**Функционал и Логика Работы**

Бот будет иметь четкую иерархическую структуру команд (меню) для простоты навигации:

1. **Главное меню** (отображается по команде `/start` или `/help`):
2. **Лекционные материалы**

* Раздел 1: Программирование
* Раздел 2: Базы данных
* Раздел 3: Сети
* Раздел 4: …

1. **Задать вопрос нейросети**
2. **Сделать краткий конспект (из аудио)**
3. **Помощь / О боте**

**Путь Пользователя (User Flow)**

**1. Старт:** Пользователь запускает бота командой `/start`.

**2. Приветствие:** Бот отправляет приветственное сообщение с кратким описанием своих возможностей и показывает главное меню с кнопками.

**3. Работа с лекциями:**

* Пользователь нажимает " Лекционные материалы".
* Бот показывает список доступных тем/курсов (например, "1 курс", "2 курс" или "Программирование", "Алгоритмы").
* Пользователь выбирает тему.
* Бот показывает список лекций в этой теме (например, "Лекция 1: Введение в Python", "Лекция 2: Типы данных").
* Пользователь выбирает лекцию.
* Бот отправляет сообщение: "По этой лекции доступны:"

1. Кнопка ` Аудиозапись` (ссылка на файл в Google Drive)
2. Кнопка ` Презентация` (ссылка на PDF/PPTX в Google Drive)
3. Кнопка ` Краткий конспект` (заранее подготовленный или сгенерированный текст)

**Вопрос к нйросети**

* Пользователь нажимает "Задать вопрос нейросети".
* Бот запрашивает: "Введите ваш вопрос по теме информатики:".
* Пользователь вводит текстовый вопрос (например, "Объясни принцип ООП").
* Бот отправляет запрос через API (например, DeepSeek-V3.1) и stream-ом присылает ответ, имитируя печать.

**Создание конспекта из аудио**

* + Пользователь нажимает " Сделать краткий конспект".
  + Бот просит: "Загрузите аудиозапись лекции (формат MP3, WAV)."
  + Пользователь загружает аудиофайл прямо в чат.
  + Бот подтверждает получение: "Аудио получено, обрабатываю... Это может занять несколько минут."
  + Бот отправляет аудио в Speech-to-Text API (например, Яндекс SpeechKit), получает текст.
  + Полученный текст отправляется в LLM с промтом: "Сделай краткий структурированный конспект этого текста, выдели основные тезисы и определения."
  + Бот присылает пользователю готовый конспект.

**Технологический Стек**

* **Язык программирования:** Python 3.11+
* **Фреймворк для бота:** python-telegram-bot v20.x (асинхронная версия - современный и мощный стандарт), Telebot (простая и лёгкая библиотека для Telegram Bot API), Aiogram (асинхронная библиотека для Telegram Bot API. Построена на asyncio).
* **База данных:**
  + **Вариант 1 (Проще):** SQLite3 - для небольшого проекта и быстрого старта.
  + **Вариант 2 (Надежнее):** PostgreSQL - если планируется много пользователей и данных.
* **Работа с Google Drive:** Google Drive API v3. Библиотека google-api-python-client, google-auth.
* **Нейросети:**
  + **Для чата (Question-Answering):** , DeepSeek-V3.1
  + **Для транскрибации аудио (Speech-to-Text):** Яндекс SpeechKit.
* **Хостинг:** …

**Механика Работы и Архитектура (Кратко)**

1. **Команда /start:** Бот заносит пользователя (его user\_id) в таблицу users в БД, если его там еще нет.
2. **Хранение данных о лекциях:** В БД создается таблица lectures с полями: id, topic, name, drive\_audio\_id, drive\_slides\_id, summary\_text.
3. **Ссылки на Google Drive** хранятся не как полные URL, а как ID файлов.
4. Бот формирует **прямую ссылку для скачивания**
5. Получение файлов с **Google Drive:**

* Бот имеет сервисный аккаунт в Google Cloud.
* Файлы на Google Drive должны быть размещены в папке, доступной на чтение для этого сервисного аккаунта (по его email).
* Бот по ID файла из БД запрашивает метаданные файла и формирует прямую ссылку для пользователя.

1. **Работа с нейросетями:**

* "Задать вопрос": Текст пользователя напрямую (с небольшим контекстным промтом, например, "Ты - помощник по информатике. Ответь на вопрос студента") отправляется в DeepSeek-V3.1.
* "Конспект из аудио": Полученный файл сохраняется временно на сервере. Отправляется в Яндекс SpeechKit. Текст отправляется в DeepSeek-V3.1 с промтом "Создай конспект".

1. **Состояния** **(States):** Для обработки многошаговых действий (как загрузка аудио) используется ConversationHandler из python-telegram-bot. Он запоминает, что бот ждет от пользователя в данный момент.

**Ключевые моменты для удобства (UX)**

* Инлайн-кнопки: Весь основной функционал должен быть доступен через кнопки под сообщениями (ReplyKeyboardMarkup, InlineKeyboardMarkup), а не через ввод команд вручную.
* Состояние "Загрузки": При долгих операциях (обработка аудио) бот должен отправить сообщение "Обрабатываю...", а потом отредактировать его или удалить, когда результат будет готов.
* Обработка ошибок: Бот должен корректно обрабатывать все ошибки (проблемы с API, неверный ввод пользователя) и сообщать об этом понятным языком, а не "падать".
* Кэширование: Для одинаковых вопросов к нейросети можно использовать кэш из БД, чтобы не тратить токены и время.

**Блок-схема работы бота**

