Серверная часть системы должна обеспечивать хранение и обработку учебных данных, выполнение запросов к базе данных, расчёт необходимых показателей, формирование отчётных материалов и обращение к интеллектуальным моделям. Для хранения структурированной учебной информации целесообразно использовать реляционную базу данных, содержащую сведения об учебной деятельности и участниках образовательного процесса.

Программная структура системы должна быть организована по модульному принципу. Такая организация программного кода должна

обеспечить разделение пользовательского взаимодействия, работы с базой данных, вычислительной логики и интеллектуальной обработки данных. Это должно упростить сопровождение системы и обеспечить возможность дальнейшего расширения функционала.

Для выполнения интеллектуальных функций в системе должно быть предусмотрено использование предобученных моделей искусственного интеллекта. Результаты работы моделей должны проходить дополнительную программную обработку: проверку структуры ответа, нормализацию данных, фильтрацию некорректных элементов и преобразование результата в формат, пригодный для отображения пользователю.

Система должна поддерживать информационный обмен между Telegram-ботом, серверной частью, базой данных и интеллектуальными модулями. Данные, полученные от пользователя, должны передаваться на обработку, а результаты анализа, генерации или выборки из базы данных должны возвращаться пользователю в виде текстовых сообщений или файлов.

В системе должна быть предусмотрена обработка ошибок. При отсутствии данных в базе, неправильном формате пользовательского ввода, невозможности извлечь текст из файла, ошибке генерации или ошибке анализа изображения бот должен информировать пользователя о возникшей проблеме в понятной форме.

2.2.2 Требования к функциональным характеристикам

Функциональные характеристики разрабатываемой системы должны включать две группы возможностей: базовый функционал и интеллектуальный функционал.

Базовый функционал должен быть связан с обработкой структурированных учебных данных, хранящихся в реляционной базе данных. Он должен обеспечивать автоматизацию рутинных организационно-аналитических задач учителя.

В системе должны быть предусмотрены следующие базовые функции:

– просмотр расписания учителя на неделю с группировкой занятий по дням,

– просмотр оценок выбранного ученика за заданный период,

– расчёт среднего балла ученика по предметам,

– расчёт средневзвешенного балла с учётом типа работы и веса оценки,

– определение прогнозируемой итоговой оценки на основе средневзвешенного балла,

– подсчёт количества пропусков ученика за выбранный период,

– формирование текстового отчёта об успеваемости ученика,

– отправка сформированного отчёта родителю или законному представителю по электронной почте.

Интеллектуальный функционал должен быть связан с генерацией, распознаванием и анализом образовательных материалов с использованием предобученных моделей искусственного интеллекта и алгоритмов постобработки.

В системе должны быть предусмотрены следующие интеллектуальные функции:

– генерация плана урока по предмету, классу и теме,

– генерация плана урока с опорой на загруженный учебный материал,

– генерация письменной работы по предмету, классу, теме и выбранному типу заданий,

– генерация письменной работы с опорой на загруженный учебный материал,

– формирование ответов, пояснений и критериев оценивания для сгенерированной письменной работы,

– анализ письменной работы учащегося с помощью vision-модели,

– определение типовых ошибок работы, анализ логики решения,

– формирование рекомендуемой отметки за письменную работу.

Для всех функциональных сценариев система должна формировать результат в понятном для пользователя виде. Если выполнение операции требует длительной обработки, бот должен уведомлять пользователя о начале соответствующего процесса.

2.2.3 Требования к организации входных и выходных данных

Входные данные системы должны включать команды пользователя, текстовые сообщения, документы, изображения, а также данные, хранящиеся в реляционной базе данных.

Команды пользователя должны использоваться для запуска отдельных функциональных сценариев Telegram-бота. Каждая команда должна определять характер взаимодействия пользователя с системой: получение информации из базы данных, формирование отчёта, генерация учебного материала или анализ письменной работы.

Текстовые входные данные должны использоваться для передачи параметров, необходимых для выполнения выбранного сценария. Такие данные должны вводиться пользователем в ходе диалога с ботом и обрабатываться системой в соответствии со сценарием. Система должна предусматривать проверку наличия обязательных параметров и корректность их формата.

Документы, загружаемые пользователем, должны использоваться как источник учебного материала для интеллектуальных функций системы. В системе должна быть предусмотрена возможность обработки файлов текстовых форматов: docx, pdf и txt.

Изображения должны использоваться в сценарии анализа письменных работ. Входные графические данные должны содержать читаемое изображение письменного решения, достаточное освещение и минимальные искажения перспективы. Качество изображения должно влиять на точность

распознавания и последующего анализа, поэтому система должна предусматривать информирование пользователя о требованиях к загружаемым фотографиям.

Данные, хранящиеся в реляционной базе данных, должны использоваться в функциональных сценариях, связанных с получением и анализом учебной информации. Структурированное хранение данных должно обеспечивать доступ к расписанию, сведениям об успеваемости и посещаемости учащихся, а также к контактной информации, необходимой для формирования и последующей отправки отчётных материалов.

Выходные данные системы должны включать текстовые сообщения, структурированную учебную информацию, файлы отчётов, результаты генерации учебных материалов и результаты анализа письменных работ.

Текстовые сообщения целесообразно использовать для отображения инструкций, уведомлений о ходе выполнения операций, сообщений об ошибках и результатов простых пользовательских запросов. При этом структурирование текста должно обеспечивать компактное и удобное для восприятия представление учебной информации.

Файлы отчётов должны применяться для представления аналитической информации в форме, пригодной для просмотра, хранения и передачи родителям или законным представителям учащихся. Содержание таких отчётов формируется на основе сведений из базы данных и дополнительных параметров, введённых пользователем в ходе взаимодействия с системой.

Результаты интеллектуальной генерации должны представлять собой структурированные учебные материалы, пригодные для дальнейшего использования педагогом. Такие материалы должны формироваться на русском языке, иметь логичную структуру и включать основные элементы, необходимые для соответствующего типа учебного материала.

Результаты анализа письменных работ должны представляться в виде текстовой сводки, содержащей общую оценку выполнения работы, комментарии по результатам анализа и сведения, необходимые педагогу для

принятия итогового решения. Система должна учитывать, что результаты автоматизированного анализа носят рекомендательный характер и могут требовать проверки со стороны учителя.

Все выходные данные, предназначенные для пользователя, должны быть представлены на русском языке, иметь понятную структуру и соответствовать выбранному функциональному сценарию. При невозможности сформировать результат система должна возвращать сообщение с описанием причины ошибки или указанием на некорректность входных данных.

2.3 Выбор методов и алгоритмов для реализации системы

Для достижения поставленных целей в разработке системы выбраны методы и алгоритмы, соответствующие функциональному назначению интеллектуального бота-помощника учителя. Выбор методов и алгоритмов осуществлялся на основе ранее проведенного анализа существующих решений, а также с учётом необходимости реализации как базового функционала, связанного с обработкой учебных данных, так и интеллектуального функционала, связанного с генерацией учебных материалов и анализом письменных работ учащихся.

Для расчета итоговой оценки учащегося выбрана комбинация метода простого среднего арифметического и метода средневзвешенной оценки. Простое среднее арифметическое будет использоваться для получения общего представления об уровне успеваемости ученика за выбранный период. Средневзвешенная оценка будет применяться для более точного анализа, так как позволит учитывать различную значимость отдельных видов работ.

В системе каждая запись об оценке будет содержать значение отметки, тип работы, вес, идентификатор ученика и идентификатор учебного предмета. При расчёте показателей успеваемости из базы данных будут извлечены все оценки выбранного ученика за указанный период. Далее для

каждого предмета будут рассчитываться средний балл, средневзвешенный балл, количество оценок и прогнозируемая итоговая оценка. Вес оценки может быть задан непосредственно в базе данных. Если он отсутствует, будет использоваться значение по умолчанию в зависимости от типа работы: домашнее задание, тест или контрольная работа. Такой подход позволит учитывать, что контрольная работа имеет большую значимость, чем обычная текущая отметка.

Для определения прогнозируемой итоговой оценки выбран алгоритм сопоставления средневзвешенного балла с пятибалльной шкалой. Если средневзвешенный балл составляет 4,5 и выше, прогнозируемая оценка будет приниматься равной «5»; если балл находится в диапазоне от 3,5 до 4,49 – «4»; от 2,5 до 3,49 – «3»; при меньшем значении – «2».

Для формирования сводки по оценкам учащегося выбран метод текстового отчёта. В рамках системы текстовый отчёт будет являться наиболее целесообразным способом представления данных, так как он будет легко восприниматься пользователем, может быть отправлен в сообщении Telegram, сохранён в PDF-файле или передан родителю по электронной почте. При формировании отчёта будет использоваться информация об ученике, его оценках по предметам, среднем и средневзвешенном балле, прогнозируемой итоговой оценке, количестве пропусков за выбранный период и комментарий учителя.

Для формирования отчётов в виде файлов будет выбран формат PDF. Данный формат удобен для просмотра, хранения и передачи законным представителям учащихся. Алгоритм формирования PDF-отчёта будет включать поиск ученика в базе данных, получение оценок, группировку оценок по предметам, расчёт показателей успеваемости, подсчёт пропусков и генерацию итогового документа.

Для отправки отчёта родителю или законному представителю выбран метод отправки электронного письма с вложением через SMTP. Такой способ позволит автоматизировать передачу сформированного PDF-файла без

необходимости вручную сохранять документ и отправлять его через почтовый клиент. В системе будет использоваться SMTP-сервер. Перед отправкой необходимо будет проверять наличие SMTP-настроек и существование PDF-файла, после чего отчёт будет прикреплен к письму как вложение.

Для автоматизированного составления учебных материалов выбран подход на основе больших языковых моделей. Данный метод используется для генерации плана урока и письменной работы по заданным параметрам. Для локального запуска языковой модели выбран инструмент Ollama, который позволяет загружать и использовать предобученные LLM на персональном компьютере без обращения к внешним облачным сервисам. В качестве входных данных будут применяться параметры предмета, класса и ряд дополнительных параметров. Для письменной работы дополнительно будет задаваться тип заданий и их характер. Такой подход выбран в связи с тем, что большие языковые модели способны формировать связный текст на естественном языке и адаптировать результат под заданную структуру.

Для управления результатом генерации выбран метод промт-инжиниринга. Для каждого сценария будет сформирован отдельный промт, в котором будет указана роль модели, входные данные, требования к содержанию, ограничения и структура итогового ответа. При генерации плана урока модель должна сформировать цель, задачи, планируемые результаты, оборудование, формы работы и этапы урока. При генерации письменной работы модель должна сформировать название, инструкцию, задания, ответы, пояснения и критерии оценивания.

Для обеспечения качества результата выбран метод постобработки и валидации ответа языковой модели. Это необходимо, так как LLM может вернуть неполный или частично некорректный ответ. В случае генерации плана урока будут проверяться задачи, планируемые результаты, оборудование, формы работы, этапы урока и длительность этапов.

Некорректные или нехарактерные элементы должны быть удалены, а при отсутствии необходимых данных будут использованы резервные значения. В случае генерации письменной работы будет проверяться количество заданий, тип заданий, наличие вариантов ответа, правильных ответов, пояснений и критериев оценивания. Если модель вернет недостаточное количество заданий, система должна дополнить результат резервными заданиями, чтобы итоговая структура соответствовала запросу пользователя.

Для генерации учебных материалов с опорой на внешний источник выбран метод предварительного извлечения и очистки текста из загруженного файла. Пользователь может предоставить учебный материал в формате .docx, .pdf или .txt. После загрузки файла система должна извлечь текст, очистить его от лишних пробелов, служебных строк, номеров страниц и других элементов, не относящихся к содержанию темы. При необходимости текст должен быть сокращен до установленного ограничения, чтобы не перегружать языковую модель. Подготовленный фрагмент должен быть включен в промт и использоваться как контекст для генерации плана урока или письменной работы.

Для автоматизированной проверки письменных работ учащихся выбран комбинированный подход, включающий анализ изображения с помощью мультимодальной модели и последующее алгоритмическое сравнение решения с эталоном. Для анализа изображений будет использована vision-модель, запускаемая локально через Ollama. Такой подход выбран вместо обучения собственной OCR-модели, так как он позволит использовать предобученную vision-модель для извлечения из фотографии не только отдельных символов, но и структуры решения: номера задания, условия, шагов решения, итогового ответа и неуверенно распознанных фрагментов.

На первом этапе анализа письменной работы будет использоваться мультимодальная модель, способная обрабатывать изображение и текстовую инструкцию. Фотография работы предварительно должна быть сжата,

преобразована в формат base64 и передана модели вместе с промтом. В промте должно быть задано требование выделить отдельные задания, распознать условие, шаги решения, итоговый ответ и сомнительные фрагменты. Результат должен быть представлен в структурированном формате JSON, что позволит передать его в следующий модуль обработки.

На втором этапе будет использоваться алгоритм сравнения решения учащегося с эталонным решением. Для этого распознанные данные эталонной работы и работы ученика должны быть приведены к единому виду. Далее выполняется нормализация итоговых ответов и шагов решения: удаляются лишние пробелы, унифицируются похожие символы, приводятся к единому виду знаки и переменные. После этого должны сравниваться итоговые ответы. Если ответ ученика совпадет с эталонным ответом, задание будет считаться полностью верным.

Если итоговый ответ не совпадет с эталоном, должен выполняться анализ промежуточных шагов решения. Для этого необходимо будет сравнить последовательность шагов ученика с последовательностью шагов эталонного решения. Если первые шаги совпадут, но далее обнаружится расхождение, решение может быть признано частично верным. Такой подход позволит учитывать не только конечный результат, но и ход рассуждений учащегося. Это важно для школьных письменных работ, так как ученик может допустить ошибку на позднем этапе, сохранив при этом правильную логику начала решения.

Для более детального анализа выбран алгоритм классификации типовых ошибок. В системе будут выделены такие виды ошибок, как ошибка в знаке, арифметическая ошибка, ошибка переноса или преобразования, ошибка на позднем этапе решения и логическая ошибка. Тип ошибки будет определяться на основе сравнения соответствующих шагов эталонного и ученического решения. Это позволит сформировать не только итоговый статус задания, но и комментарий, полезный для учителя при проверке работы.

Для формирования рекомендуемой отметки за письменную работу выбран балльный алгоритм. Каждое задание может быть признано полностью верным, частично верным или неверным. За полностью верное задание будет начисляться полный балл, за частично верное — часть балла, за неверное — ноль баллов. После обработки всех заданий необходимо рассчитать процент выполнения работы, на основании которого сформировать рекомендуемую отметку по пятибалльной шкале. Если процент выполнения составит 85% и выше, рекомендуется оценка «5»; от 65% до 84% — «4»; от 40% до 64% — «3»; при меньшем результате — «2».

2.4 Проектирование архитектуры системы

Архитектура Telegram-бота помощника учителя представлена в приложении В.1.

Архитектуру Telegram-приложения целесообразно построить по модульному принципу. Модульная архитектура должна обеспечить логическое разделение компонентов системы, упростить сопровождение программного кода и создать возможность для дальнейшего расширения функционала. В рамках такой архитектуры отдельные группы модулей должны отвечать за взаимодействие с пользователем, обработку пользовательских сценариев, работу с базой данных, выполнение расчётов, формирование отчётов, генерацию учебных материалов и анализ изображений письменных работ.

Проектируемая архитектура Telegram-бота должна включать три взаимосвязанных уровня: уровень пользовательского взаимодействия, уровень прикладной логики и уровень работы с данными и интеллектуальными сервисами.

Первый уровень представляет собой уровень пользовательского взаимодействия. Он должен быть реализован средствами Telegram-бота. На данном уровне система должна обеспечивать приём команд, текстовых сообщений, документов и фотографий, ведение диалога с пользователем,

запрос необходимых параметров и выдачу результата в форме сообщений или файлов. Данный уровень должен предоставлять пользователю единый интерфейс доступа ко всем функциям системы.

Второй уровень представляет собой уровень прикладной логики. На данном уровне должны располагаться модули, отвечающие за обработку пользовательских запросов, управление состояниями диалога и реализацию отдельных функциональных сценариев. К таким сценариям относятся получение расписания, просмотр оценок учащихся, расчёт показателей успеваемости, формирование отчётов, генерация плана урока, генерация письменной работы и анализ письменных работ по фотографиям. Уровень прикладной логики должен связывать Telegram-интерфейс с внутренними сервисами системы и определять, какие действия необходимо выполнить в ответ на конкретную команду пользователя.

Третий уровень представляет собой уровень работы с данными и интеллектуальными сервисами. На данном уровне должны располагаться модули, обеспечивающие доступ к реляционной базе данных, выполнение SQL-запросов, расчёт производных показателей, формирование PDF-отчётов, отправку отчётов по электронной почте, извлечение текста из загруженных учебных материалов, обращение к языковой модели и vision-модели, а также постобработку результатов их работы. Данный уровень должен обеспечивать выполнение основных вычислительных и интеллектуальных операций, результаты которых затем передаются на уровень прикладной логики и отображаются пользователю средствами Telegram-бота.

Взаимодействие уровней должно осуществляться последовательно. Пользователь инициирует сценарий через Telegram-бота, после чего уровень прикладной логики определяет необходимые действия и передаёт данные в соответствующие сервисные модули. Сервисные модули выполняют запросы к базе данных, расчёты, генерацию или анализ данных, после чего возвращают результат обработчику сценария. Далее результат преобразуется в форму, удобную для пользователя, и отправляется в Telegram.

3 Реализация интеллектуального бота-помощника учителя

3.1 Реализация базы данных и подготовка исходных данных

3.1.1 Формирование и генерация исходных данных

Для разработки системы был сформирован искусственный набор данных, моделирующий учебный процесс в общеобразовательной организации. Использование искусственных данных обусловлено отсутствием доступа к реальным персональным данным учащихся. При этом структура набора данных приближена к реальной образовательной системе и позволяет проверить работу основных функций бота.

В рамках набора данных рассматривается одна школа, в которой обучаются два учебных класса. Каждый класс включает учащихся с различным уровнем успеваемости и посещаемости. Также набор данных включает учебные предметы, по каждому из которых у учащихся имеются оценки. Для анализа посещаемости в базу данных добавлены сведения о присутствии или отсутствии учащихся на занятиях.

Для задачи отображения расписания учителя на неделю была сформирована модель расписания, приближенная к реальной: соблюдены условия по количеству рабочих дней, наличию метод-дня, количеству и длительности уроков. Также в ней учитываются номера уроков, время начала и окончания занятий, учебный предмет и класс.

Для автоматизации процесса создания набора данных был разработан python-скрипт, выполняющий последовательную генерацию всех необходимых таблиц. Скрипт реализует следующие одиннадцать этапов:

1) формирование перечня учебных классов,
2) формирование списка учащихся с уникальными идентификаторами и распределением по классам,
3) генерация перечня учебных предметов,
4) формирование данных об учителях,

5) формирование расписания учебных занятий,

6) формирование оценок учащихся по учебным предметам. Оценки генерируются случайным образом с использованием весов, имитирующих типичное распределение отметок,

7) формирование данных о посещаемости учащихся,

8) формирование данных о родителях или законных представителях учащихся,

9) формирование связей между учащимися и их родителями или законными представителями,

10) сохранение сгенерированных данных в формате CSV,

11) импорт подготовленных таблиц в базу данных SQLite для последующего использования в проекте.

3.1.2 Реализация базы данных SQLite

Для хранения исходных данных была реализована база данных SQLite, к которой обращаются сервисные модули системы.

Сформированная база данных включает следующие логические сущности, которые представлены в виде отдельных таблиц:

- данные об учащихся – таблица «student»,
- данные об оценках учащихся – таблица «mark»,
- данные о посещаемости занятий – таблица «attendance»,
- данные об учебном классе – таблица «class»,
- данные о законных представителях учащихся – таблица «parents»,
- данные о связи сущностей «ученик»-«родитель» – таблица «student_parent»,
- данные о расписании учебных занятий – таблица «teacher_schedule»,
- данные об учителе – таблица «teachers»,
- данные об учебных предметах – таблица «subject».

Работа с базой данных в системе организована через сервисные модули. Каждый модуль отвечает за отдельную группу операций: получение расписания, поиск ученика, получение оценок, расчёт успеваемости, группировку данных для отчёта и подсчёт пропусков. Такой подход позволяет отделить логику работы с базой данных от обработчиков Telegram-команд.

3.2 Разработка Telegram-бота и организация взаимодействия модулей

3.2.1 Создание Telegram-бота

Для реализации интеллектуального бота-помощника учителя было принято решение использовать платформу Telegram. Такой формат взаимодействия обеспечивает пользователю доступ к функциям системы с мобильного устройства или персонального компьютера без установки отдельного программного обеспечения. Взаимодействие с системой осуществляется через диалоговые команды, последовательные сценарии ввода данных и получение результата в виде сообщений или файлов.

Для создания Telegram-бота необходимо начать диалог с BotFather – официальным ботом Telegram, предназначенным для регистрации и настройки новых ботов. С помощью команды /newbot был создан бот-помощник учителя – techerAssistantAIBot (рис. 1).

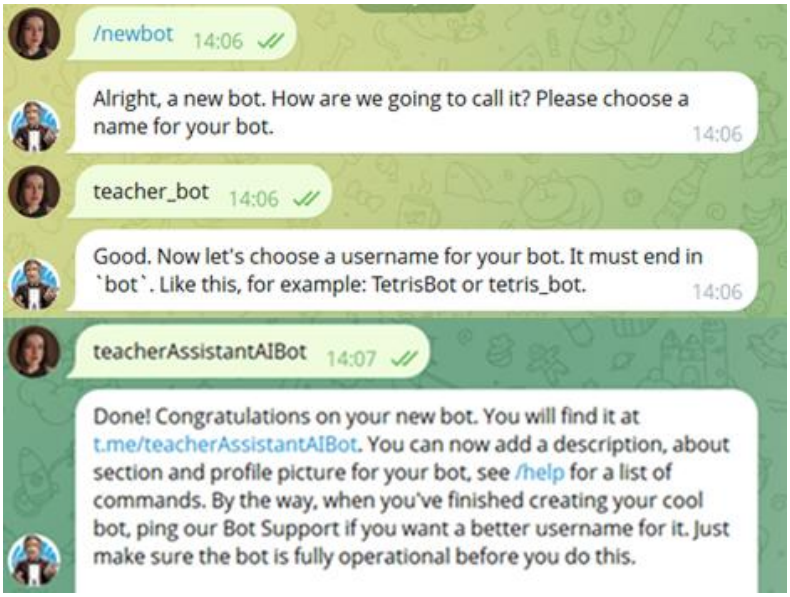


Рисунок 1 – Пример диалога с BotFather

Далее был получен токен, необходимый для дальнейшей разработки и запуска бота. Токен используется программой для подключения к Telegram Bot API. В разработанной системе он передаётся через файл окружения и считывается в конфигурационном модуле. Если токен не найден, запуск системы прерывается с сообщением об ошибке.

Для реализации программной части Telegram-бота была использована библиотека aiogram. Она позволяет обрабатывать команды пользователя, управлять состояниями диалога и отправлять пользователю текстовые сообщения, документы.

Главный файл программы отвечает за создание объекта бота, подключение обработчиков команд и запуск механизма обработки входящих сообщений от Telegram.

После подключения всех обработчиков для бота задаётся меню команд. В меню входят команды /start, /schedule, /marks, /final_grade, /report, /send_report, /lesson_plan, /check_work_photo, /written_work. Наличие такого меню упрощает взаимодействие пользователя с системой, так как основные функции можно выбрать в самом интерфейсе Telegram.

3.2.2 Организация взаимодействия Telegram-бота с базой данных и сервисными модулями

Схема зависимостей модулей Telegram-бота приведена в приложении В.2.

Организация взаимодействия Telegram-бота с внутренними компонентами системы строится на принципе функционального разделения. Telegram-обработчики выполняют роль внешнего интерфейсного слоя, отвечающего за получение сообщений пользователя, определение типа запроса, управление последовательностью диалога и передачу параметров в соответствующие сервисы. При этом сами обработчики не осуществляют

предметной обработки данных, а лишь инициируют вызов необходимого модуля.

В ходе разработки были выделены несколько основных групп сервисных модулей.

К первой группе относятся модули работы с учебными данными. Они обеспечивают извлечение, фильтрацию, группировку и интерпретацию информации, хранящейся в базе данных. Через эти модули реализуются сценарии просмотра расписания, получения оценок учащихся, анализа посещаемости, расчёта средних и средневзвешенных оценок, а также подготовки данных для формирования итоговых отчётов. Основной задачей таких модулей является выполнение запросов к реляционной базе данных и преобразование полученной информации в форму, пригодную для дальнейшего использования в Telegram-интерфейсе.

Ко второй группе относятся аналитические сервисные модули, предназначенные для вычисления производных показателей. Эти модули выполняют обобщение и интерпретацию информации. Например, на основе набора оценок ученика рассчитываются средние значения, средневзвешенные показатели, количество оценок и прогнозируемая итоговая оценка. Таким образом, эти сервисы выступают промежуточным звеном между набором исходных данных и их представлением пользователю в виде сводки или отчётного документа.

К третьей группе относятся модули формирования отчётных материалов. Они используются в сценариях, связанных с созданием PDF-отчёта об успеваемости ученика и отправкой такого отчёта родителю или законному представителю. В данном случае Telegram-бот получает от пользователя данные ученика, выбранный период и комментарий учителя, после чего передаёт параметры в сервис формирования отчёта. Сервис выполняет поиск ученика, получает необходимые учебные данные, рассчитывает показатели успеваемости и формирует PDF-файл. Далее

сформированный файл может быть отправлен пользователю в Telegram или передан в модуль отправки электронной почты.

К четвертой группе относятся модули интеллектуальной обработки, реализующие функции генерации плана урока, генерации письменной работы и анализа письменных работ учащихся. В первом случае Telegram-бот, получив от пользователя предмет, класс и тему урока, передаёт соответствующие параметры в сервис генерации, где выполняется обращение к языковой модели и последующая постобработка результата. Во втором случае бот дополнительно получает тип письменной работы и количество заданий, после чего передаёт параметры в сервис генерации письменной работы. При необходимости пользователь может загрузить учебный материал, который предварительно обрабатывается и используется как основа для генерации.

Отдельный интеллектуальный сценарий системы связан с анализом письменных работ учащихся. В данном случае бот обеспечивает загрузку двух изображений: фотографии эталонного решения и фотографии работы ученика. После этого полученные данные передаются в модуль vision-анализа, который выполняет распознавание структуры решения. Далее сформированный результат направляется в модуль сравнительного анализа, где определяются совпадения и расхождения между решениями, выявляются типовые ошибки, оценивается частичная правильность выполнения задания и формируется рекомендуемая отметка.

Взаимодействие Telegram-бота с сервисными модулями организовано в сценарном режиме. Такой подход позволяет поддерживать функции, выполнение которых требует последовательного ввода нескольких параметров. Например, для получения оценок конкретного учащегося пользователь сначала вводит фамилию, имя и класс ученика, затем выбирает интересующий период, после чего система выполняет выборку данных и отображает результат. При генерации плана урока также используется поэтапный ввод параметров: предмета, класса и темы занятия. Для генерации

письменной работы дополнительно выбирается тип работы и количество заданий. Таким образом, Telegram-бот не может ограничиться простой реакцией на отдельное сообщение, он должен учитывать контекст уже выполненных шагов и хранить промежуточное состояние диалога.

Telegram-бот функционирует как интеграционная оболочка, объединяющая несколько каналов обработки информации. Он не только принимает команды от пользователя, но и обеспечивает согласование между различными типами входных и выходных данных. Например, текстовая команда учителя может инициировать обращение к базе данных, результатом которого будет аналитическая сводка. В другом случае загрузка изображения приводит к запуску vision-модуля, а затем к аналитическому модулю сравнения решений. Загрузка документа может инициировать извлечение текста учебного материала и его последующее использование при генерации плана урока или письменной работы. Таким образом, Telegram-бот является координирующим компонентом, обеспечивающим взаимодействие между различными внутренними сервисами.

3.3 Реализация базового функционала системы

3.3.1 Просмотр расписания учителя

Одной из базовых функций системы является просмотр расписания учителя. Данная функция необходима для быстрого получения информации о занятиях на учебную неделю без обращения к бумажному расписанию. Получение расписания осуществляется через команду Telegram-бота /schedule.

При вводе пользователем команды /schedule запрос поступает в соответствующий обработчик Telegram-бота. Обработчик не выполняет прямую работу с базой данных, а обращается к сервисному модулю, отвечающему за получение расписания.

Информация о расписании хранится в базе данных в таблице teacher_schedule. В данной таблице содержатся сведения о дне недели,

номере урока, времени начала и окончания занятия, предмете и классе. Сервисный модуль выполняет SQL-запрос к базе данных и извлекает записи расписания. При получении данных используется сортировка по дням недели и номерам уроков, что позволяет отобразить расписание в удобной для пользователя форме.

После получения данных из базы обработчик группирует записи по дням недели. Далее для каждого дня формируется отдельный текстовый блок, в котором указываются номер урока, временной интервал, предмет и класс. Расписание отображается в виде структурированного текстового сообщения. Такой формат позволяет учителю быстро просмотреть всю необходимую информацию в интерфейсе Telegram.

Если данные о расписании отсутствуют, бот отправляет пользователю сообщение о том, что расписание не найдено.

3.3.2 Просмотр оценок ученика

Следующей базовой функцией системы является просмотр оценок ученика. Данная функция позволяет учителю получить сведения об успеваемости конкретного учащегося за выбранный период. Получение оценок осуществляется через команду Telegram-бота /marks.

При вводе команды /marks запрос поступает в соответствующий обработчик Telegram-бота. Далее бот переводит пользователя в сценарий последовательного ввода данных. Для получения информации об успеваемости необходимо указать фамилию, имя и класс учащегося в заданном формате. После получения этих данных система передаёт параметры в сервисный модуль, выполняющий поиск ученика в базе данных. Поиск осуществляется по ФИО и классу.

Если ученик найден, Telegram-бот сохраняет его идентификатор, ФИО и класс в промежуточном состоянии диалога и предлагает пользователю выбрать период для просмотра оценок. В системе предусмотрен выбор одной

из учебных четвертей, всего учебного года или произвольного периода, заданного двумя датами.

Информация об оценках хранится в базе данных в таблице marks. Для отображения названия учебного предмета данные из таблицы оценок соединяются с таблицей subjects. После выбора периода сервисный модуль выполняет SQL-запрос к базе данных и извлекает оценки выбранного ученика с учётом дат начала и окончания периода. Полученные записи сортируются по названию предмета и дате выставления оценки.

После получения данных обработчик формирует структурированный ответ для пользователя. Оценки группируются по предметам, а внутри каждого предмета отображаются дата, значение оценки и тип работы. Такой формат позволяет учителю быстро оценить успеваемость ученика по каждому учебному предмету и увидеть, за какие виды работ были выставлены отметки.

Если за выбранный период у ученика нет оценок, бот отправляет соответствующее сообщение. Если ученик не найден в базе данных, сценарий завершается, а пользователь получает уведомление о том, что учащийся с указанными данными отсутствует.

Данные об оценках также используются в дальнейших аналитических процессах: при расчёте среднего и средневзвешенного балла, формировании итогового отчёта по ученику и отправке отчётов об успеваемости родителям или законным представителям.

3.3.3 Расчёт среднего, средневзвешенного балла и посещаемости

Одной из базовых аналитических функций системы является расчёт среднего и средневзвешенного балла, а также учёт посещаемости учащегося. Данная функция позволяет учителю получить обобщённую информацию об успеваемости конкретного ученика за выбранный период. Вычисление средних показателей по предметам, прогнозируемой итоговой оценки и

данных о пропусках осуществляется через команду Telegram-бота /final_grade.

При вводе команды /final_grade запрос поступает в соответствующий обработчик Telegram-бота. Далее бот организует последовательный сценарий взаимодействия с пользователем. Сначала учителю необходимо ввести фамилию, имя и класс учащегося. После получения этих данных система передаёт параметры в сервисный модуль, который выполняет поиск ученика в базе данных. Если ученик найден, его идентификатор, ФИО и класс сохраняются в промежуточном состоянии диалога для дальнейшей обработки.

После определения ученика бот предлагает выбрать период для расчёта успеваемости. В системе предусмотрен выбор одной из учебных четвертей, всего учебного года или произвольного периода, заданного двумя датами. Такой подход позволяет выполнять расчёт не только для стандартных учебных периодов, но и для любого временного промежутка, необходимого учителю.

Информация для расчёта извлекается из базы данных. Оценки учащегося берутся из таблицы marks, а названия предметов — из таблицы subjects. Сервисный модуль получает все оценки выбранного ученика за указанный период и группирует их по учебным предметам. Для каждого предмета рассчитывается количество оценок, средний балл и средневзвешенный балл. При расчёте средневзвешенного балла учитывается вес оценки. Если вес не указан явно, используется значение по умолчанию в зависимости от типа работы: домашнее задание, тест или контрольная работа.

На основе средневзвешенного балла система также формирует прогнозируемую итоговую оценку по каждому предмету. Для этого используется шкала сопоставления: при средневзвешенном балле 4,5 и выше прогнозируется оценка «5», при значении от 3,5 — «4», при значении от 2,5 — «3», при меньшем значении — «2». Такой результат носит

рекомендательный характер и предназначен для предварительной оценки текущей успеваемости учащегося.

Для учёта посещаемости система дополнительно обращается к данным таблицы attendance. Сервисный модуль подсчитывает количество записей со статусом отсутствия за выбранный период. Полученное значение используется в итоговой сводке вместе с показателями успеваемости. Результат включает не только данные об оценках, но и информацию о пропусках учащегося.

После выполнения расчётов Telegram-бот формирует ответ в виде компактной текстовой сводки. В сообщении отображаются учебные предметы, средний балл, средневзвешенный балл, количество оценок и прогнозируемая итоговая оценка. Отдельно указывается общее количество пропусков за выбранный период.

Если за выбранный период у учащегося отсутствуют оценки, бот сообщает пользователю, что данных для расчёта нет, и завершает сценарий. Если ученик не найден в базе данных, пользователь также получает соответствующее уведомление.

3.3.4 Формирование и отправка отчётов

Следующей возможностью базового функционала является формирование PDF-отчётов об успеваемости ученика и их отправка родителям или законным представителям учащегося. Данная функция позволяет учителю не только получить аналитические данные об успеваемости, но и представить их в виде отдельного документа, пригодного для хранения и передачи.

Формирование файла-отчёта осуществляется через команду Telegram-бота /report. При вводе данной команды запрос поступает в соответствующий обработчик, после чего бот организует последовательный сценарий ввода данных. Пользователь указывает фамилию, имя и класс учащегося, выбирает период формирования отчёта и при необходимости вводит комментарий

учителя. Если комментарий не требуется, пользователь может указать специальный символ, после чего отчёт формируется без дополнительного текстового пояснения.

После получения входных данных обработчик передаёт параметры в сервис формирования отчёта. На данном этапе выполняется поиск ученика в базе данных, получение оценок по предметам, расчёт показателей успеваемости и подсчёт количества пропусков за выбранный период. Оценки группируются по учебным предметам, после чего на основе полученных данных формируется итоговый PDF-файл.

PDF-отчёт содержит основные сведения об ученике, класс, выбранный период, дату формирования документа, оценки по предметам, сводку по успеваемости, прогнозируемые итоговые оценки и данные о посещаемости. При наличии комментария учителя он также включается в содержание отчёта. Для генерации PDF-файла используется отдельный модуль, отвечающий за оформление документа, вывод текстовых блоков и таблиц.

Если ученик не найден в базе данных или за выбранный период отсутствуют данные для формирования отчёта, бот отправляет пользователю соответствующее сообщение. Также предусмотрена проверка факта создания PDF-файла. Если файл не был сформирован или не найден по указанному пути, пользователь получает уведомление о невозможности сформировать отчёт.

После успешного формирования отчёта бот отправляет PDF-файл пользователю в Telegram. В подписи к документу указывается ФИО ученика, класс и период отчёта. После чего учитель может сразу открыть документ, проверить его содержание и использовать его в дальнейшей работе.

Отправка отчёта родителю или законному представителю осуществляется через команду /send_report. Сценарий работы данной команды похож на сценарий формирования отчёта, однако дополнительно выполняется поиск родителя, связанного с выбранным учеником. Система определяет основной контакт родителя или законного представителя и

проверяет наличие адреса электронной почты. Если родитель не найден или email не указан, бот сообщает пользователю о невозможности отправки отчёта.

После выбора ученика, периода и ввода комментария учителя система формирует PDF-отчёт по тем же данным, что и в сценарии /report. Далее сформированный файл передаётся в сервис отправки электронной почты. Отправка осуществляется через SMTP-сервер, параметры которого задаются в файле окружения. Письмо содержит краткий текст с указанием ученика и периода отчёта, а PDF-файл прикрепляется к письму как вложение.

Если при отправке письма возникает ошибка, бот сообщает пользователю о проблеме. Если письмо успешно отправлено, пользователь получает уведомление о том, что отчёт был передан родителю или законному представителю. Таким образом, система автоматизирует не только подготовку отчётных материалов, но и их передачу заинтересованным участникам образовательного процесса.

3.4 Реализация интеллектуального функционала системы

3.4.1 Генерация плана урока с использованием LLM

Одной из интеллектуальных функций системы является генерация плана урока с использованием большой языковой модели. Данная функция позволяет учителю получить структурированный поурочный план на основе параметров, задаваемых в Telegram-боте. Генерация плана урока осуществляется через команду /lesson_plan.

При вводе команды /lesson_plan запрос поступает в соответствующий обработчик Telegram-бота. Далее бот организует последовательный сценарий взаимодействия с пользователем. На первом этапе пользователю предлагается выбрать способ генерации: свободная генерация по предмету, классу и теме или генерация с опорой на файл учебного материала. После выбора режима пользователь последовательно вводит название предмета,

класс и тему урока. Эти данные сохраняются в промежуточном состоянии диалога и затем передаются в сервисный модуль генерации плана урока.

В случае свободной генерации входными данными для языковой модели являются предмет, класс и тема урока. На основе этих параметров система формирует структурированный промпт, содержащий требования к результату генерации. В промпте задаётся роль модели, указываются входные данные, общая продолжительность урока, требования к стилю, состав обязательных разделов и формат итогового ответа. Модель должна сформировать план урока, включающий цель, задачи, планируемые результаты, оборудование, формы работы и последовательность этапов занятия.

Дополнительно в системе предусмотрен режим генерации с опорой на учебный материал. В этом случае после ввода предмета, класса и темы пользователь отправляет файл в формате .docx, .pdf или .txt. Загруженный файл сохраняется во временную директорию, после чего передаётся в модуль извлечения текста. Из документа извлекается текстовое содержимое, которое очищается от лишних пробелов, служебных строк, номеров страниц и других элементов, не относящихся к учебному содержанию. При необходимости текст сокращается до установленного ограничения, чтобы не перегружать языковую модель. После обработки файл удаляется, а извлечённый текст передаётся в сервис генерации как дополнительный источник информации.

В качестве инструмента генерации используется LLM-модель, развёрнутая через Ollama. Обращение к модели выполняется через HTTP API. Адрес API, название модели и режим использования LLM задаются в файле окружения. Такой подход позволяет запускать языковую модель локально и управлять параметрами её использования в программной части системы.

Сформированный промпт передаётся в языковую модель. Ответ модели должен быть представлен в формате JSON. Такой формат позволяет автоматически обработать результат и привести его к единой структуре.

После получения ответа выполняется программная обработка результата: система выделяет JSON-фрагмент, преобразует его в программную структуру данных, проверяет наличие обязательных блоков и корректность отдельных элементов. Если модель возвращает неполный или некорректный результат, используются резервные значения, позволяющие сохранить общую структуру плана урока.

В программной части были сохранены рамки генерации для повышения устойчивости результата и управляемости содержанием. Алгоритм задаёт общую продолжительность урока, контролирует допустимые формы работы, ограничивает структуру ответа и выполняет нормализацию полученного результата. В частности, проверяются оборудование, формы работы, названия этапов урока и длительность каждого этапа. Если сумма длительностей этапов не соответствует общей продолжительности урока, выполняется перераспределение времени. Также из результата удаляются нехарактерные или некорректные элементы, которые не соответствуют школьному уроку.

После завершения генерации и постобработки результат возвращается в обработчик Telegram-бота. Далее план урока форматируется в текстовое сообщение. В итоговом ответе отображаются предмет, класс, тема, тип урока, источник генерации, цель, задачи, планируемые результаты, оборудование, формы работы и ход урока с указанием этапов и их длительности. Полученный текст отправляется пользователю в Telegram.

3.4.2 Генерация письменной работы с использованием LLM

Следующей интеллектуальной функцией системы является генерация письменной работы с использованием большой языковой модели. Данная функция предназначена для автоматизированного формирования учебных заданий по заданным параметрам. Генерация письменной работы осуществляется через команду /written_work.

При вводе команды /written_work запрос поступает в соответствующий обработчик Telegram-бота. Далее бот организует последовательный сценарий взаимодействия с пользователем. На первом этапе пользователю предлагается выбрать способ генерации: свободная генерация по предмету, классу и теме или генерация с опорой на файл учебного материала. После выбора режима пользователь последовательно вводит название предмета, класс и тему письменной работы. Затем система предлагает выбрать тип письменной работы: задания с развернутым ответом, тест с вариантами ответа или смешанная работа. После этого пользователь указывает количество заданий от 3 до 15. Полученные данные сохраняются в промежуточном состоянии диалога и передаются в сервисный модуль генерации письменной работы.

В случае свободной генерации входными данными для языковой модели являются предмет, класс, тема, тип письменной работы и количество заданий. На основе этих параметров система формирует структурированный промпт, содержащий требования к результату генерации. В промпте задаётся назначение модели как помощника учителя, указываются входные данные, требования к содержанию, уровню сложности, типу заданий и формату итогового ответа. Модель должна сформировать письменную работу, включающую название, инструкцию для учащихся, перечень заданий, ответы, пояснения и критерии оценивания.

Дополнительно в системе предусмотрен режим генерации с опорой на учебный материал. В этом случае после ввода основных параметров пользователь отправляет файл в формате .docx, .pdf или .txt. Загруженный файл сохраняется во временную директорию, после чего передаётся в модуль извлечения текста. Из документа извлекается текстовое содержимое, которое очищается и подготавливается для передачи в языковую модель. После обработки временный файл удаляется, а извлечённый текст используется как дополнительный источник информации при формировании письменной работы.

Для генерации письменной работы используется LLM-модель, развёрнутая через Ollama. Обращение к модели выполняется через HTTP API. Адрес API, название модели и дополнительные параметры задаются в файле окружения. Так как генерация письменной работы может занимать больше времени, особенно при использовании загруженного учебного материала, в сервисном модуле предусмотрено отдельное значение тайм-аута для данного сценария.

Сформированный промпт передается в языковую модель. Ответ модели должен быть представлен в формате JSON, поскольку такой способ структурирования данных позволяет автоматически обработать полученную информацию и привести её к единому виду. В итоговой структуре должны содержаться основные элементы письменной работы: название, инструкция, перечень заданий, тип каждого задания, варианты ответа для тестовых вопросов, правильные ответы, пояснения и критерии оценивания. После получения ответа выполняется разбор JSON-структуры и проверка корректности сформированного результата.

Следует отметить, что в программной части предусмотрена постобработка результата генерации. Данный этап необходим для повышения устойчивости работы модуля, так как языковая модель может вернуть неполный или частично некорректный ответ. В процессе проверки уточняется тип каждого задания, наличие текста вопроса, правильного ответа, пояснения и количества баллов. Для тестовых заданий дополнительно контролируется наличие вариантов ответа A, B, C и D. Если отдельные элементы отсутствуют, система добавляет резервные значения, что позволяет сохранить общую структуру письменной работы.

Также выполняется контроль количества заданий. Если модель сформировала меньше заданий, чем было указано пользователем, результат дополняется резервными элементами. Если количество заданий превышает заданное значение, в итоговую работу включается только необходимая часть. Таким образом, обеспечивается соответствие результата параметрам,

заданным пользователем, и сохраняется целостность структуры итогового ответа.

После завершения генерации и постобработки результат возвращается в обработчик Telegram-бота. Далее письменная работа форматируется в текстовое сообщение. В итоговом ответе отображаются название работы, предмет, класс, тема, тип работы, источник генерации, инструкция, задания, количество баллов, ответы и пояснения для учителя, а также критерии оценивания. Если итоговый текст получается слишком длинным, он делится на несколько сообщений с учётом ограничений Telegram.

3.4.3 Анализ письменных работ учащихся на основе vision-модели

Одной из интеллектуальных функций системы является анализ письменных работ учащихся на основе vision-модели. Данная функция предназначена для предварительной проверки решения ученика по фотографии и сопоставления его с эталонным решением. Анализ письменных работ осуществляется через команду Telegram-бота /check_work_photo.

При вводе команды /check_work_photo запрос поступает в соответствующий обработчик Telegram-бота. Далее бот организует последовательный сценарий взаимодействия с пользователем. Сначала пользователю необходимо загрузить фотографию правильного решения, после чего система сохраняет путь к полученному изображению во временном состоянии диалога. Затем бот запрашивает фотографию письменной работы ученика. Таким образом, входными данными для анализа являются два изображения: эталонное решение и работа учащегося.

После получения каждой фотографии система выбирает подходящий вариант изображения из набора размеров, передаваемых Telegram, и сохраняет файл во временную директорию. Далее оба изображения передаются в сервисный модуль vision-анализа. После завершения обработки временные файлы удаляются, что позволяет не хранить загруженные фотографии дольше, чем это необходимо для выполнения сценария.

В качестве интеллектуального инструмента анализа используется vision-модель, развёрнутая через Ollama. В отличие от классического OCR-сценария, при котором изображение сначала преобразуется в текст, а затем отдельно интерпретируется, vision-модель позволяет воспринимать фотографию письменной работы как целостный источник информации. Это важно для анализа математических решений, поскольку значимы не только отдельные символы, но и общая структура записи, последовательность преобразований и взаимное расположение элементов.

Перед передачей изображения в модель выполняется его предварительная обработка. Файл открывается программными средствами, приводится к цветовому формату RGB, уменьшается до заданного максимального размера и сжимается в формате JPEG. После этого изображение кодируется в base64 и передаётся в запросе к vision-модели. Такой способ позволяет отправить изображение в модель через API Ollama вместе с текстовой инструкцией.

Для анализа изображения формируется промпт, в котором указывается, что модель должна проанализировать фото решения. В инструкции задаётся требование выделить отдельные задания, если их несколько, распознать номер задания, условие, шаги решения, итоговый ответ и сомнительные фрагменты. Отдельно указывается, что модель должна возвращать результат строго в формате JSON.

После получения ответа от vision-модели выполняется его программная обработка. Система пытается преобразовать ответ модели в JSON-структуру. Если в ответе присутствует дополнительный текст, из него выделяется JSON-фрагмент. Далее для каждого найденного задания выполняется нормализация структуры: приводятся к единому виду номер задания, подпись задания, текст условия, распознанные шаги решения, итоговый ответ и сомнительные фрагменты. Это позволяет согласовать ответ интеллектуальной модели с требованиями дальнейшего аналитического сравнения.

Интеллектуальный модуль предназначен не только для распознавания итогового ответа, но и для выделения структуры решения. В результате обработки изображения система формирует набор заданий, в котором для каждого задания определяются основные элементы: номер, условие, последовательность шагов, итоговый ответ и фрагменты, требующие дополнительного внимания. Эталонное решение и работа учащегося обрабатываются отдельно, после чего оба структурированных результата передаются в модуль сравнительного анализа для последующего сопоставления.

Качество анализа письменной работы во многом определяется характеристиками исходного изображения. Поэтому перед загрузкой фотографии бот предоставляет пользователю рекомендации по подготовке снимка: один лист должен полностью находиться в кадре, изображение должно быть хорошо освещено, снято под ровным углом, без сильных теней и размытия. Также желательно, чтобы задания были визуально отделены друг от друга. Соблюдение данных требований повышает вероятность корректного распознавания структуры решения и итоговых ответов.

Таким образом, использование vision-модели позволяет получить более содержательный результат по сравнению с классическим OCR-подходом. Система анализирует не только отдельные символы, но и общую структуру решения: номера заданий, последовательность преобразований, итоговый ответ и фрагменты, распознанные с низкой уверенностью. Вместе с тем качество анализа по-прежнему зависит от освещённости, ракурса съёмки, наличия теней, читаемости рукописного текста и сложности самой работы.

3.4.4 Механизм оценки качества решения и формирования рекомендуемой отметки

После анализа фотографий письменной работы с помощью vision-модели выполняется следующий этап обработки - оценка качества решения и формирование рекомендуемой отметки. Данный механизм предназначен для

сопоставления эталонного решения и решения учащегося, определения степени правильности выполнения заданий, выявления типовых ошибок и расчёта итогового результата работы.

Входными данными для данного этапа являются два структурированных набора данных: результат распознавания эталонного решения и результат распознавания работы ученика. Оба набора формируются на предыдущем этапе анализа изображений и содержат сведения об отдельных заданиях: номер задания, подпись задания, текст условия, распознанные шаги решения, итоговый ответ и сомнительные фрагменты. После получения этих данных обработчик команды /check_work_photo передаёт их в сервисный модуль сравнительного анализа.

На первом этапе выполняется сопоставление заданий из эталонного решения и работы ученика. Для этого используется номер задания или его подпись. Значения предварительно нормализуются: из них удаляются лишние обозначения, пробелы и символы, не влияющие на смысл номера задания. Такой подход позволяет сопоставить задания даже в том случае, если они были распознаны с небольшими отличиями в оформлении.

Далее для каждого распознанного задания проводится сравнение итогового ответа учащегося с эталонным вариантом. Перед выполнением сравнения ответы предварительно нормализуются: удаляются лишние пробелы, приводятся к единому виду знаки равенства и минуса, а также обозначения переменных. Такая обработка необходима для того, чтобы ответы, совпадающие по смыслу, не определялись системой как различные только из-за особенностей записи или неточностей распознавания. Если после нормализации ответ ученика совпадает с эталонным ответом, задание признаётся полностью верным.

Если итоговый ответ отличается от эталонного, система переходит к анализу промежуточных шагов решения. На данном этапе сопоставляются распознанные шаги эталонного решения и работы учащегося. Алгоритм определяет количество совпадающих начальных преобразований, а также

общую степень пересечения последовательности шагов. В контексте вышесказанного такой подход позволяет учитывать частичную правильность решения, поскольку учащийся может верно начать выполнение задания, но допустить ошибку на одном из последующих этапов.

При анализе промежуточных шагов система определяет, может ли решение быть зачтено частично. Если первые шаги решения совпадают с эталоном, но затем появляется расхождение, учащемуся может быть начислена часть балла. В текущей реализации за частично верное решение может быть начислено 0,5 балла или 0,25 балла в зависимости от степени совпадения решения с эталоном. Если последовательность шагов не выделена или решение расходится с эталоном с самого начала, задание признаётся неверным.

Дополнительно выполняется классификация возможной ошибки. Система анализирует первое расхождение между эталонным и ученическим решением и определяет тип ошибки. В программной реализации выделяются ошибка в знаке, арифметическая ошибка, ошибка переноса или преобразования, ошибка на позднем этапе решения и логическая ошибка. Если данных для уверенной классификации недостаточно, система фиксирует, что решение требует дополнительной проверки.

Для каждого задания формируется отдельный результат проверки. Он включает номер задания, текст эталонного задания и задания ученика, эталонный ответ, ответ ученика, признак совпадения итогового ответа, статус выполнения, начисленные баллы, тип ошибки и комментарии. Статус задания может принимать одно из трёх значений: полностью верно, частично верно или неверно. Такой формат позволяет учителю видеть не только итоговый вывод системы, но и основания, на которых он был сформирован.

После обработки всех заданий система рассчитывает общий результат работы. Для этого суммируются баллы, начисленные за отдельные задания, определяется количество полностью верных, частично верных и неверных заданий. Далее на основе отношения набранных баллов к общему количеству

заданий формируется рекомендуемая отметка по пятибалльной шкале. Если процент выполнения составляет 85% и выше, рекомендуется отметка «5»; при результате от 65% до 84% — «4»; при результате от 40% до 64% — «3»; при меньшем результате — «2».

После завершения сравнительного анализа результат возвращается в обработчик Telegram-бота. Далее бот формирует итоговое сообщение для пользователя. В сообщении отображается общее количество заданий, число полностью верных, частично верных и неверных решений, набранные баллы, рекомендуемая оценка и общий вывод. После этого по каждому заданию выводится более подробная информация: статус, баллы, тип ошибки, эталонный ответ, ответ ученика, совпадение итогового ответа, комментарии и распознанные шаги решения ученика.

Результат автоматизированной проверки носит рекомендательный характер. Это связано с тем, что качество анализа зависит от точности распознавания изображения, читаемости почерка, полноты эталонного решения и корректности выделения промежуточных шагов. Поэтому итоговая отметка, сформированная системой, должна рассматриваться как предварительная рекомендация для учителя, а не как полностью автоматическая замена педагогической проверки.

ЗАКЛЮЧЕНИЕ

В результате выполнения магистерской диссертации была разработана система «Интеллектуальный бот-помощник учителя», предназначенная для автоматизации части рутинных и интеллектуальных задач педагогической деятельности.

В ходе работы была изучена предметная область, связанная с применением интеллектуальных систем в образовании. Были рассмотрены особенности педагогической деятельности, существующие образовательные решения, а также методы и алгоритмы, применяемые для работы с учебными данными.

На основе проведённого анализа были определены требования к разрабатываемой системе, сформулированы основные функциональные возможности бота-помощника, спроектирована архитектура интеллектуального Telegram-бота и реализованы основные функциональные возможности.

Разработанная система позволяет работать с учебными данными, формировать отчёты об успеваемости, генерировать учебные материалы и выполнять предварительный анализ письменных работ учащихся с использованием предобученных моделей искусственного интеллекта.

Разработанный бот-помощник может использоваться как инструмент поддержки деятельности учителя. Он позволяет сократить время на выполнение ряда повторяющихся операций, упростить доступ к учебной информации, сформировать отчётные материалы и получить предварительную оценку письменной работы учащегося. При этом результаты интеллектуального анализа носят рекомендательный характер и должны рассматриваться как помощь педагогу, а не как полная замена профессиональной проверки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тельнова А. Д. Применение систем на базе искусственного интеллекта в сфере образования // Вестник науки. 2025. №6 (87).
2. Жилкишбаева Г.С., Мамедова А. М. Искусственный интеллект в дистанционном обучении: новые подходы и инструменты // Yessenov science journal, 2024.
3. Искусственный интеллект в образовании: Изменение темпов обучения. Аналитическая записка ИИТО ЮНЕСКО / Стивен Даггэн; ред. С.Ю. Князева; пер. с англ.: А.В. Паршакова. — Москва : Институт ЮНЕСКО по информационным технологиям в образовании, 2020.
4. Титова С. В. Технологические решения на базе искусственного интеллекта в обучении иностранным языкам: аналитический обзор // Вестн. Моск. ун-та. Сер. 19. Лингвистика и межкультурная коммуникация. 2024. Т. 27. № 2. С. 18–37.
5. Сбер Университет – Duolingo: как искусственный интеллект составляет персонализированное обучение языкам [Электронный ресурс] URL: https://sberuniversity.ru/edutech-club/pulse/keysy/34861/.
6. Blog Duolingo – Introducing Duolingo Max, a learning experience powered by GPT-4 [Электронный ресурс] URL: https://blog.duolingo.com/duolingo-max/.
7. Educators Technology – What is Eduaide AI and How to Use it In Your Teaching [Электронный ресурс] URL: https://www.educatorstechnology.com/2024/11/what-is-eduaide-ai-and-how-to-use-it-in-your-teaching.html.
8. Статистика успеваемости: ключевые показатели и методы анализа [Электронный ресурс] URL: https://sky.pro/wiki/analytics/statistika-uspevaemosti-klyuchevye-pokazateli-i-metody-analiza/.
9. Как считаются итоговые оценки в школе? [Электронный ресурс] URL: https://okddd.ru/news/kak-schitayutsya-itogovye-otsenki-v-shkole/.

10. К. А. Бобров, В. Д. Шульман, К. П. Власов Анализ технологий распознавания текста из изображения // Международный журнал гуманитарных и естественных наук, 2022г., №3-2.

11. Гайдуков Н.П., Савкова Е.О. Обзор методов распознавания рукописного текста // Международная научно-техническая конференция студентов, аспирантов и молодых учёных "Информационно-управляющие системы и компьютерный мониторинг – 2012".

12. Распознать почерк с помощью нейросети – цифровка текста, написанного от руки [Электронный ресурс] URL: https://vc.ru/ai/1981401-raspoznavanie-pocherka-s-pomoshchyu-neyrosetey.

13. NLP: что это такое и как она работает [Электронный ресурс] URL: https://skillbox.ru/media/code/nlp-chto-eto-takoe-i-kak-ona-rabotaet/.

14. Столбун Е. А., Полоско Е. И., Ефремов А.А. Искусственный интеллект и обработка естественного языка для классификации текстов // 59-я Научная Конференция Аспирантов, Магистрантов и Студентов БГУИР, Минск, 2023.

15. Модели машинного обучения: возможности и примеры использования [Электронный ресурс] URL: https://thecode.media/modeli-mashinnogo-obucheniya/.

16. Deep Learning (глубокое обучение): основы [Электронный ресурс] URL: https://timeweb.cloud/blog/deep-learning-osnovy.

17. Zheng Y., Huang S., Zeng X. Knowledge-enhanced large language models for automatic lesson plan generation. Humanities and Social Sciences Communications, 2025.

18. Снова о распознавании рукописного текста, на этот раз с помощью CRNN [Электронный ресурс] URL: https://habr.com/ru/articles/720614/.

19. AlKendi W., Gechter F., Heyberger L., Guyeux C. Advancements and Challenges in Handwritten Text Recognition: A Comprehensive Survey. Journal of imaging, 2024.

20. Hong-Yu Guo, Chuang Wang, Fei Yin, Xiao-Hui Li, Cheng-Lin Liu, Vision–language pre-training for graph-based handwritten mathematical expression recognition. Pattern Recognition, 2025.

21. Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, Haofen Wang Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv, 2023.

22. Все, что вам нужно знать о Ламе 3 | Самая мощная модель с открытым исходным кодом | Концепции использования [Электронный ресурс] URL: https://www.unite.ai/ru/все%2C-что-вам-нужно-знать-о-самой-мощной-модели-с-открытым-исходным-кодом-llama-3%2C-а-также-концепции-ее-использования/.

23. Shorten C., Pierse C., Smith T.B., Cardenas E., Sharma A., Trengrove J., van Luijt Weaviate B. StructuredRAG: JSON Response Formatting with Large Language Models. arXiv, 2024.

24. Кузнецов С.Д. Базы данных. Факультет ВМК МГУ, фонд «Вольное дело», teach-in лекции ученых МГУ, 2020.

25. Жао Элис. SQL. Pocket guide. 4-е изд. Пер. с англ. - Астана: «Спринт Бук», 2024.

26. SQLite Consortium. SQLite Documentation [Электронный ресурс] URL: https://www.sqlite.org/docs.html.

27. Bird S., Klein E., Loper E. Natural Language Processing with Python. O’Reilly Media, 2009.

28. Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО "Альфа-книга', 2018.

29. Daniel Jurafsky, James H. Martin. Speech and Language Processing. Third Edition draft, 2025.

30. Давлетов А. Р. Современные методы машинного обучения и технология OCR для автоматизации обработки документов //

Международный научный журнал «Вестник науки» №10 (67) Том 5. Октябрь 2023 г.

31. Ultralytics: Популярные модели OCR с открытым исходным кодом и принципы их работы [Электронный ресурс] URL: https://www.ultralytics.com/ru/blog/popular-open-source-ocr-models-and-how-they-work.

32. Skillbox Media. Telegram Bot API: полное руководство по созданию ботов [Электронный ресурс] URL: https://skillbox.ru/media/code/telegram-bot-api/#stk-3.

33. Ollama. Ollama's documentation [Электронный ресурс] URL: https://docs.ollama.com/.

34. Structured Outputs – Ollama [Электронный ресурс] URL: https://docs.ollama.com/capabilities/structured-outputs.

35. Pranjal Kumar. Large language models (LLMs): survey, technical frameworks, and future challenges. Artificial Intelligence Review, An International Science and Engineering Journal, 2024.

36. Qwen Team, Alibaba Group. Qwen2.5-VL Technical Report, 2025.

37. OpenAI ChatGPT [Электронный ресурс] URL: https://chatgpt.com. Промпты: «Подбери и кратко охарактеризуй современные образовательные сервисы, использующие искусственный интеллект для поддержки деятельности учителя», «Выполни поиск научных статей по применению искусственного интеллекта в образовании. Укажи основные идеи, методы и ограничения каждого источника. Сформируй подобранные статьи в список с указанием ссылки на источник», «Сократи предоставленный текст так, чтобы он сохранил основную мысль, но стал более компактным и соответствовал теме работы», «Проверь текст на логические повторы, стилистические неточности и избыточные формулировки. Предложи более академичную редакцию без изменения смысла», «Переведи статью с английского языка на русский. Выдели главную мысль статьи».

ПРИЛОЖЕНИЕ А

(обязательное)

Интеллектуальный бот-помощник учителя

Листинг программы

Приложение А.1 – Файл main.py

```
import asyncio
import logging

from aiogram import Bot, Dispatcher
from aiogram.types import BotCommand

from bot.config import BOT_TOKEN
from bot.handlers.start import router as start_router
from bot.handlers.schedule import router as schedule_router
from bot.handlers.marks import router as marks_router
from bot.handlers.final_grade import router as
final_grade_router
from bot.handlers.report import router as report_router
from bot.handlers.send_report import router as
send_report_router
from bot.handlers.lesson_plan import router as
lesson_plan_router
from bot.handlers.check_work_photo import router as
check_work_photo_router
from bot.handlers.written_work import router as
written_work_router

async def main():
    logging.basicConfig(level=logging.INFO)

    bot = Bot(token=BOT_TOKEN)
    dp = Dispatcher()

    dp.include_router(start_router)
    dp.include_router(schedule_router)
    dp.include_router(marks_router)
    dp.include_router(final_grade_router)
    dp.include_router(report_router)
    dp.include_router(send_report_router)
```

```
    dp.include_router(lesson_plan_router)
    dp.include_router(check_work_photo_router)
    dp.include_router(written_work_router)

    await bot.set_my_commands(
    [
        BotCommand(command="start", description="Запуск бота"),
        BotCommand(command="schedule", description="Показать
расписание"),
        BotCommand(command="marks", description="Показать оценки
ученика"),
        BotCommand(command="final_grade", description="Средний и
средневзвешенный балл"),
        BotCommand(command="report", description="Сформировать
PDF-отчет"),
        BotCommand(command="send_report", description="Отправить
отчет родителю"),
        BotCommand(command="lesson_plan",
description="Сформировать план урока"),
        BotCommand(command="check_work_photo",
description="Проверка работы по фото"),
        BotCommand(command="written_work",
description="Сформировать письменную работу"),
    ]
)

    print("Бот запущен...")
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```

Приложение А.2 – Файл marks.py

```
from aiogram import Router
from aiogram.filters import Command
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.context import FSMContext
from aiogram.types import Message

from bot.services.marks_service import find_student,
get_student_marks

router = Router()

PERIODS = {
    "1": {
        "label": "1 четверть",
        "date_from": "2024-09-01",
        "date_to": "2024-10-31", },
    "2": {
        "label": "2 четверть",
        "date_from": "2024-11-01",
        "date_to": "2024-12-29", },
    "3": {
        "label": "3 четверть",
        "date_from": "2025-01-09",
        "date_to": "2025-03-23", },
    "4": {
        "label": "4 четверть",
        "date_from": "2025-04-01",
        "date_to": "2025-05-31", },
    "5": {
        "label": "весь учебный год",
        "date_from": None,
        "date_to": None, }, }

class MarksSearchState(StatesGroup):
```

```
    waiting_for_student_data = State()
    waiting_for_period_choice = State()
    waiting_for_custom_period = State()

def get_clean_text(message: Message) -> str | None:
    if not message.text:
        return None

    text = message.text.strip()
    return text if text else None

def get_period_choice_text() -> str:
    return (
        "Выберите период для просмотра оценок:\n\n"
        "1 - 1 четверть\n"
        "2 - 2 четверть\n"
        "3 - 3 четверть\n"
        "4 - 4 четверть\n"
        "5 - весь учебный год\n"
        "6 - свой период\n\n"
        "Введите номер варианта." )

@router.message(Command("marks"))
async def cmd_marks(message: Message, state: FSMContext):
    await
state.set_state(MarksSearchState.waiting_for_student_data)
    await message.answer(
        "Введите данные ученика в формате:\n"
        "Фамилия Имя Класс\n\n"
        "Пример:\n"
        "Иванов Иван 7А" )

def format_subject_block(subject_name: str, rows: list) -> str:
    lines = []
    lines.append(f"{'Дата':<12} {'Оценка':<8} {'Тип работы'}")
```

```
    lines.append("-" * 36)

    for row in rows:
        date_str = str(row["date"])[:10]
        mark_str = str(row["mark"])
        work_type = str(row["work_type"])
        lines.append(f"{date_str:<12} {mark_str:<8}
{work_type}")

    table = "\n".join(lines)
    return f"<b>{subject_name}</b>\n<pre>{table}</pre>"

def format_marks(rows, student_name: str, class_name: str,
period_label: str) -> list[str]:
    if not rows:
        return [
            f"За выбранный период у ученика {student_name}
({class_name}) нет оценок." ]

    grouped = {}
    for row in rows:
        subject = row["subject_name"]
        grouped.setdefault(subject, []).append(row)

    messages = [
        f"<b>Оценки ученика:</b> {student_name}
({class_name})\n"
        f"<b>Период:</b> {period_label}" ]

    for subject, subject_rows in grouped.items():
        messages.append(format_subject_block(subject,
subject_rows))

    return messages
```

```
@router.message(MarksSearchState.waiting_for_student_data)
async def process_student_data(message: Message, state:
FSMContext):
    text = get_clean_text(message)

    if not text:
        await message.answer(
            "Пожалуйста, отправьте данные ученика текстом в
формате:\n"
            "Фамилия Имя Класс" )
        return

    parts = text.split()

    if len(parts) < 3:
        await message.answer(
            "Неверный формат.\n"
            "Введите данные так:\n"
            "Фамилия Имя Класс" )
        return

    class_name = parts[-1]
    full_name = " ".join(parts[:-1])

    student = find_student(full_name, class_name)

    if not student:
        await message.answer(
            f"Ученик '{full_name}' в классе '{class_name}' не
найден." )
        await state.clear()
        return

    await state.update_data(
        student_id=student["student_id"],
```

```
        full_name=student["full_name"],
        class_name=student["class_name"], )

    await
state.set_state(MarksSearchState.waiting_for_period_choice)
    await message.answer(get_period_choice_text())

@router.message(MarksSearchState.waiting_for_period_choice)
async def process_period_choice(message: Message, state:
FSMContext):
    text = get_clean_text(message)

    if not text:
        await message.answer("Пожалуйста, введите номер
периода.")
        return

    choice = text.strip()

    if choice == "6":
        await
state.set_state(MarksSearchState.waiting_for_custom_period)
        await message.answer(
            "Введите период в формате:\n"
            "ГГГГ-ММ-ДД ГГГГ-ММ-ДД\n\n"
            "Пример:\n"
            "2024-09-01 2024-10-31" )
        return

    period = PERIODS.get(choice)

    if not period:
        await message.answer(
            "Неверный вариант периода.\n\n"
            + get_period_choice_text() )
```

```
        return

    await send_marks_result(
        message=message,
        state=state,
        period_label=period["label"],
        date_from=period["date_from"],
        date_to=period["date_to"], )

@router.message(MarksSearchState.waiting_for_custom_period)
async def process_custom_period(message: Message, state:
FSMContext):
    text = get_clean_text(message)

    if not text:
        await message.answer(
            "Пожалуйста, введите период текстом в формате:\n"
            "ГГГГ-ММ-ДД ГГГГ-ММ-ДД" )
        return

    parts = text.split()

    if len(parts) != 2:
        await message.answer(
            "Неверный формат периода.\n"
            "Введите две даты через пробел:\n"
            "2024-09-01 2024-10-31" )
        return

    date_from, date_to = parts

    await send_marks_result(
        message=message,
        state=state,
        period_label=f"свой период: {date_from} — {date_to}",
```

```
        date_from=date_from,
        date_to=date_to, )

async def send_marks_result(
    message: Message,
    state: FSMContext,
    period_label: str,
    date_from: str | None,
    date_to: str | None, ):
    data = await state.get_data()

    student_id = data["student_id"]
    full_name = data["full_name"]
    class_name = data["class_name"]

    marks = get_student_marks(
        student_id,
        date_from=date_from,
        date_to=date_to, )

    messages = format_marks(
        marks,
        full_name,
        class_name,
        period_label, )

    for text in messages:
        await message.answer(text, parse_mode="HTML")

    await state.clear()
```

Приложение А.3 – Файл marks_service.py

```
from bot.services.db import get_connection

def find_student(full_name: str, class_name: str):
    conn = get_connection()
    cursor = conn.cursor()

    cursor.execute(
        """
        SELECT student_id, full_name, class_name
        FROM students
        WHERE lower(full_name) = lower(?) AND upper(class_name)
= upper(?)
        """,
        (full_name.strip(), class_name.strip()),
    )

    student = cursor.fetchone()
    conn.close()
    return student

def get_student_marks(
    student_id: int,
    date_from: str | None = None,
    date_to: str | None = None,
):
    conn = get_connection()
    cursor = conn.cursor()

    cursor.execute(
        """
        SELECT
            m.date,
            s.subject_name,
            m.mark,
```

```
        m.work_type
    FROM marks m
    JOIN subjects s ON m.subject_id = s.subject_id
    WHERE m.student_id = ?
      AND (? IS NULL OR m.date >= ?)
      AND (? IS NULL OR m.date <= ?)
    ORDER BY
        s.subject_name,
        m.date
    """,
    (
        student_id,
        date_from,
        date_from,
        date_to,
        date_to,
    ),
)

rows = cursor.fetchall()
conn.close()
return rows
```

Приложение А.4 – Файл check_work_photo.py

```
import asyncio
from pathlib import Path
from uuid import uuid4

from aiogram import Router
from aiogram.filters import Command
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from aiogram.types import Message, PhotoSize

from bot.services.vision_solution_service import
extract_solutions_from_image
from bot.services.work_check_service import
compare_solution_sets

router = Router()

class CheckWorkPhotoState(StatesGroup):
    waiting_for_reference_photo = State()
    waiting_for_student_photo = State()

UPLOAD_DIR = Path("uploaded_work_photos")
UPLOAD_DIR.mkdir(exist_ok=True)

def choose_reasonable_photo(photos: list[PhotoSize]) ->
PhotoSize:
    if not photos:
        raise ValueError("Список фото пуст.")

    if len(photos) == 1:
        return photos[0]

    if len(photos) == 2:
        return photos[0]
```

```
    return photos[-2]

async def download_telegram_photo(message: Message, photo:
PhotoSize, prefix: str) -> Path:
    file = await message.bot.get_file(photo.file_id)

    file_path = UPLOAD_DIR / f"{prefix}_{uuid4().hex}.jpg"
    await message.bot.download_file(file.file_path,
destination=file_path)

    return file_path

def safe_delete_file(file_path: str | None):
    if not file_path:
        return

    try:
        path = Path(file_path)
        if path.exists():
            path.unlink()
    except Exception:
        pass

@router.message(Command("check_work_photo"))
async def cmd_check_work_photo(message: Message, state:
FSMContext):
    await
state.set_state(CheckWorkPhotoState.waiting_for_reference_photo)
    await message.answer(
        "Отправьте фото правильного решения.\n\n"
        "Рекомендации:\n"
        "— один лист в кадре;\n"
        "— хороший свет;\n"
        "— ровный ракурс;\n"
```

```
        "— без сильных теней и размытия;\n"
        "— желательно, чтобы каждое задание было хорошо отделено
визуально." )

@router.message(CheckWorkPhotoState.waiting_for_reference_photo)
async def process_reference_photo(message: Message, state:
FSMContext):
    if not message.photo:
        await message.answer("Пожалуйста, отправьте именно фото
правильного решения.")
        return

    try:
        selected_photo = choose_reasonable_photo(message.photo)
        ref_path = await download_telegram_photo(message,
selected_photo, "ref")
    except Exception as e:
        await message.answer(
            "Не удалось получить фото правильного решения.\n"
            f"Техническая ошибка: {e}" )
        await state.clear()
        return

    await state.update_data(reference_photo_path=str(ref_path))
    await
state.set_state(CheckWorkPhotoState.waiting_for_student_photo)

    await message.answer("Фото правильного решения сохранено.
Теперь отправьте фото работы ученика.")

@router.message(CheckWorkPhotoState.waiting_for_student_photo)
async def process_student_photo(message: Message, state:
FSMContext):
    if not message.photo:
```

```
        await message.answer("Пожалуйста, отправьте именно фото
работы ученика.")
        return

    data = await state.get_data()
    ref_path = data.get("reference_photo_path")

    if not ref_path:
        await message.answer("Не найдено фото эталонного
решения. Запустите команду заново.")
        await state.clear()
        return

    try:
        selected_photo = choose_reasonable_photo(message.photo)
        student_path = await download_telegram_photo(message,
selected_photo, "student")
    except Exception as e:
        await message.answer(
            "Не удалось получить фото работы ученика.\n"
            f"Техническая ошибка: {e}" )
        await state.clear()
        return

    await message.answer("Проверяю работу. Анализ фото может
занять некоторое время.")

    try:
        reference_data = await asyncio.to_thread(
            extract_solutions_from_image,
            str(ref_path),
            "правильного решения", )

        student_data = await asyncio.to_thread(
            extract_solutions_from_image,
```

```
            str(student_path),
            "работы ученика", )

        result = compare_solution_sets(reference_data,
student_data)

    except Exception as e:
        safe_delete_file(str(ref_path))
        safe_delete_file(str(student_path))

        await message.answer(
            "Не удалось завершить анализ фото.\n"
            f"Техническая ошибка: {e}" )
        await state.clear()
        return

    lines = []
    lines.append("<b>Проверка письменной работы</b>")
    lines.append("")
    lines.append(f"<b>Всего заданий:</b>
{result.get('total_tasks', 0)}")
    lines.append(f"<b>Полностью верных:</b>
{result.get('correct_count', 0)}")
    lines.append(f"<b>Частично верных:</b>
{result.get('partial_count', 0)}")
    lines.append(f"<b>Неверных:</b> {result.get('wrong_count',
0)}")
    lines.append(f"<b>Баллы:</b> {result.get('earned_points',
0):.2f} / {result.get('total_tasks', 0)}")
    lines.append(f"<b>Рекомендуемая оценка:</b>
{result.get('recommended_grade', 'не определена')}")

    general_comments = result.get("general_comments", [])
    if general_comments:
        lines.append("")
```

```
        lines.append("<b>Общий вывод:</b>")
        for idx, comment in enumerate(general_comments,
start=1):
            lines.append(f"{idx}. {comment}")

    for item in result.get("results", []):
        lines.append("")
        label = item.get("task_label") or f"Задание
№{item.get('task_number', '?')}"
        lines.append(f"<b>{label}</b>")

        status = item.get("status", "wrong")
        if status == "full_correct":
            status_text = "полностью верно"
        elif status == "partial_correct":
            status_text = "частично верно"
        else:
            status_text = "неверно"
        lines.append(f"<b>Статус:</b> {status_text}")
        lines.append(f"<b>Баллы за задание:</b>
{item.get('points', 0):.2f}")

        error_type = item.get("error_type", "")
        if error_type == "sign_error":
            lines.append("<b>Тип ошибки:</b> ошибка в знаке")
        elif error_type == "arithmetic_error":
            lines.append("<b>Тип ошибки:</b> арифметическая
ошибка")
        elif error_type == "transfer_error":
            lines.append("<b>Тип ошибки:</b> ошибка переноса или
преобразования")
        elif error_type == "late_error":
            lines.append("<b>Тип ошибки:</b> ошибка на позднем
этапе решения")
        elif error_type == "logic_error":
```

```
            lines.append("<b>Тип ошибки:</b> нарушение логики
решения")

        if item.get("task_text_reference"):
            lines.append(f"<b>Эталон:</b>
{item['task_text_reference']}")

        if item.get("task_text_student") and
item["task_text_student"] != item.get("task_text_reference",
""):
            lines.append(f"<b>Распознано у ученика:</b>
{item['task_text_student']}")

        lines.append(f"<b>Эталонный ответ:</b>
{item.get('reference_answer') or 'не распознан'}")
        lines.append(f"<b>Ответ ученика:</b>
{item.get('student_answer') or 'не распознан'}")
        lines.append(
            f"<b>Совпадение итогового ответа:</b> "
            f"{'да' if item.get('answer_matches') else 'нет'}" )

        comments = item.get("comments", [])
        if comments:
            lines.append("<b>Комментарий:</b>")
            for idx, comment in enumerate(comments, start=1):
                lines.append(f"{idx}. {comment}")
        student_steps = item.get("student_steps", [])
        if student_steps:
            lines.append("<b>Шаги решения ученика:</b>")
            for idx, step in enumerate(student_steps, start=1):
                lines.append(f"{idx}. {step}")
    await message.answer("\n".join(lines), parse_mode="HTML")
    safe_delete_file(str(ref_path))
    safe_delete_file(str(student_path))
    await state.clear()
```

Приложение А.5 – Файл work_check_service.py

```
import re
from typing import Optional

def normalize_answer(text: str) -> str:
    return (
        str(text).lower()
        .replace(" ", "")
        .replace("=", "")
        .replace("x", "x")
        .replace("ё", "е")
        .replace("−", "-")
        .replace("–", "-")
        .strip())

def normalize_task_number(text: str) -> str:
    text = str(text).strip().lower()
    text = text.replace("задание", "").replace("№",
"").replace("n", "")
    text = re.sub(r"[^0-9a-za-я]", "", text)
    return text

def normalize_step(step: str) -> str:
    return (
        str(step).lower()
        .replace(" ", "")
        .replace("x", "x")
        .replace("ё", "е")
        .replace("−", "-")
        .replace("–", "-")
        .strip())

def index_works_by_number(works: list[dict]) -> dict[str, dict]:
    result = {}
```

```
    for work in works:
        task_number =
normalize_task_number(work.get("task_number", ""))
        if not task_number:
            task_label = work.get("task_label", "")
            task_number = normalize_task_number(task_label)

        if task_number:
            result[task_number] = work

    return result

def split_equation(step: str) -> tuple[Optional[str],
Optional[str]]:
    step_norm = normalize_step(step)
    if "=" not in step_norm:
        return None, None
    left, right = step_norm.split("=", 1)
    return left, right

def extract_signed_numbers(text: str) -> list[int]:
    return [int(x) for x in re.findall(r"(?<![a-za-я])[-
+]?\d+(?![a-za-я])", text)]

def calculate_prefix_match(reference_steps: list[str],
student_steps: list[str]) -> int:
    ref_norm = [normalize_step(x) for x in reference_steps if
str(x).strip()]
    student_norm = [normalize_step(x) for x in student_steps if
str(x).strip()]

    count = 0
    for ref_step, student_step in zip(ref_norm, student_norm):
        if ref_step == student_step:
            count += 1
```

```
        else:
            break
    return count

def calculate_step_overlap(reference_steps: list[str],
student_steps: list[str]) -> float:
    if not reference_steps or not student_steps:
        return 0.0

    ref_norm = [normalize_step(x) for x in reference_steps if
str(x).strip()]
    student_norm = [normalize_step(x) for x in student_steps if
str(x).strip()]

    if not ref_norm or not student_norm:
        return 0.0

    matches = 0
    used = set()

    for ref_step in ref_norm:
        for idx, student_step in enumerate(student_norm):
            if idx in used:
                continue
            if ref_step == student_step:
                matches += 1
                used.add(idx)
                break

    return matches / max(len(ref_norm), 1)

def detect_sign_error(ref_step: str, student_step: str) -> bool:
    ref_norm = normalize_step(ref_step)
    student_norm = normalize_step(student_step)
```

```
    if ref_norm == student_norm:
        return False

    ref_no_sign = ref_norm.replace("+", "").replace("-", "")
    student_no_sign = student_norm.replace("+", "").replace("-",
"")

    if ref_no_sign != student_no_sign:
        return False

    return ("+" in ref_norm or "-" in ref_norm) and ("+" in
student_norm or "-" in student_norm)

def detect_arithmetic_error(ref_step: str, student_step: str) ->
bool:
    ref_left, ref_right = split_equation(ref_step)
    st_left, st_right = split_equation(student_step)

    if None in (ref_left, ref_right, st_left, st_right):
        return False

    if ref_left != st_left and ref_right != st_right:
        return False

    ref_nums = extract_signed_numbers(ref_left + " " +
ref_right)
    st_nums = extract_signed_numbers(st_left + " " + st_right)

    if not ref_nums or not st_nums:
        return False

    if ref_left == st_left and ref_right != st_right:
        return True

    if ref_right == st_right and ref_left != st_left:
```

```
        return True

    return False

def detect_transfer_error(ref_step: str, student_step: str) ->
bool:
    ref_left, ref_right = split_equation(ref_step)
    st_left, st_right = split_equation(student_step)

    if None in (ref_left, ref_right, st_left, st_right):
        return False

    ref_all = sorted(extract_signed_numbers(ref_left + " " +
ref_right))
    st_all = sorted(extract_signed_numbers(st_left + " " +
st_right))

    if not ref_all or not st_all:
        return False

    if ref_all != st_all:
        return False

    if ref_left != st_left or ref_right != st_right:
        return True

    return False

def classify_first_error(reference_steps: list[str],
student_steps: list[str]) -> str:
    if not reference_steps or not student_steps:
        return "insufficient_data"

    ref_step = reference_steps[0]
    student_step = student_steps[0]
```

```
    if normalize_step(ref_step) == normalize_step(student_step):
        return "none"

    if detect_sign_error(ref_step, student_step):
        return "sign_error"

    if detect_arithmetic_error(ref_step, student_step):
        return "arithmetic_error"

    if detect_transfer_error(ref_step, student_step):
        return "transfer_error"

    return "logic_error"

def detect_partial_correctness(reference: dict, student: dict,
answer_matches: bool) -> tuple[bool, float, list[str], str]:
    comments = []

    if answer_matches:
        return False, 0.0, comments, "none"

    ref_steps = reference.get("recognized_steps", [])
    student_steps = student.get("recognized_steps", [])

    if not student_steps:
        comments.append("В решении ученика не удалось выделить
последовательность шагов.")
        return False, 0.0, comments, "no_solution"

    prefix_match = calculate_prefix_match(ref_steps,
student_steps)
    overlap = calculate_step_overlap(ref_steps, student_steps)
    first_error_type = classify_first_error(ref_steps,
student_steps)
```

```
    if prefix_match == 0:
        if first_error_type == "sign_error":
            comments.append("Ошибка в знаке возникает уже на
первом шаге решения.")
        elif first_error_type == "arithmetic_error":
            comments.append("Арифметическая ошибка возникает уже
на первом шаге решения.")
        elif first_error_type == "transfer_error":
            comments.append("Ошибка переноса или преобразования
возникает уже на первом шаге решения.")
        else:
            comments.append("Решение расходится с эталоном уже с
первого шага.")
        return False, 0.0, comments, first_error_type

    if prefix_match >= 2:
        comments.append("Первые шаги решения выполнены верно,
ошибка возникает на более позднем этапе.")
        return True, 0.5, comments, "late_error"

    if prefix_match == 1:
        if overlap >= 0.5:
            comments.append("Начало решения верное, но далее
допущена ошибка. Возможен частичный зачёт.")
            return True, 0.5, comments, "late_error"
        comments.append("Первый шаг решения верный, но дальше
логика решения нарушена.")
        return True, 0.25, comments, "late_error"

    return False, 0.0, comments, "logic_error"

def compare_single_work(reference: dict, student: dict) -> dict:
    ref_answer = reference.get("final_answer", "")
    student_answer = student.get("final_answer", "")
```

```
    ref_norm = normalize_answer(ref_answer)
    student_norm = normalize_answer(student_answer)

    answer_matches = bool(ref_norm and student_norm and ref_norm
== student_norm)

    comments = []

    if not ref_answer:
        comments.append("Не удалось уверенно распознать итоговый
ответ в эталонном решении.")

    if not student_answer:
        comments.append("Не удалось уверенно распознать итоговый
ответ в работе ученика.")

    if ref_answer and student_answer:
        if answer_matches:
            comments.append("Итоговый ответ ученика совпадает с
эталонным ответом.")
        else:
            comments.append("Итоговый ответ ученика не совпадает
с эталонным ответом.")

    ref_steps = reference.get("recognized_steps", [])
    student_steps = student.get("recognized_steps", [])

    if len(student_steps) < len(ref_steps) and student_steps:
        comments.append("Решение ученика выглядит более кратким,
чем эталонное.")
    elif len(student_steps) > len(ref_steps):
        comments.append("В решении ученика больше шагов, чем в
эталоне. Нужна ручная проверка преобразований.")
```

```
    if student.get("uncertain_fragments"):
        comments.append("В работе ученика есть неуверенно
распознанные фрагменты.")

    partial_correct, partial_points, partial_comments,
error_type = detect_partial_correctness(
        reference,
        student,
        answer_matches, )
    comments.extend(partial_comments)

    if answer_matches:
        status = "full_correct"
        points = 1.0
    elif partial_correct:
        status = "partial_correct"
        points = partial_points
    else:
        status = "wrong"
        points = 0.0

    return {
        "task_number": student.get("task_number") or
reference.get("task_number", ""),
        "task_label": student.get("task_label") or
reference.get("task_label", ""),
        "task_text_reference": reference.get("task_text", ""),
        "task_text_student": student.get("task_text", ""),
        "reference_answer": ref_answer,
        "student_answer": student_answer,
        "answer_matches": answer_matches,
        "partial_correct": partial_correct,
        "status": status,
        "points": points,
        "error_type": error_type,
```

```
        "comments": comments,
        "reference_steps": ref_steps,
        "student_steps": student_steps, }

def calculate_recommended_grade(earned_points: float,
total_tasks: int) -> str:
    if total_tasks <= 0:
        return "не определена"

    percent = (earned_points / total_tasks) * 100

    if percent >= 85:
        return "5"
    if percent >= 65:
        return "4"
    if percent >= 40:
        return "3"
    return "2"

def compare_solution_sets(reference_data: dict, student_data:
dict) -> dict:
    reference_works = reference_data.get("works", [])
    student_works = student_data.get("works", [])

    ref_index = index_works_by_number(reference_works)
    student_index = index_works_by_number(student_works)

    all_numbers = sorted(set(ref_index.keys()) |
set(student_index.keys()))

    results = []
    general_comments = []

    if not all_numbers:
        return {
```

```
            "results": [],
            "general_comments": [
                "Не удалось распознать структуру заданий ни в
эталонном решении, ни в работе ученика." ],
            "total_tasks": 0,
            "correct_count": 0,
            "partial_count": 0,
            "wrong_count": 0,
            "earned_points": 0.0,
            "recommended_grade": "не определена",}

    for number in all_numbers:
        ref_work = ref_index.get(number)
        student_work = student_index.get(number)

        if ref_work and not student_work:
            results.append(
                {
                    "task_number": number,
                    "task_label": ref_work.get("task_label",
f"Задание №{number}"),
                    "task_text_reference":
ref_work.get("task_text", ""),
                    "task_text_student": "",
                    "reference_answer":
ref_work.get("final_answer", ""),
                    "student_answer": "",
                    "answer_matches": False,
                    "partial_correct": False,
                    "status": "wrong",
                    "points": 0.0,
                    "error_type": "missing_student_solution",
                    "comments": ["В работе ученика не найдено
решение для этого задания."],
```

```
                    "reference_steps":
ref_work.get("recognized_steps", []),
                    "student_steps": [], } )
            continue

        if student_work and not ref_work:
            results.append(
                {
                    "task_number": number,
                    "task_label": student_work.get("task_label",
f"Задание №{number}"),
                    "task_text_reference": "",
                    "task_text_student":
student_work.get("task_text", ""),
                    "reference_answer": "",
                    "student_answer":
student_work.get("final_answer", ""),
                    "answer_matches": False,
                    "partial_correct": False,
                    "status": "wrong",
                    "points": 0.0,
                    "error_type": "missing_reference_solution",
                    "comments": ["В эталонном решении не найдено
соответствующее задание."],
                    "reference_steps": [],
                    "student_steps":
student_work.get("recognized_steps", []), }  )
            continue

        results.append(compare_single_work(ref_work,
student_work))

    correct_count = sum(1 for item in results if item["status"]
== "full_correct")
```

```
    partial_count = sum(1 for item in results if item["status"]
== "partial_correct")
    wrong_count = sum(1 for item in results if item["status"] ==
"wrong")
    earned_points = sum(item["points"] for item in results)
    total_tasks = len(results)

    recommended_grade =
calculate_recommended_grade(earned_points, total_tasks)

    general_comments.append(f"Полностью верно решено:
{correct_count} из {total_tasks}.")
    general_comments.append(f"Частично верно решено:
{partial_count} из {total_tasks}.")
    general_comments.append(f"Суммарный результат:
{earned_points:.2f} из {total_tasks}.")
    general_comments.append(f"Рекомендуемая оценка по текущей
шкале: {recommended_grade}.")

    return {
        "results": results,
        "general_comments": general_comments,
        "total_tasks": total_tasks,
        "correct_count": correct_count,
        "partial_count": partial_count,
        "wrong_count": wrong_count,
        "earned_points": earned_points,
        "recommended_grade": recommended_grade,
    }
```

Приложение А.6 – Файл vision_solution_service.py

```
import base64
import json
import os
from io import BytesIO
from pathlib import Path

import requests
from PIL import Image
from dotenv import load_dotenv

load_dotenv()

OLLAMA_URL = os.getenv("OLLAMA_CHAT_URL",
"http://localhost:11434/api/chat")
OLLAMA_VISION_MODEL = os.getenv("OLLAMA_VISION_MODEL",
"qwen2.5vl:7b")

def compress_image_to_base64(image_path: str, max_size: int =
1600, jpeg_quality: int = 80) -> str:
    path = Path(image_path)
    if not path.exists():
        raise FileNotFoundError(f"Файл не найден: {image_path}")

    with Image.open(path) as img:
        img = img.convert("RGB")
        img.thumbnail((max_size, max_size))

        buffer = BytesIO()
        img.save(buffer, format="JPEG", quality=jpeg_quality,
optimize=True)
        return base64.b64encode(buffer.getvalue()).decode("utf-
8")

def build_solution_extraction_prompt(role: str) -> str:
```

```
    return f"""
Ты анализируешь фото {role} по алгебре.

На фото может быть одно или несколько заданий.
Если заданий несколько, выделяй их отдельно.
Особенно ориентируйся на подписи вида:
- "Задание №1"
- "Задание №2"
- "Задание №3"

Требования:
1. Для каждого задания выдели номер задания.
2. Распознай условие задания, если оно видно.
3. Распознай шаги решения.
4. Выдели итоговый ответ.
5. Укажи сомнительные фрагменты, если текст плохо читается.
6. Не выдумывай недостающие части. Если что-то не видно, так и
напиши.
7. Если на фото только часть решения, честно укажи это.
8. Не пиши ничего вне JSON.

Верни строго JSON следующего вида:
{{
  "works": [
    {{
      "task_number": "строка",
      "task_label": "строка",
      "task_text": "строка",
      "recognized_steps": ["строка", "строка"],
      "final_answer": "строка",
      "uncertain_fragments": ["строка", "строка"]
    }}
  ]
}}
""".strip()
```

```
def _normalize_work_item(item: dict) -> dict:
    if not isinstance(item, dict):
        return {
            "task_number": "",
            "task_label": "",
            "task_text": "",
            "recognized_steps": [],
            "final_answer": "",
            "uncertain_fragments": [], }

    steps = item.get("recognized_steps", [])
    if not isinstance(steps, list):
        steps = []

    uncertain = item.get("uncertain_fragments", [])
    if not isinstance(uncertain, list):
        uncertain = []

    return {
        "task_number": str(item.get("task_number", "")).strip(),
        "task_label": str(item.get("task_label", "")).strip(),
        "task_text": str(item.get("task_text", "")).strip(),
        "recognized_steps": [str(x).strip() for x in steps if
str(x).strip()],
        "final_answer": str(item.get("final_answer",
"")).strip(),
        "uncertain_fragments": [str(x).strip() for x in
uncertain if str(x).strip()], }

def parse_json_response(content: str) -> dict:
    content = content.strip()

    try:
        return json.loads(content)
```

```
    except json.JSONDecodeError:
        pass

    start = content.find("{")
    end = content.rfind("}")

    if start == -1 or end == -1 or end <= start:
        raise RuntimeError(f"В ответе модели не найден JSON-
объект: {content[:800]}")

    json_part = content[start:end + 1]

    try:
        return json.loads(json_part)
    except json.JSONDecodeError as e:
        raise RuntimeError(f"Не удалось разобрать JSON-фрагмент
ответа модели: {json_part[:800]}") from e

def extract_solutions_from_image(image_path: str, role: str =
"решения") -> dict:
    image_b64 = compress_image_to_base64(image_path)
    prompt = build_solution_extraction_prompt(role)

    try:
        response = requests.post(
            OLLAMA_URL,
            json={
                "model": OLLAMA_VISION_MODEL,
                "stream": False,
                "format": "json",
                "messages": [
                    {
                        "role": "user",
                        "content": prompt,
                        "images": [image_b64],
```

```
                    } ],
                "options": {
                    "temperature": 0
                }, },
            timeout=420, )
        response.raise_for_status()
    except requests.exceptions.Timeout as e:
        raise RuntimeError(
            "Ollama не успела обработать фото. Попробуйте фото
меньшего размера "
            "или более лёгкую vision-модель."
        ) from e

    data = response.json()
    content = data["message"]["content"].strip()
    parsed = parse_json_response(content)

    works = parsed.get("works", [])
    if not isinstance(works, list):
        works = []

    normalized_works = [_normalize_work_item(item) for item in
works]

    return {
        "works": normalized_works
    }
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Интеллектуальный бот-помощник учителя

Презентация

Выпускная квалификационная работа на тему: “Интеллектуальный бот-помощник учителя”

Автор работы: студент группы 24ВВИм1 Тельнова А. Д.
Научный руководитель: к.т.н., доцент кафедры ВТ Митрохина Н.Ю.

Пенза, 2026

Актуальность исследования

Нагрузка на педагога включает не только проведение уроков, но и ведение документации, проверку работ, подготовку материалов и взаимодействие с родителями.

Искусственный интеллект позволяет частично автоматизировать повторяющиеся операции и поддержать методическую работу учителя.

Необходима система, объединяющая работу с учебными данными, генерацию материалов и предварительный анализ письменных работ.

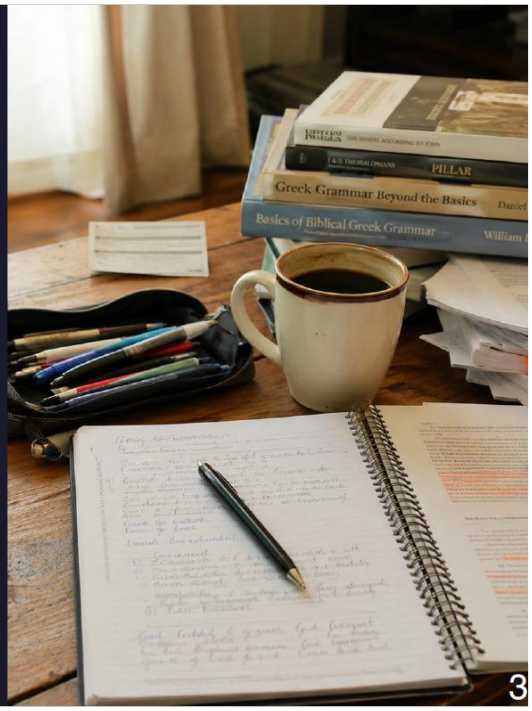
2

Цель, предмет и объект исследования

Цель исследования: исследование возможностей применения предобученных нейросетевых моделей в образовательной сфере и разработка интеллектуального Telegram-бота-помощника учителя.

Объект исследования: профессиональная деятельность педагога, включая административно-организационные процессы, реализуемые в рамках должностных обязанностей.

Предмет исследования: методы и программные средства автоматизации административно-организационной деятельности педагога.



3

Задачи исследования

01 **Анализ предметной области**

Изучение педагогической деятельности поможет идентифицировать рутинные задачи, выявить потребности учителей и определить области, требующие автоматизации через использование ИИ-решений.

02 **Проектирование архитектуры бота-помощника**

Создание архитектуры Telegram-бота, которая интегрирует все функции, обеспечит доступ к данным и позволит учителям легко взаимодействовать с системой.

03 **Реализация функционала системы**

Разработка алгоритмов для работы с учебными данными, интеграция LLM и vision-моделей для генерации учебных материалов, анализа данных и автоматизации рутинных процессов.

4

Аналоги и ограничения

Duolingo

Платформа использует ИИ для персонализации обучения, анализа результатов пользователя и предоставления обратной связи.

Eduaide.Ai

Eduaide.ai

Eduaide.ai - нейросетевой помощник для учителя: создание планов уроков, заданий, оценочных материалов, обратной связи и образовательного контента.

Ограничения:

- Ограничение функционала в бесплатной версии
- Высокая стоимость платной подписки
- Недоступность ресурса пользователям из России
- Узкая направленность

5

Стек технологий

SQLite

Ollama

6

Архитектура бота-помощника учителя
Пользователь
Telegram-бот
Сервисные модули
Данные и ИИ (SQLite, Ollama)
7

Интеллектуальные модули системы
Telegram-бот
Промт/ изображение
Ollama
LLM или Vision-модель
Постобработка
Результат
8

Базовый функционал системы

01 **Расписание**

Просмотр расписания учителя на любой период.

02 **Успеваемость**

Просмотр оценок учащихся, расчет итогового балла.

03 **Посещаемость**

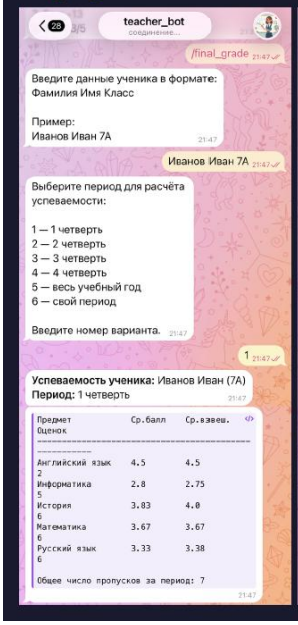
Учёт посещаемости с автоматизированной отчётностью.

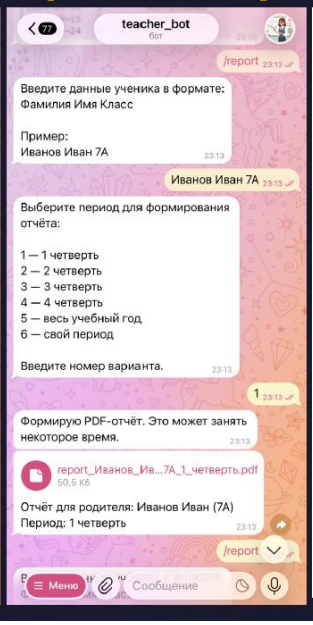
04 **Отчетность**

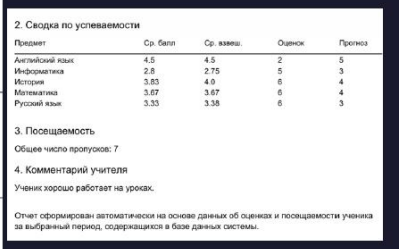
Формирование отчетности об успеваемости учащегося и последующая отправка отчета родителю/законному представителю.

9

Демонстрация работы приложения







10

Интеллекту-альный функционал системы

01 **Генерация планов уроков**

Создание планов уроков по заданным параметрам: генерация по материалу, свободная генерация.

02 **Генерация письменных работ**

Создание письменных работ по заданным параметрам: генерация по материалу, свободная генерация. Настройка типа работы, заданий.

03 **Анализ письменных работ**

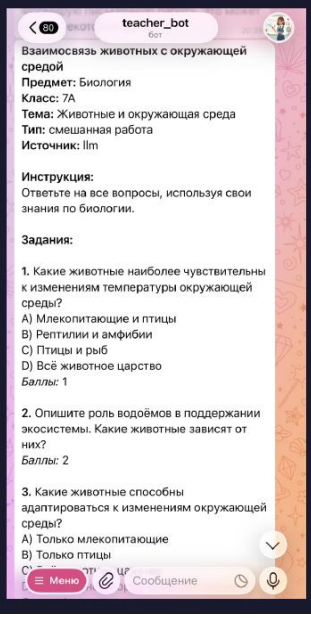
Распознавание и анализ письменных работ учащихся по фото.

04 **Рекомендации**

Рекомендация оценок за работу на основе анализа письменных заданий.

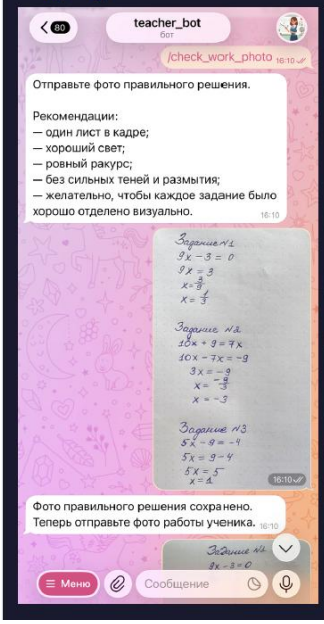
11

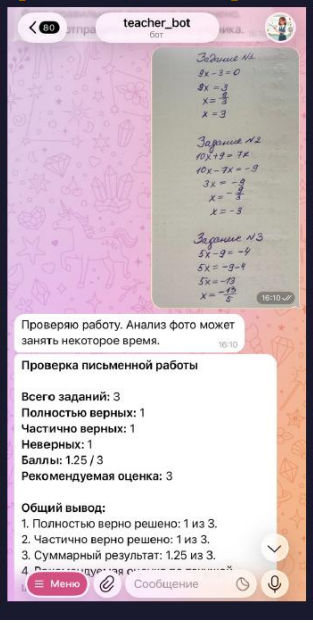
Демонстрация работы приложения

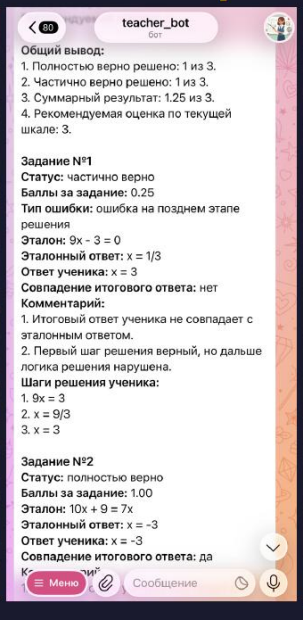


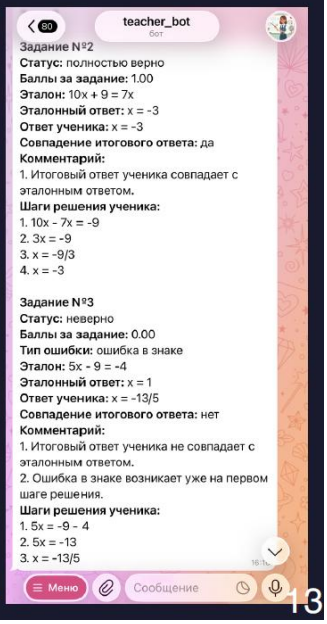
12

Демонстрация работы приложения









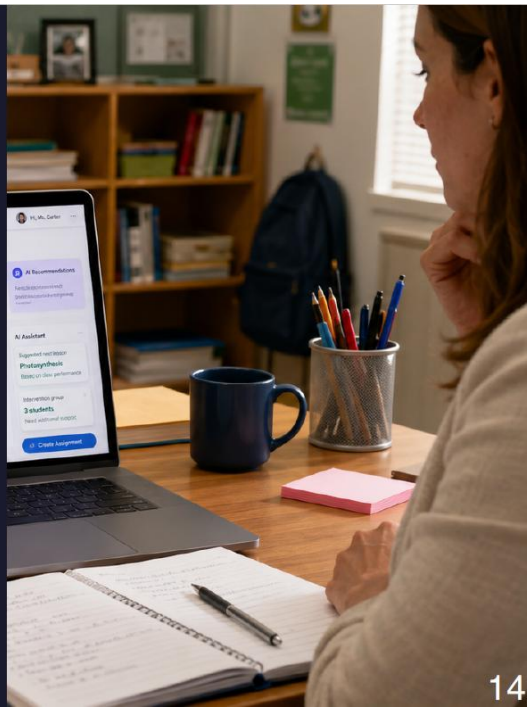
13

Заключение

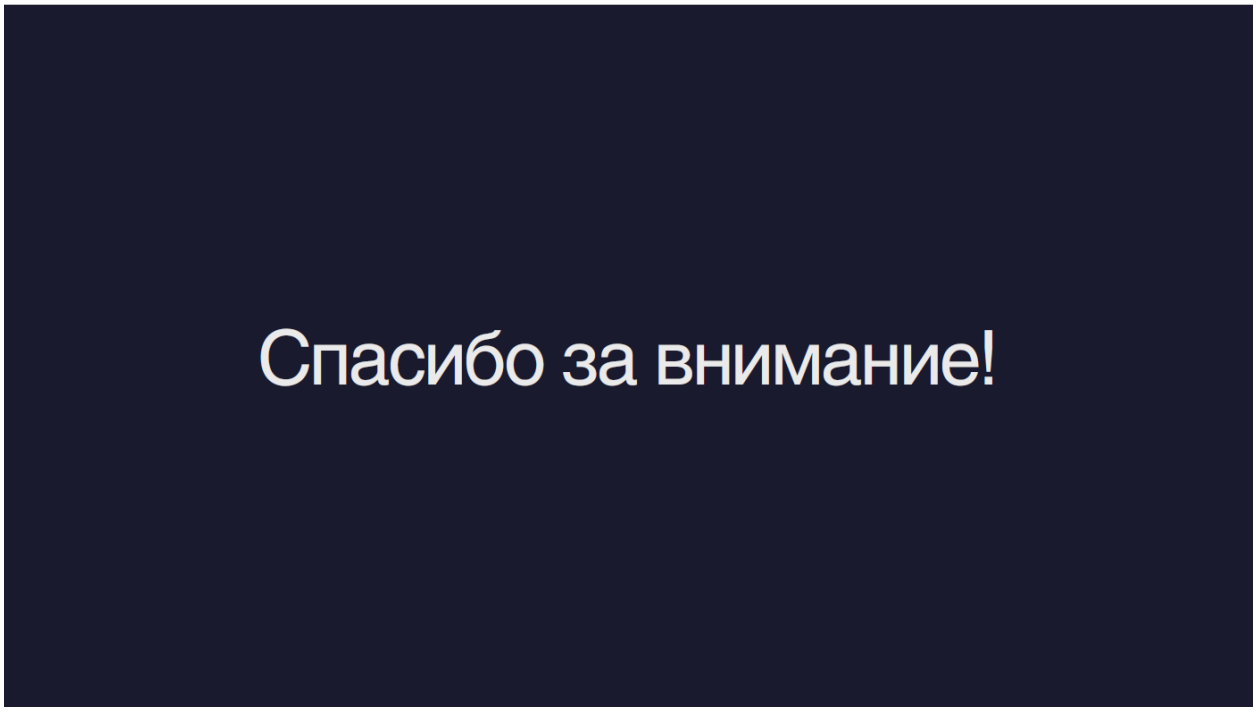
- Проведён анализ предметной области, существующих решений, методов и алгоритмов;
- Сформулированы требования к интеллектуальному боту-помощнику учителя;
- Спроектирована архитектура системы и подготовлена структура учебных данных;
- Реализован Telegram-бот с базовыми и интеллектуальными функциями.

Система может использоваться как инструмент поддержки учителя: для ускорения работы с учебными данными, подготовки материалов и предварительного анализа работ учащихся.

Опубликована статья “Применение систем на базе искусственного интеллекта в сфере образования” // Вестник науки, 2025.



14


Спасибо за внимание!

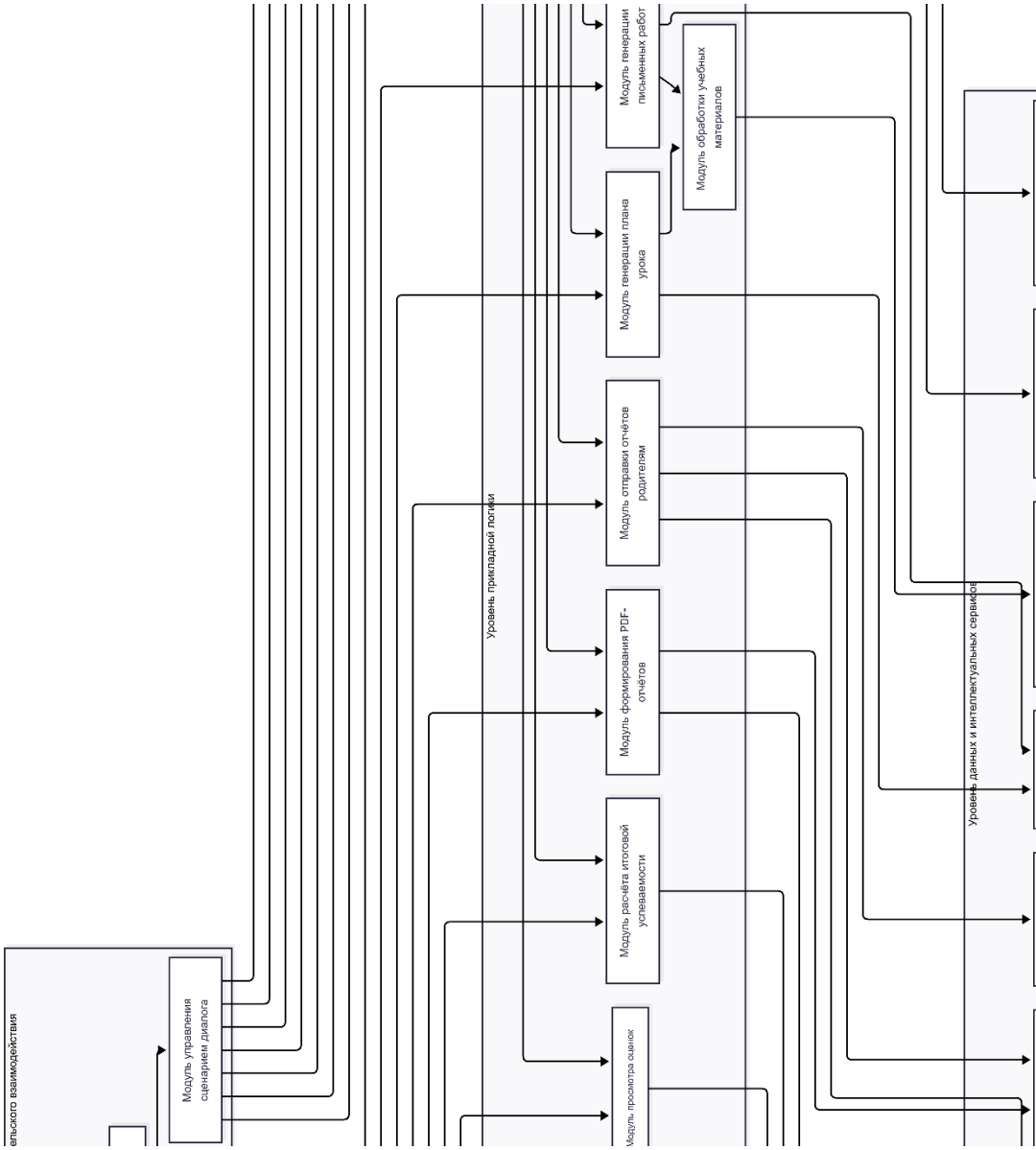
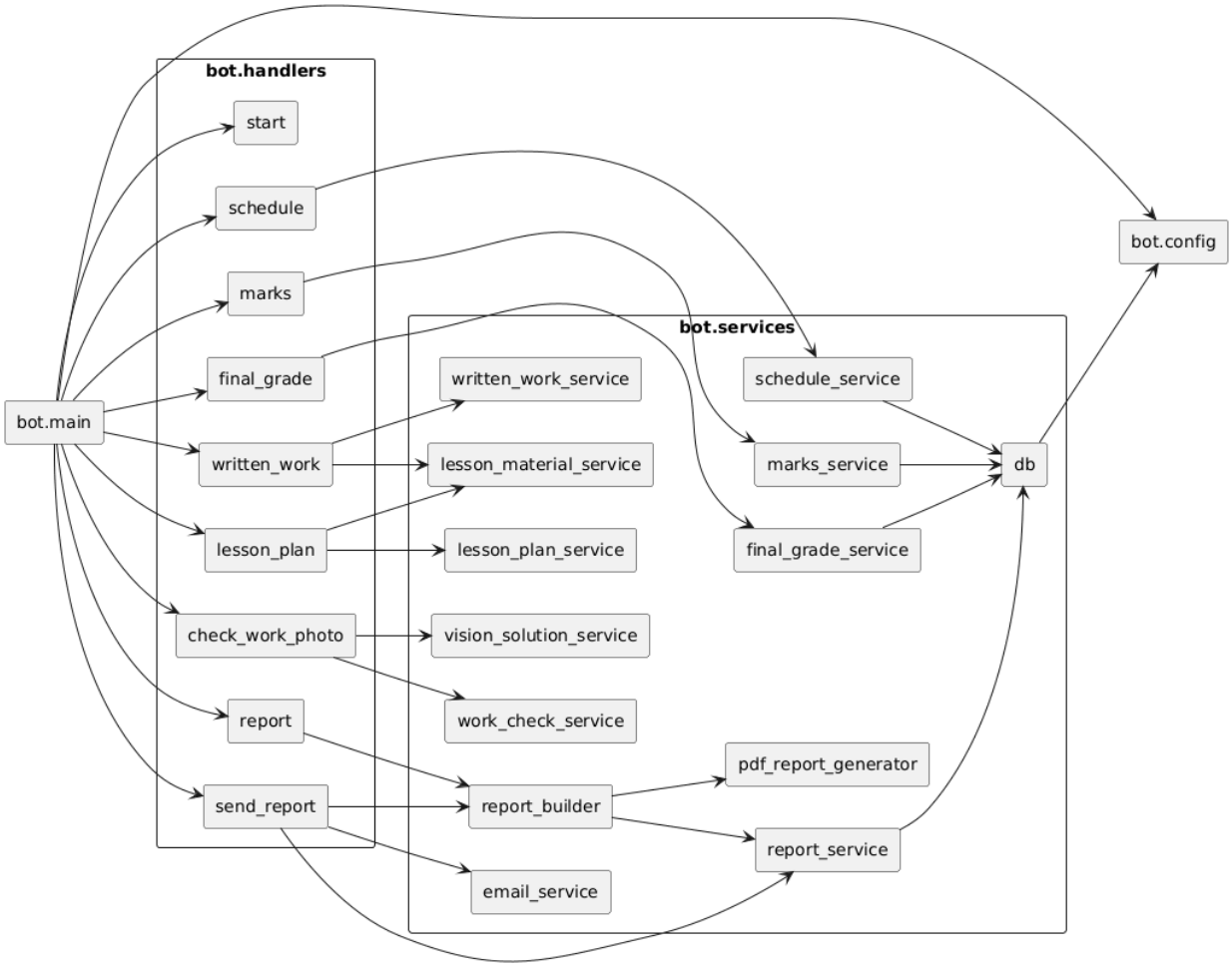
ПРИЛОЖЕНИЕ В

(рекомендуемое)

Интеллектуальный бот-помощник учителя

Схемы

Приложение В.1 – Архитектура Telegram-бота помощника учителя



Приложение В.2 – Схема зависимостей модулей Telegram-бота