**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

МОСКОВСКОЙ ОБЛАСТИ

**«Ногинский колледж»**

**Дипломная работа**

**По теме: «Разработка чат-бота для абитуриентов ГБПОУ «Ногинский колледж» с использованием нейросетевых технологий»**

Выполнил (а) обучающийся

группы 4ИСПр1

Булаткин Даниил Алексеевич

(ф.и.о.)

Руководитель дипломной работы от ОУ

Путилов Евгений Александрович

(ф.и.о.)

Ногинск, 2025 год

Содержание

## Введение

В современном образовании цифровые технологии всё шире применяются для автоматизации процессов и повышения качества обслуживания. Одним из перспективных инструментов в этой области являются чат-боты – программы, способные вести диалог с пользователями в удобном текстовом формате. Они позволяют значительно ускорить ответы на типовые вопросы и разгрузить сотрудников учебного заведения. В условиях, когда абитуриентам приходится ориентироваться во множестве сведений о специальностях, сроках подачи документов и правилах поступления, наличие круглосуточного цифрового помощника становится важным преимуществом.

Тема данной курсовой работы выбрана из практических соображений: колледжи стремятся упростить и сделать более прозрачными процессы приёма, чтобы повысить доверие абитуриентов к образовательному учреждению. Созданный чат-бот позволит абитуриенту быстро получить ответ на вопрос в любое время суток, что избавит как молодого человека, так и сотрудников приёмной комиссии от многих лишних звонков и встреч.

В работе использованы современные достижения в области нейросетей и обработки естественного языка. Чат-бот, разработанный в рамках проекта, опирается на возможности OpenAI и включает языковую модель Mistral-7B для генерации ответов. Практическая ценность проекта заключается не только в демонстрации технологий ИИ, но и в создании реального инструмента, который может взять на себя часть функций приёмной комиссии и сделать коммуникацию с абитуриентами проще и эффективнее.

В процессе выполнения курсовой работы были выполнены следующие основные этапы: проведён анализ аналогичных решений в других образовательных организациях; определены потребности абитуриентов и сотрудников колледжа; разработаны ключевые функции и архитектура бота; создан рабочий прототип с помощью современных инструментов разработки; проведено тестирование и сформулированы выводы.

Данная курсовая работа направлена на создание конкретного цифрового решения, способного реально улучшить качество обслуживания поступающих и повысить эффективность работы приёмной комиссии.

## Теоретическая часть

### 1.1 Анализ предметной области

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж» – это многопрофильное учебное заведение, готовящее специалистов среднего звена по разным направлениям. Колледж был основан в декабре 2015 года и входит в систему профессионального образования Московской области. Его миссия состоит в содействии профессиональной самореализации студентов, расширении их возможностей и раскрытии талантов.

Высшим органом управления колледжа является дирекция, во главе которой стоит директор. Задачи организации и развития колледжа распределены между заместителями директора, каждый из которых курирует определённое направление: учебно-методическую работу, воспитательную деятельность, учебно-производственные процессы, экономику и финансы, административно-хозяйственные вопросы, информационные технологии, безопасность и кадровую политику.

Учебная деятельность колледжа осуществляется в нескольких корпусах, расположенных в городах Ногинск и Балашиха. Каждый корпус оборудован всем необходимым: учебные кабинеты и лаборатории, методические кабинеты, библиотеки и отделы профориентации. Такая многокорпусная структура обеспечивает разветвлённую инфраструктуру для обучения. Управление филиалами ведётся через систему ответственных лиц, подотчётных центральной дирекции.

Ключевыми подразделениями колледжа являются: учебно-методический отдел (разрабатывающий образовательные программы), отдел содействия трудоустройству выпускников, информационный отдел (ответственный за цифровизацию процессов) и приёмная комиссия, осуществляющая набор абитуриентов. Особое значение для проекта имеет информационный (IT) отдел колледжа: этот отдел поддерживает IT-инфраструктуру, разрабатывает программные решения и внедряет новые технологические платформы в образовательный процесс.

В настоящий момент взаимодействие абитуриентов с колледжем организовано стандартными способами: телефонные консультации, личные приёмы в приёмной комиссии и обмен письмами по электронной почте. Приёмная комиссия работает в будние дни с 9:00 до 16:00, что создаёт неудобства для тех поступающих, которые не могут обратиться в выходные или поздно вечером. Такая система имеет несколько слабых мест: отсутствие доступа к информации вне рабочего времени, повторяющиеся однотипные вопросы, которые требуют одних и тех же ответов, отсутствие автоматической фильтрации запросов и ограниченные возможности удалённой консультации.

Выявленные проблемы свидетельствуют о необходимости цифрового помощника – чат-бота, который смог бы отвечать на вопросы абитуриентов независимо от времени суток. Во-первых, чат-бот повысит доступность информации (особенно в нерабочие часы). Во-вторых, он избавит сотрудников колледжа от рутинного повторения стандартных ответов и позволит автоматизировать сбор статистики по вопросам. Таким образом, внедрение нейросетевого чат-бота рассматривается как элемент цифровой трансформации процесса взаимодействия с абитуриентами.

Колледж располагает корпоративной сетью, объединяющей все корпуса, системой электронного документооборота и официальным сайтом. Имеются локальные базы данных студентов и преподавателей. В настоящее время чат-бот не интегрирован с существующей IT-инфраструктурой, поэтому для его работы потребуются специальные API-интерфейсы. С их помощью будет обеспечена синхронизация данных с базами приёмной комиссии и других учебных подразделений колледжа.

Разрабатываемый чат-бот должен войти в экосистему цифровых сервисов колледжа. Предполагается его взаимодействие с внутренними системами: базами данных образовательных программ, системой онлайн-записи, календарём приёмной кампании и модулями онлайн-тестирования. Одновременно необходимо соблюдать требования безопасности: все процессы работы чат-бота должны соответствовать законодательству РФ о защите персональных данных, чтобы гарантировать конфиденциальность информации абитуриентов.

Ожидается, что внедрение интеллектуального чат-бота даст следующие эффекты:

* **расширение охвата абитуриентов** – бот может обслуживать более широкий круг поступающих (на основе статистики предполагается увеличение аудитории на 40–50%);
* **снижение нагрузки на приёмную комиссию** – за счёт автоматизации частых запросов сотрудники смогут сосредоточиться на сложных задачах;
* **круглосуточная поддержка** – поступающие получают возможность получить ответ вне зависимости от рабочего графика;
* **автоматизация сбора статистики** – система будет фиксировать и анализировать запросы, что поможет улучшать качество консультаций;
* **персонализация образовательных рекомендаций** – на основе истории диалога бот сможет подсказывать абитуриентам подходящие специальности и программы.

Успешная реализация проекта потребует сотрудничества разработчиков с учебно-методическим отделом (для актуализации информации об образовательных программах) и отделом профориентации (для формирования сценариев общения с абитуриентами). Совместно с ними будут подобраны шаблоны ответов и сценарии общения, максимально соответствующие нуждам будущих студентов.

### 1.2 Техническое задание

**Цель создания продукта.** Разработка интеллектуального Telegram-бота на основе технологий OpenAI, обеспечивающего круглосуточное автоматизированное консультирование абитуриентов ГБПОУ «Ногинский колледж». Бот должен предоставлять актуальную информацию о поступлении, программах обучения и других аспектах колледжа, поддерживая удобный персонализированный диалог с пользователями.

**Состав подсистем и функциональные задачи.** Система включает несколько подсистем:

* **Модуль обработки естественного языка (NLP).** Использует OpenAI API (и интегрируемую языковую модель Mistral-7B) для распознавания пользовательских запросов и генерации ответов на русском языке в контексте диалога.
* **Интерфейс пользователя (Telegram-бот).** Организует взаимодействие с абитуриентами через мессенджер Telegram, обеспечивая доступ к функциям бота на мобильных и десктопных устройствах.
* **Подсистема управления базой знаний.** Хранит и обновляет структурированную информацию о поступлении, специальностях, расписании и правилах колледжа. Предусмотрены механизмы простого редактирования контента сотрудниками без глубоких технических знаний.
* **Административный модуль.** Предназначен для сотрудников колледжа. Позволяет обновлять базу знаний, просматривать статистику обращений, формировать и корректировать сценарии диалога с пользователями (планируется в ходе последующей доработки проекта).

Функциональные задачи системы:

* Приём и обработка вопросов абитуриентов в реальном времени через Telegram.
* Предоставление точной и актуальной информации по темам поступления и обучения.
* Поддержка FAQ и сценариев для типовых запросов.
* Возможность передачи сложных вопросов оператору при необходимости.
* Сбор и анализ статистики запросов для улучшения качества обслуживания.

**Требования к подсистемам.**

* **Модуль NLP** должен обеспечивать высокую точность распознавания запросов и генерации ответов с учётом контекста диалога, используя возможности OpenAI API (Mistral-7B).
* **Пользовательский интерфейс** реализуется как Telegram-бот с интуитивно понятным интерфейсом, поддерживающим форматирование сообщений (жирный текст, курсив и т.д.) для удобства восприятия.
* **База знаний** должна быть структурированной и легко обновляемой сотрудниками колледжа без необходимости программирования. Информация должна храниться в формате, удобном для поиска и редактирования.
* **Административный модуль** предусматривает разграничение прав доступа: разные учётные записи для администраторов, сотрудников и разработчиков, с шифрованием конфиденциальных данных (например, API-ключей).

**Требования к информационной базе, программному и техническому обеспечению.** Необходимо использовать облачные сервисы для обработки языка (OpenAI API) и для генерации ответов. База знаний и логи взаимодействий должны храниться в защищённой базе данных с регулярным резервным копированием. Серверная часть должна обеспечивать стабильное соединение с OpenAI API и Telegram Bot API, а также поддерживать одновременную работу не менее 100 пользователей без потери производительности. Обмен сообщениями с абитуриентами осуществляется через Telegram Bot API по защищённому протоколу. Все операции с персональными данными пользователей должны соответствовать российскому законодательству о защите информации.

**Общие требования к системе.** Система должна быть доступна круглосуточно. Задержка ответа чат-бота не должна превышать 2 секунд, чтобы обеспечить оперативное общение. Обеспечивается высокая отказоустойчивость и масштабируемость (система должна корректно работать при увеличении нагрузки и позволять добавлять новые модули). Интерфейс чат-бота и административной панели должен быть понятным и удобным. При проектировании учтена возможность дальнейшего расширения функциональности — например, добавление голосового ввода или интеграция с другими сервисами.

### 1.3 Анализ программного и технического обеспечения

**Программное обеспечение.** Для разработки чат-бота использовались следующие средства:

* **PyCharm** – интегрированная среда разработки (IDE) для Python. Она обеспечивает комфортное написание, тестирование и отладку кода, имеет автодополнение и инструменты работы с виртуальными окружениями.
* **GitHub Desktop** – клиент для работы с системой контроля версий Git. С его помощью организовано ведение исходного кода проекта и совместная разработка через репозиторий.
* **Mozilla Firefox** – веб-браузер, применялся для тестирования веб-интерфейсов, проверки документации и взаимодействия с онлайн-сервисами (например, OpenAI Playground).
* **Telegram Desktop** – официальный клиент мессенджера Telegram. Использовался для тестирования работы бота на стороне пользователя, проверки отправки и получения сообщений.
* **Postman** – инструмент для тестирования HTTP-запросов к API. Позволяет выполнять и анализировать запросы к OpenAI API и Telegram Bot API (например, проверять корректность формата запросов и ответов).
* **Библиотеки Python:**
  + aiogram – асинхронная библиотека для удобной работы с Telegram Bot API (обработка сообщений, команд, состояний диалога).
  + openai – официальная библиотека для взаимодействия с OpenAI API (отправка запросов к языковой модели и получение ответов).
  + requests – библиотека для работы с HTTP-запросами (использовалась также для отправки запросов к другим веб-сервисам, если понадобится).
  + python-dotenv – библиотека для чтения конфиденциальных данных (ключей API) из файла .env, что позволяет хранить секретные ключи вне кода.

Этот набор программного обеспечения обеспечивает полный цикл разработки, тестирования и интеграции чат-бота с внешними сервисами OpenAI и Telegram.

**Техническое обеспечение.** Разработка и тестирование проводились на персональном компьютере со следующими характеристиками:

* **Процессор:** AMD Ryzen 5 2600.
* **Оперативная память:** 16 ГБ DDR4.
* **Графический адаптер:** NVIDIA GeForce GTX 1060 (использовался преимущественно для обработки дополнительных вычислительных задач, хотя основной расчёт идёт на стороннем сервере OpenAI).
* **Операционная система:** Windows 10.

Серверная часть развёрнута на локальном сервере колледжа. Это обеспечивает полный контроль над серверными ресурсами, повышает надёжность и безопасность системы. Связь с OpenAI API и Telegram API осуществляется через стабильное интернет-соединение. В настоящий момент дополнительные аппаратные средства безопасности (например, VPN или оборудование для шифрования трафика) не используются, однако их внедрение может быть рассмотрено в дальнейшем для повышения уровня защиты данных.

### 1.4 Проектирование программного продукта

При проектировании чат-бота была создана концептуальная модель, отражающая основные бизнес-процессы, информационные потоки и функциональные компоненты системы. Главная задача бота – автоматизация консультирования абитуриентов по вопросам поступления, учебных программ и организации учёбы. В основе взаимодействия лежит обработка естественного языка с помощью нейросетевой модели: абитуриент задаёт вопрос в Telegram, бот передаёт текстовую формулировку модели OpenAI (Mistral-7B) и возвращает сформированный ответ.

**Информационная модель и бизнес-процессы.** Процесс работы системы можно описать так: пользователь (абитуриент) отправляет текстовый запрос через Telegram-бот с вопросом о колледже (например, о специальностях, необходимых документах или сроках подачи заявлений). Чат-бот принимает запрос и формирует к нему запрос к API OpenAI, указывая системную подсказку, ориентированную на помощь абитуриентам Ногинского колледжа. OpenAI API обрабатывает запрос и возвращает сгенерированный ответ. Бот получает ответ от модели и отправляет его пользователю.

В перспективе планируется административный интерфейс, через который сотрудники колледжа смогут: обновлять базу знаний (загружать новые шаблоны ответов, вносить изменения в информацию), просматривать статистику обращений (например, частоту запросов по тематикам) и корректировать сценарии общения.

В данной реализации классической реляционной базы данных для хранения знаний не создаётся, поскольку основная логика ответов обеспечивается внешней нейросетью. Вместо этого локально сохраняются **логи диалогов** и **настройки бота**. Основные объекты данных системы:

* **Пользователь:** уникальный идентификатор Telegram (ID), история сообщений и запросов.
* **Запрос:** текст сообщения, отправленного пользователем.
* **Ответ:** текст ответа, сгенерированный нейросетью.
* **Настройки бота:** параметры генерации (значения температуры, максимальной длины ответа, системных подсказок).
* **Статистика:** совокупная информация о количестве запросов, темах вопросов и т.д.

**Логическая модель системы.** В архитектуре чат-бота выделены несколько ключевых модулей:

* **Модуль взаимодействия с Telegram API.** Принимает и отправляет сообщения через Telegram.
* **Модуль обработки запросов.** Получает сообщения от пользователей, формирует запросы к OpenAI API и обрабатывает ответы.
* **Модуль интеграции с OpenAI API.** Отвечает за установление соединения с внешним сервисом ИИ и получение сгенерированных ответов.
* **Модуль логирования.** Сохраняет историю диалогов и служебную информацию (время запросов, результаты).
* **Административный модуль** (планируемый) – интерфейс для сотрудников колледжа (обновление данных, анализ запросов).

Для формализации архитектуры использовались UML-диаграммы: диаграмма прецедентов (Use Case), диаграмма компонентов и диаграмма последовательности. На Use Case-диаграмме выделены сценарии взаимодействия: абитуриент отправляет запрос и получает ответ, администратор может обновлять базу знаний и просматривать статистику. Компонентная диаграмма отражает модули системы и их связи: Telegram API связан с модулем взаимодействия, который через модуль обработки запросов обращается к OpenAI API; модуль обработки запросов также сохраняет данные через модуль логирования; административный модуль взаимодействует с модулем логирования для доступа к статистике. На диаграмме последовательности показано: абитуриент -> чат-бот (Bot) -> OpenAI API -> чат-бот -> абитуриент – процесс обмена сообщениями с вызовом ИИ-сервиса для генерации ответа.

## Практическая часть

Приступая к практической реализации, была произведена настройка рабочей среды. Первым шагом была установка интегрированной среды разработки **PyCharm**. С дистрибутива JetBrains выполнялись стандартные действия по инсталляции: создание ярлыков на рабочем столе и ассоциация файлов .py с данной IDE. После установки выполнена первоначальная настройка среды: выбран интерфейс (тёмная тема), настроены основные параметры редактора кода (шрифт, отступы), что улучшило удобство разработки.

Далее в PyCharm создан новый проект для реализации Telegram-бота. Проект размещён в отдельной директории, что обеспечивает изоляцию от других разработок и упрощает управление исходными файлами. При создании проекта настроена структура папок так, чтобы логически разделить компоненты системы (например, отделить скрипты бота от скриптов админки).

На следующем этапе было создано **виртуальное окружение Python**. Оно выполняет изоляцию зависимостей проекта: в нём устанавливаются необходимые библиотеки без влияния на глобальные пакеты. Виртуальное окружение создавалось командой python -m venv myenv. После создания окружения оно было вручную назначено интерпретатором проекта через настройки PyCharm (File → Settings → Project → Python Interpreter → Add Interpreter → Add Local Interpreter). Правильная настройка интерпретатора важна для корректной работы автодополнения кода и встроенных инструментов PyCharm; без этого среда разработки не могла бы полноценно отслеживать установленные пакеты.

Во время подготовки среды возникла проблема с доступностью пакетного менеджера pip. Оказалось, что по умолчанию pip не был установлен в созданном окружении из-за особенностей используемого дистрибутива Python или настроек PATH. Для решения была загружена официальная утилита get-pip.py, после чего она была выполнена командой python get-pip.py внутри виртуального окружения. Это позволило вручную инсталлировать pip и обеспечить его корректную работу.

После восстановления работы pip приступили к установке необходимых библиотек: вводилась команда pip install aiogram openai. Библиотека **aiogram** необходима для удобной работы с Telegram Bot API и поддерживает асинхронное программирование, что обеспечивает высокую производительность бота при обработке множества запросов. Библиотека **openai** служит для взаимодействия с OpenAI API и предоставляет методы для отправки запросов к языковой модели и получения ответов.

К сожалению, автоматическая установка библиотек через pip впервые не сработала из-за нескольких причин: возможные конфликты зависимостей, нестабильное сетевое подключение к репозиторию PyPI или ограничения прав в системе. В связи с этим была выполнена **ручная установка** библиотек. Для этого исходные коды библиотек aiogram и openai были загружены с официальных репозиториев GitHub и PyPI. В каждой папке с библиотекой запускался скрипт python setup.py install. В результате библиотеки были успешно встроены в виртуальное окружение, и теперь aiogram и openai доступны для использования в проекте.

**Заключение этапа настройки среды разработки:** В результате подготовительных действий создана полнофункциональная среда разработки для проекта чат-бота. В неё входят: IDE PyCharm с правильно настроенным интерпретатором Python, изолированное виртуальное окружение для управления зависимостями, функционирующий пакетный менеджер pip и установленные необходимые библиотеки (aiogram, openai и др.). Такая конфигурация обеспечивает надёжную основу для дальнейшей разработки, тестирования и отладки программного продукта.

### Этап 8: Написание основного кода чат-бота с интеграцией ИИ

После настройки среды разработки и установки всех компонентов началась реализация основной логики Telegram-бота. В коде проекта использовались возможности библиотек aiogram и openai для взаимодействия с мессенджером и языковой моделью соответственно.

**Импорт модулей.** В начале скрипта были импортированы необходимые библиотеки и классы. Из aiogram подключались основные компоненты:

* Bot – класс для работы с Telegram Bot API,
* Dispatcher – менеджер событий для маршрутизации входящих сообщений к обработчикам,
* types – модуль, содержащий типы данных для сообщений и других объектов,
* Command – фильтр для обработки команд (например, /start),
* Message – класс для работы с текстовыми сообщениями.  
  Также был подключён модуль asyncio для работы с асинхронными функциями, а из библиотеки openai – клиент для отправки запросов к языковой модели.

**Конфигурация токенов и API-ключей.** Следующим шагом настроены конфиденциальные данные. Необходимо два ключа доступа:

* **Токен Telegram-бота** – уникальная строка, полученная от сервиса BotFather в Telegram. Она необходима для аутентификации бота при общении с Bot API и выглядит как набор чисел и букв.
* **API-ключ OpenAI** – секретный ключ доступа к сервису обработки естественного языка (OpenAI API или альтернативному облачному сервису ИИ). Этот ключ хранится в файле конфигурации (например, .env) и загружается с помощью dotenv при старте программы.

После получения ключей и определения адреса базового URL сервиса (при использовании альтернативы OpenAI, такой как AIML API), инициализируется клиент API. Такой подход позволяет сохранить стандартный интерфейс библиотеки openai для запросов, но перенаправлять их на выбранный провайдер услуг искусственного интеллекта. Благодаря этому в будущем можно легко переключиться между разными поставщиками ИИ без существенных изменений в коде бота.

**Создание объектов бота и диспетчера.** После настройки доступа был создан объект Telegram-бота с использованием полученного токена. Для более удобного форматирования ответов в сообщениях установлено значение parse\_mode="HTML", что позволяет использовать жирный текст, курсив и другие элементы разметки при отправке ответа. Затем инициализирован диспетчер (Dispatcher), который является центральным элементом обработки обновлений. Диспетчер получает события от Telegram API и направляет их к соответствующим обработчикам (функциям-обработчикам), настроенным ниже.

**Обработчик команды /start.** Первой реализованной функцией бота стал обработчик команды /start. Это стандартная команда начала диалога с ботом в Telegram. В момент получения /start бот отправляет приветственное сообщение: «Привет! Я чат-бот Ногинского колледжа. Чем могу помочь?». Это сообщение информирует пользователя о назначении бота и приглашает к общению. Обработчик выполнен как асинхронная функция с декоратором @dp.message(Command("start")), который автоматически вызывает её при входящем событии команды /start.

**Универсальный обработчик сообщений.** Основной механизм работы бота реализован через единый обработчик всех текстовых сообщений. Он определён декоратором @dp.message() без дополнительных фильтров, что делает его «ловушкой» для любых текстовых запросов, не являющихся командами. При поступлении сообщения от пользователя этот обработчик выполняет логику «запрос–ответ»: текст запроса передаётся в языковую модель для генерации ответа.

В обработчике сначала формируется контекстный запрос к OpenAI API. В него включаются: системная подсказка и сообщение пользователя. Системная роль (параметр system) получает текст вроде «Вы помогаете абитуриентам Ногинского колледжа», чтобы фокусировать модель на тематике колледжа и специфике приёмной кампании. Роль пользователя (user) содержит сам вопрос абитуриента. Модель Mistral-7B-Instruct обрабатывает этот контекст и возвращает ответ на русском языке. Затем бот получает сгенерированный ответ и отправляет его абитуриенту в Telegram. Подход «каждое сообщение – отдельный запрос к ИИ» обеспечивает естественное, похожее на диалог взаимодействие, без необходимости знать специальные команды.

**Интеграция с моделью Mistral-7B.** Ключевой особенностью реализации стало использование языковой модели **Mistral-7B-Instruct-v0.2**. Mistral 7B – это открытая модель на 7.3 миллиарда параметров, разработанная компанией Mistral AI и выпущенная под лицензией Apache 2.0. Она демонстрирует высокую эффективность благодаря передовым механизмам: Grouped-query attention (GQA) для ускорения вывода и Sliding Window attention (SWA) для качественной работы с длинными текстами. Благодаря этому Mistral-7B выдаёт высококачественные ответы при относительно низкой задержке вывода.

Для оптимальной работы были настроены параметры генерации текста модели:

* **temperature** (0.7) – этот параметр отвечает за «творчество» модели. Значение 0.7 даёт баланс между детерминированностью и разнообразием ответов, делая их естественными и живыми.
* **max\_tokens** (256) – ограничивает максимальную длину ответа. Выбранное значение обеспечивает развернутые ответы при сохранении их читабельности и контролирует затраты на запросы.

Также настроена система ролей (messages) для сеанса с моделью:

* Системная роль (system) определяет контекст работы бота («Вы помогаете абитуриентам Ногинского колледжа»). Это ограничивает сферу ответов модели тематикой колледжа.
* Роль пользователя (user) содержит фактический текст запроса абитуриента.

**Обработка ошибок и исключений.** Для надёжности системы важен комплексный механизм обработки возможных ошибок. Код бота обёрнут в конструкцию try-except, что позволяет безопасно реагировать на непредвиденные ситуации (принцип graceful degradation). Обработаны различные типы исключений: сетевые ошибки при обращении к API, превышение квот запросов, недоступность внешних сервисов, ошибки парсинга входящих данных и др. При возникновении ошибки пользователь получает информативное сообщение, например: «Извините, при обработке вашего запроса произошла ошибка. Попробуйте позже.», – чтобы не оставить абитуриента без обратной связи и не прерывать общий диалоговый процесс.

**Главная функция запуска.** В завершение кода реализована функция main(), которая запускает бота. Она использует механизм long polling: бот постоянно опрашивает сервера Telegram о новых сообщениях. Структура файла настроена так, что при прямом запуске модуля (if \_\_name\_\_ == "\_\_main\_\_":) вызывается asyncio.run(main()). Это гарантирует, что главный цикл обработки сообщений будет запущен только при непосредственном исполнении данного скрипта, а не при его импорте как модуля.

**Заключение этапа разработки кода.** В результате реализации был создан полнофункциональный Telegram-бот, интегрированный с современными технологиями ИИ. Архитектура решения обеспечивает **высокую производительность** благодаря асинхронной обработке запросов, **надёжность** за счёт комплексной системы обработки ошибок и **высокое качество ответов** благодаря использованию продвинутой языковой модели Mistral-7B. Система также обладает **масштабируемостью**: её можно расширять для обслуживания большого количества одновременных пользователей без значительных переделок. Чёткая структура кода и использование стандартных библиотек Python обеспечивают **простоту сопровождения** и возможность дальнейшего развития проекта. Всё это создаёт прочную основу для добавления новых функций (например, голосового ввода или интеграции с дополнительными сервисами) и делает систему готовой к эксплуатации в работе приёмной комиссии.