Приложение Б

Листинг

```
import telebot
from telebot import types
import json
import os
import random
import time

TOKEN = '8404575855:AAHMTeQhO579sdN7zl6J67oVvHlRqsODCyM'
ADMIN_ID = [671065337]

DB_FILE = 'users.json'
FEEDBACK_FILE = 'feedback.json'
CONTENT_FILE = 'content.json'

bot = telebot.TeleBot(TOKEN)

def load_json(filename, default):
    if not os.path.exists(filename): return default
    try:
        with open(filename, 'r', encoding='utf-8') as f:
            return json.load(f)
    except:
        return default

def save_json(filename, data):
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=4)

bot_content = load_json(CONTENT_FILE, {"theory_data": {}, "cases": {}})
users_db = load_json(DB_FILE, {})

TEST_QUESTIONS = [
    {"q": "На чем основан механизм действия техники «Три да»?", "options": ["На жестком давлении", "На создании установки на согласие", "На запутывании собеседника"], "correct": "На создании установки на согласие"},
    {"q": "На чем фокусируется внимание в технике «Принципиал»?", "options": ["На личных качествах", "На эмоциях", "На интересах, фактах и критериях"], "correct": "На интересах, фактах и критериях"},
    {"q": "Главное преимущество метода «Маленькие ходы»?", "options": ["Получить всё и сразу", "Создает динамику и снижает риск отказа", "Запутывает оппонента"], "correct": "Создает динамику и снижает риск отказа"},
    {"q": "Что характеризует технику «Игра в одни ворота»?", "options": ["Установление контроля над повесткой", "Передача инициативы", "Игнорирование аргументов"], "correct": "Установление контроля над повесткой"},
    {"q": "Задача замены «нет» на «если»?", "options": ["Прекратить переговоры", "Перевести диалог в обсуждение условий", "Показать безразличие"], "correct": "Перевести диалог в обсуждение условий"},
    {"q": "Что является нарушением этикета?", "options": ["Спокойный тон", "Разъяснение позиций", "Перебивание собеседника"], "correct": "Перебивание собеседника"},
    {"q": "Какой сигнал тела показывает интерес?", "options": ["Резкие жесты", "Наклон корпуса вперед", "Закрытая поза"], "correct": "Наклон корпуса вперед"},
    {"q": "К чему приводит отделение проблемы от личностей?", "options": ["К росту напряжения", "К снижению прозрачности", "К совместному поиску решения"], "correct": "К совместному поиску решения"},
    {"q": "Какая техника снижает психологический барьер?", "options": ["Три да", "Игра в одни ворота", "Принципиал"], "correct": "Три да"},
    {"q": "Риск техники «Игра в одни ворота»?", "options": ["Потеря инициативы", "Затягивание времени", "Защитная реакция собеседника"], "correct": "Защитная реакция собеседника"}
]

PRACTICUM_QUIZ = [
    {"q": "«Давайте пока не будем спорить о финальной сумме. Предлагаю для начала согласовать только график...»", "options": ["Три да", "Маленькие ходы", "Принципиал"], "correct": "Маленькие ходы"},
    {"q": "«Коллеги, я сам буду модерировать встречу. Сначала первый пункт, вопросы в конце...»", "options": ["Игра в одни ворота", "Если вместо нет", "Язык тела"], "correct": "Игра в одни ворота"},
    {"q": "«Мы готовы снизить цену на 10%, ЕСЛИ вы заключите договор на два года...»", "options": ["Три да", "Принципиал", "Если вместо нет"], "correct": "Если вместо нет"},
    {"q": "«Давайте уберем эмоции и посмотрим на рынок. Вот статистика зарплат...»", "options": ["Маленькие ходы", "Принципиал", "Три да"], "correct": "Принципиал"},
    {"q": "«Вам нравится дизайн? (Да) Удобно? (Да) Хотите пользоваться? (Да) Тогда оформляйте!»", "options": ["Три да", "Игра в одни ворота", "Маленькие ходы"], "correct": "Три да"}
]
```

```
TRAINING_SCENARIOS = {
    "Три да": [
        {
            "q": "Ситуация 1: Вы хотите продать клиенту дорогую подписку. С чего начнете?",
            "options": [
                {"text": "Вам ведь важно сэкономить время сотрудников?", "correct": True, "feedback": "✅Правильно: это очевидная выгода, клиент скажет «Да»"},
                {"text": "Купите подписку, она очень выгодная!", "correct": False, "feedback": "❌Ошибка: Слишком в лоб"},
                {"text": "У нас сейчас действуют скидки до конца месяца.", "correct": False, "feedback": "❌Ошибка: Информирование, а не вовлечение"}
            ]
        },
        {
            "q": "Ситуация 2: Клиент ответил «Да». Ваш следующий шаг?",
            "options": [
                {"text": "А качество сервиса и отсутствие сбоев для вас имеет значение?", "correct": True, "feedback": "✅ Правильно: второе «Да»"},
                {"text": "Тогда платите по счету.", "correct": False, "feedback": "❌Ошибка: Рано переходить к продаже"},
                {"text": "Ну тогда оформляем договор?", "correct": False, "feedback": "❌Ошибка: Слишком резкий переход"}
            ]
        },
        {
            "q": "Ситуация 3: Вы просите начальника отпустить вас пораньше. Начало разговора:",
            "options": [
                {"text": "Иван Иваныч, мы же сегодня закрыли отчет вовремя?", "correct": True, "feedback": "✅Правильно: факт, с которым нельзя спорить"},
                {"text": "Можно мне сегодня домой пораньше?", "correct": False, "feedback": "❌Ошибка: Прямая просьба, легко отказать"},
                {"text": "Я устал, пойду я.", "correct": False, "feedback": "❌Ошибка: Непрофессионально"}
            ]
        },
        {
            "q": "Ситуация 4: Разговор с другом. Хотите пойти в кино на ужастик.",
            "options": [
                {"text": "Слушай, ты же любишь иногда пощекотать себе нервы?", "correct": True, "feedback": "✅Правильно: апелляция к интересам друга"},
                {"text": "Пошли на «Пилу 10», там круто!", "correct": False, "feedback": "❌Ошибка: Навязывание своего вкуса"},
                {"text": "Мне скучно, пошли в кино.", "correct": False, "feedback": "❌Ошибка: Эгоистичная просьба"}
            ]
        },
        {
            "q": "Ситуация 5: Финальный аккорд. Вы получили 2 «Да». Как переходим к сути?",
            "options": [
                {"text": "Тогда предлагаю выбрать этот вариант, он как раз учитывает всё вышесказанное.", "correct": True, "feedback": "✅Правильно: логичное завершение цепочки"},
                {"text": "Ну, раз ты согласен, давай сделаем как я хочу.", "correct": False, "feedback": "❌Ошибка: Звучит как манипуляция"},
                {"text": "Значит, ты обязан согласиться.", "correct": False, "feedback": "❌Ошибка: Давление"}
            ]
        }
    ],
    "Если вместо нет": [
        {
            "q": "Ситуация 1: Клиент: «Сделайте нам скидку 50%». Вы не можете дать такую скидку.",
            "options": [
                {"text": "Мы можем обсудить такую скидку, если вы оплатите абонемент на год вперед.", "correct": True, "feedback": "✅Правильно: условие «win-win»"},
                {"text": "Нет, это невозможно, у нас фиксированные цены.", "correct": False, "feedback": "❌Ошибка: Тупик в переговорах"},
                {"text": "Мы не работаем с теми, кто не ценит наш труд.", "correct": False, "feedback": "❌Ошибка: Грубость"}
            ]
        },
        {
            "q": "Ситуация 2: Начальник: «Выйди на работу в воскресенье».",
            "options": [
                {"text": "Я выйду в воскресенье, если получу отгул в понедельник.", "correct": True, "feedback": "✅Правильно: обмен ресурсами"},
```

```
            {"text": "Ни за что, это мой законный выходной.", "correct": False, "feedback": "❌Ошибка: Конфликт"},
            {"text": "Ладно, выйду, куда деваться...", "correct": False, "feedback": "❌Ошибка: Позиция жертвы"}
        ]
    },
    {
        "q": "Ситуация 3: Заказчик: «Нужно сдать проект завтра утром» (а реально нужно 3 дня).",
        "options": [
            {"text": "Я сдам завтра черновой вариант, если финальную версию мы утвердим в среду.", "correct": True, "feedback": "✅Правильно: изменение рамок задачи"},
            {"text": "Нет, я физически не успею.", "correct": False, "feedback": "❌Ошибка: Отказ"},
            {"text": "Я постараюсь, но не обещаю.", "correct": False, "feedback": "❌Ошибка: Неуверенность"}
        ]
    },
    {
        "q": "Ситуация 4: Покупатель: «Доставьте мне это бесплатно».",
        "options": [
            {"text": "Мы оформим бесплатную доставку, если сумма вашего заказа будет от 5000 рублей.", "correct": True, "feedback": "✅Правильно: условие"},
            {"text": "Доставка у нас платная для всех.", "correct": False, "feedback": "❌Ошибка: Жесткий отказ"},
            {"text": "Извините, но нет.", "correct": False, "feedback": "❌Ошибка: Оправдание"}
        ]
    },
    {
        "q": "Ситуация 5: Друг просит в долг большую сумму.",
        "options": [
            {"text": "Я дам тебе эту сумму, если мы напишем расписку и заверим её у нотариуса.", "correct": True, "feedback": "✅Правильно: защита своих интересов"},
            {"text": "Денег нет.", "correct": False, "feedback": "❌ Ошибка: Ложь или отказ"},
            {"text": "Бери, конечно, мы же друзья. (Риск потерять деньги)", "correct": False, "feedback": "❌Ошибка: Риск потерять деньги"}
        ]
    }
],
"Маленькие ходы": [
    {
        "q": "Ситуация 1: Вы хотите продать сложный консалтинг за 1 млн рублей. С чего начать?",
        "options": [
            {"text": "Давайте начнем с платного аудита ваших продаж за 10 тыс. рублей?", "correct": True, "feedback": "✅ Правильно: низкий порог входа"},
            {"text": "Купите сразу полный пакет за миллион, это окупится.", "correct": False, "feedback": "❌Ошибка: Слишком рискованно для клиента"},
            {"text": "Давайте встретимся и сразу подпишем договор.", "correct": False, "feedback": "❌Ошибка: Давление"}
        ]
    },
    {
        "q": "Ситуация 2: Нужно убедить ребенка убраться во всей комнате.",
        "options": [
            {"text": "Давай для начала просто соберем все кубики LEGO в коробку?", "correct": True, "feedback": "✅Правильно: простая задача"},
            {"text": "Уберись в комнате немедленно!", "correct": False, "feedback": "❌Ошибка: Глобальная задача, вызывает протест"},
            {"text": "Пока не будет чисто, мультиков не увидишь.", "correct": False, "feedback": "❌Ошибка: Шантаж"}
        ]
    },
    {
        "q": "Ситуация 3: Клиент сомневается и тянет время. Как продвинуться?",
        "options": [
            {"text": "Давайте оформим тестовый период на 3 дня? Это вас ни к чему не обязывает.", "correct": True, "feedback": "✅ Правильно: снятие страха ошибки"},
            {"text": "Берите, вы точно не пожалеете.", "correct": False, "feedback": "❌Ошибка: Пустые слова"},
            {"text": "Что именно вас смущает? Скажите честно.", "correct": False, "feedback": "❌Ошибка: Может вызвать раздражение"}
        ]
    },
    {
        "q": "Ситуация 4: Вы просите повышение зарплаты сразу в 2 раза.",
        "options": [
```

```
            {"text": "Давайте повысим оклад на 20% сейчас и вернемся к разговору через квартал?", "correct": True, "feedback": "✅Правильно: поэтапный рост"},

            {"text": "Хочу х2 или я увольняюсь.", "correct": False, "feedback": "❌Ошибка: Ультиматум"},

            {"text": "Я очень много работаю, дайте денег.", "correct": False, "feedback": "❌Ошибка: Жалоба"}

        ]

    },

    {

        "q": "Ситуация 5: Нужно внедрить новую сложную программу в отделе, сотрудники сопротивляются.",

        "options": [

            {"text": "Давайте на этой неделе просто зарегистрируемся и заполним профили.", "correct": True, "feedback": "✅Правильно: легкий первый шаг"},

            {"text": "С завтрашнего дня все работают только в новой CRM, за старую штраф.", "correct": False, "feedback": "❌ Ошибка: Шок-терапия"},

            {"text": "Это приказ директора, не обсуждается.", "correct": False, "feedback": "❌Ошибка: Административный ресурс"}

        ]

    }

  ],

  "Принципиал": [

    {

        "q": "Ситуация 1: Арендодатель поднимает цену просто «потому что я так хочу».",

        "options": [

            {"text": "Давайте посмотрим средние цены на аренду в этом районе на Куфаре", "correct": True, "feedback": "✅ Правильно: опора на рынок"},

            {"text": "Это грабеж! Вы не имеете права!", "correct": False, "feedback": "❌Ошибка: Эмоции"},

            {"text": "Ну ладно, я буду платить...", "correct": False, "feedback": "❌Ошибка: Сдача позиций"}

        ]

    },

    {

        "q": "Ситуация 2: Клиент говорит: «Ваши услуги стоят дорого».",

        "options": [

            {"text": "Давайте сравним смету с конкурентами пункт за пунктом и найдем отличия.", "correct": True, "feedback": "✅Правильно: факты и цифры"},

            {"text": "Зато у нас качественно и быстро.", "correct": False, "feedback": "❌Ошибка: Субъективно"},

            {"text": "Ну, я могу сделать небольшую скидку.", "correct": False, "feedback": "❌Ошибка: Торг без обоснования"}

        ]

    },

    {

        "q": "Ситуация 3: Спор с коллегой, чья идея для рекламы лучше.",

        "options": [

            {"text": "Давайте не спорить, а опираться на статистику продаж прошлой похожей акции.", "correct": True, "feedback": "✅Правильно: объективные данные"},

            {"text": "Моя идея лучше, потому что я опытнее тебя.", "correct": False, "feedback": "❌Ошибка: Переход на личности"},

            {"text": "Ты ничего не понимаешь в маркетинге.", "correct": False, "feedback": "❌Ошибка: Оскорбление"}

        ]

    },

    {

        "q": "Ситуация 4: Делите бюджет. Оппонент хочет забрать 80% себе.",

        "options": [

            {"text": "На каком основании вы предлагаете такую пропорцию? Давайте посчитаем вклад каждого.", "correct": True, "feedback": "✅Правильно: требование критериев"},

            {"text": "Так нечестно, давай пополам.", "correct": False, "feedback": "❌Ошибка: Базарный торг"},

            {"text": "Ну хорошо, забирайте.", "correct": False, "feedback": "❌Ошибка: Проигрыш"}

        ]

    },

    {

        "q": "Ситуация 5: Вам навязывают невыгодный пункт договора.",

        "options": [

            {"text": "Этот пункт противоречит Гражданскому кодексу (статья такая-то).", "correct": True, "feedback": "✅ Правильно: опора на закон"},

            {"text": "Уберите это немедленно.", "correct": False, "feedback": "❌Ошибка: Требование"},

            {"text": "Мне это не нравится, перепишите.", "correct": False, "feedback": "❌Ошибка: Вкусовщина"}

        ]

    }

  ],

  "Игра в одни ворота": [
```

```
        {
            "q": "Ситуация 1: Клиент постоянно просит доработки бесплатно. «Ну поправьте еще тут шрифт».",
            "options": [
                {"text": "Я сделал уже 5 правок бесплатно в рамках лояльности. Все следующие будут тарифицироваться.", "correct": True, "feedback": "✅Правильно: установка границ"},
                {"text": "Хорошо, сейчас сделаю.", "correct": False, "feedback": "❌Ошибка: Поощрение потребительского экстремизма"},
                {"text": "Вы наглеете, сколько можно?", "correct": False, "feedback": "❌Ошибка: Конфликт"}
            ]
        },
        {
            "q": "Ситуация 2: Собеседник перебивает и не дает вставить слово.",
            "options": [
                {"text": "Позвольте, я вас не перебивал. Дайте мне закончить мысль, а потом я выслушаю вас.", "correct": True, "feedback": "✅Правильно: требование уважения регламента"},
                {"text": "(Молчать и терпеть)", "correct": False, "feedback": "❌Ошибка: Сдача позиций"},
                {"text": "Заткнись и слушай!", "correct": False, "feedback": "❌Ошибка: Агрессия"}
            ]
        },
        {
            "q": "Ситуация 3: Партнер требует уступок по цене, но ничего не предлагает взамен.",
            "options": [
                {"text": "Мы идем вам навстречу уже третий раз. Где ваше встречное движение?", "correct": True, "feedback": "✅ Правильно: призыв к взаимности"},
                {"text": "Ладно, только подпишите уже договор.", "correct": False, "feedback": "❌Ошибка: Сдача"},
                {"text": "Нет, мы больше не уступим.", "correct": False, "feedback": "❌Ошибка: Тупик без объяснения"}
            ]
        },
        {
            "q": "Ситуация 4: Вам звонят в нерабочее время по пустякам.",
            "options": [
                {"text": "Я ценю наше сотрудничество, но давайте обсуждать рабочие вопросы в рабочее время.", "correct": True, "feedback": "✅Правильно: границы"},
                {"text": "Да, конечно, слушаю.", "correct": False, "feedback": "❌Ошибка: Сдача личного времени"},
                {"text": "Не звоните мне больше так поздно!", "correct": False, "feedback": "❌Ошибка: Истерика"}
            ]
        },
        {
            "q": "Ситуация 5: Оппонент давит на жалость: «Ну сделайте скидку, вам же не сложно».",
            "options": [
                {"text": "Мне не сложно, но мое рабочее время стоит денег. Любая работа должна быть оплачена.", "correct": True, "feedback": "✅Правильно: защита ценности"},
                {"text": "Ну ладно...", "correct": False, "feedback": "❌ Ошибка: Жалость победила"},
                {"text": "Сложно! Отстаньте.", "correct": False, "feedback": "❌Ошибка: Грубость"}
            ]
        }
    ]
}

user_states = {}

def get_main_keyboard(user_id):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    user = users_db.get(str(user_id), {'paid': False})

    markup.add(types.KeyboardButton("□ Уроки по переговорам"))

    if user.get('paid'):
        markup.add(types.KeyboardButton("□ Практикум"))
        markup.add(types.KeyboardButton("□ Тест по теории"), types.KeyboardButton("□ Отработка техник"))
    else:
        markup.add(types.KeyboardButton("□ Оплата"))

    markup.add(types.KeyboardButton("□ Задать вопрос"))
    return markup

def get_lessons_keyboard():
```

```
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    markup.add("□ Теория", "□ Кейсы", "□ В главное меню")
    return markup

def get_theory_keyboard():
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    topics = list(bot_content['theory_data'].keys())
    markup.add(*topics)
    markup.add("□ Назад к урокам")
    return markup

def get_cases_keyboard():
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    btns = [types.KeyboardButton(f"□ {case}") for case in bot_content['cases'].keys()]
    markup.add(*btns)
    markup.add("□ Назад к урокам")
    return markup

def get_training_menu_keyboard():
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    btns = [types.KeyboardButton(f"□ {tech}") for tech in TRAINING_SCENARIOS.keys()]
    btns.append(types.KeyboardButton("□ В главное меню"))
    markup.add(*btns)
    return markup

def get_options_keyboard(options_list):
    """Создает кнопки ответов для тестов"""
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)
    markup.add(*options_list)
    markup.add("□ Выход в меню")
    return markup

def send_question(chat_id, user_id, quiz_data, mode_title):
    state = user_states.get(user_id)
    if not state: return
    index = state['index']

    if index >= len(quiz_data):
        score = state['score']
        total = len(quiz_data)
        bot.send_message(chat_id, f"□ <b>{mode_title} завершен!</b>\nВаш результат: {score} из {total}.",
                         reply_markup=get_main_keyboard(user_id), parse_mode="HTML")
        del user_states[user_id]
        return

    question_data = quiz_data[index]
    question_text = question_data['q']

    if state['mode'] == 'training':
        options = [opt['text'] for opt in question_data['options']]
        random.shuffle(options)
    else:
        options = list(question_data['options'])
        random.shuffle(options)

    bot.send_message(chat_id, f"❓ <b>Вопрос {index + 1}/{len(quiz_data)}</b>:\n{question_text}",
                     reply_markup=get_options_keyboard(options), parse_mode="HTML")

@bot.message_handler(commands=['start'])
def send_welcome(message):
    user_id = str(message.from_user.id)
    if user_id not in users_db:
        users_db[user_id] = {'paid': False, 'username': message.from_user.username, 'payment_requested': False}
        save_json(DB_FILE, users_db)

    if message.from_user.id in user_states:
        del user_states[message.from_user.id]

    bot.send_message(message.chat.id, "Добро пожаловать в Репетитор по техникам ведения переговоров!",
```

```
 reply_markup=get_main_keyboard(message.from_user.id))


@bot.message_handler(commands=['admin'])
def admin_panel(message):
    if message.from_user.id in ADMIN_ID:
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
        markup.add("□ Заявки на оплату", "□ Сообщения")
        markup.add("□ Ред. теорию", "□ Ред. кейсы")
        markup.add("□ Выход из админки")
        bot.send_message(message.chat.id, "□□ Панель администратора", reply_markup=markup)


@bot.message_handler(content_types=['text', 'video', 'photo'])
def handle_message(message):
    user_id = message.from_user.id
    user_id_str = str(user_id)
    text = message.text
    state = user_states.get(user_id)

    if text in ["□ Выход в меню", "□ В главное меню", "□ Выход из админки"]:
        if user_id in user_states: del user_states[user_id]
        bot.send_message(message.chat.id, "Главное меню", reply_markup=get_main_keyboard(user_id))
        return

    if state:
        mode = state['mode']

        if mode == 'test':
            current_q = TEST_QUESTIONS[state['index']]
            if text == current_q['correct']:
                state['score'] += 1
                bot.send_message(message.chat.id, "✅Верно!")
            elif text in current_q['options']:
                bot.send_message(message.chat.id, f"❌Ошибка. Верный ответ: {current_q['correct']}")
            else:
                bot.send_message(message.chat.id, "Пожалуйста, выберите вариант из кнопок.")
                return

            state['index'] += 1
            send_question(message.chat.id, user_id, TEST_QUESTIONS, "Тест")
            return

        if mode == 'practicum':
            current_q = PRACTICUM_QUIZ[state['index']]
            if text == current_q['correct']:
                state['score'] += 1
                bot.send_message(message.chat.id, "✅Отличное решение!")
            elif text in current_q['options']:
                bot.send_message(message.chat.id, f"□ Неверно. Правильная техника: {current_q['correct']}")
            else:
                bot.send_message(message.chat.id, "Используйте кнопки.")
                return

            state['index'] += 1
            send_question(message.chat.id, user_id, PRACTICUM_QUIZ, "Практикум")
            return

        if mode == 'training':
            scenario_name = state['scenario']
            scenario_data = TRAINING_SCENARIOS[scenario_name]
            current_q = scenario_data[state['index']]

            selected = next((opt for opt in current_q['options'] if opt['text'] == text), None)

            if selected:
                bot.send_message(message.chat.id, selected['feedback'])
                if selected['correct']:
                    state['score'] += 1
```

```
            state['index'] += 1
            send_question(message.chat.id, user_id, scenario_data, f"Тренинг: {scenario_name}")
        else:
            bot.send_message(message.chat.id, "Выберите вариант из меню.")
        return

    if mode == 'feedback':
        msg_id = str(int(time.time())) + str(random.randint(100,999))
        feedback_entry = {
            'id': msg_id,
            'user_id': user_id,
            'username': message.from_user.username or "Неизвестно",
            'text': text,
            'date': time.strftime("%d.%m %H:%M")
        }
        fb_list = load_json(FEEDBACK_FILE, [])
        fb_list.append(feedback_entry)
        save_json(FEEDBACK_FILE, fb_list)

        bot.send_message(message.chat.id, "✅Сообщение отправлено администратору!", reply_markup=get_main_keyboard(user_id))
        for admin in ADMIN_ID:
            try: bot.send_message(admin, f"📩 <b>Новый вопрос!</b>\nОт: @{message.from_user.username}\nТекст: {text}", parse_mode="HTML")
            except: pass

        del user_states[user_id]
        return

    if mode == 'admin_reply' and user_id in ADMIN_ID:
        target_msg_id = state['target_id']
        fb_list = load_json(FEEDBACK_FILE, [])
        target_entry = next((m for m in fb_list if m['id'] == target_msg_id), None)

        if target_entry:
            try:
                bot.send_message(target_entry['user_id'], f"📩 <b>Ответ администратора:</b>\n\n{text}", parse_mode="HTML")
                bot.send_message(message.chat.id, "✅Ответ отправлен.", reply_markup=get_main_keyboard(user_id))
                fb_list = [m for m in fb_list if m['id'] != target_msg_id]
                save_json(FEEDBACK_FILE, fb_list)
            except Exception as e:
                bot.send_message(message.chat.id, f"❌Не удалось отправить (пользователь заблокировал бот?): {e}")
        else:
            bot.send_message(message.chat.id, "Сообщение не найдено в базе.")

        del user_states[user_id]
        admin_panel(message)
        return

    if mode == 'edit_theory_content' and user_id in ADMIN_ID:
        topic = state['topic']
        new_content = message.html_text if hasattr(message, 'html_text') else message.text
        bot_content['theory_data'][topic] = new_content
        save_json(CONTENT_FILE, bot_content)
        bot.send_message(message.chat.id, f"✅Теория «{topic}» обновлена.", reply_markup=get_main_keyboard(user_id))
        del user_states[user_id]
        admin_panel(message)
        return

    if mode == 'edit_case_content' and user_id in ADMIN_ID:
        case_name = state['case_name']
        new_type = 'text'
        new_file_id = None
        new_caption = message.caption or message.text or ""

        if message.content_type == 'video':
            new_type = 'video'
            new_file_id = message.video.file_id
        elif message.content_type == 'photo':
            new_type = 'photo'
```

```
            new_file_id = message.photo[-1].file_id
        elif message.content_type == 'text':
            pass
        else:
            bot.send_message(message.chat.id, "❌ Поддерживаются только Текст, Фото или Видео.")
            return

        bot_content['cases'][case_name] = {"type": new_type, "file_id": new_file_id, "caption": new_caption}
        save_json(CONTENT_FILE, bot_content)
        bot.send_message(message.chat.id, f"✅Кейс «{case_name}» обновлен.", reply_markup=get_main_keyboard(user_id))
        del user_states[user_id]
        admin_panel(message)
        return

    if not text: return

    if text == "📝 Тест по теории":
        if not users_db.get(user_id_str, {}).get('paid'):
            bot.send_message(message.chat.id, "🔒 Доступно только после оплаты.")
            return
        user_states[user_id] = {'mode': 'test', 'index': 0, 'score': 0}
        bot.send_message(message.chat.id, "🚀 Начинаем тест!", reply_markup=types.ReplyKeyboardRemove())
        send_question(message.chat.id, user_id, TEST_QUESTIONS, "Тест")

    elif text == "🎯 Практикум":
        if not users_db.get(user_id_str, {}).get('paid'):
            bot.send_message(message.chat.id, "🔒 Доступно только после оплаты.")
            return
        user_states[user_id] = {'mode': 'practicum', 'index': 0, 'score': 0}
        bot.send_message(message.chat.id, "🎯 Определяем", reply_markup=types.ReplyKeyboardRemove())
        send_question(message.chat.id, user_id, PRACTICUM_QUIZ, "Практикум")

    elif text == "🎭 Отработка техник":
        if not users_db.get(user_id_str, {}).get('paid'):
            bot.send_message(message.chat.id, "🔒 Доступно только после оплаты.")
            return
        bot.send_message(message.chat.id, "Выберите технику:", reply_markup=get_training_menu_keyboard())

    elif text.startswith("🎭 "):
        tech_name = text.replace("🎭 ", "")
        if tech_name in TRAINING_SCENARIOS:
            user_states[user_id] = {'mode': 'training', 'scenario': tech_name, 'index': 0, 'score': 0}
            bot.send_message(message.chat.id, f"Тренировка: {tech_name}", reply_markup=types.ReplyKeyboardRemove())
            send_question(message.chat.id, user_id, TRAINING_SCENARIOS[tech_name], f"Тренинг: {tech_name}")

    elif text == "💳 Оплата":
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        markup.add("✅Подтверждаю оплату", "🔙 В главное меню")
        bot.send_message(message.chat.id, "💳 Переведите 20 BYN на карту: 9112 3801 7366 0165 и отправьте скришот администратору: @chestachka\nЗатем нажмите кнопку подтверждения.", reply_markup=markup)

    elif text == "✅Подтверждаю оплату":
        users_db[user_id_str]['payment_requested'] = True
        save_json(DB_FILE, users_db)
        bot.send_message(message.chat.id, "✅Заявка отправлена. Ожидайте подтверждения админа.", reply_markup=get_main_keyboard(user_id))

    elif text == "💬 Задать вопрос":
        user_states[user_id] = {'mode': 'feedback'}
        bot.send_message(message.chat.id, "Напишите ваш вопрос одним сообщением:", reply_markup=types.ReplyKeyboardMarkup(resize_keyboard=True).add("🔙 В главное меню"))

    elif text == "📚 Уроки по переговорам": bot.send_message(message.chat.id, "Выберите раздел:", reply_markup=get_lessons_keyboard())
    elif text == "📖 Теория": bot.send_message(message.chat.id, "Выберите тему:", reply_markup=get_theory_keyboard())
```

```
    elif text == "□ Кейсы": bot.send_message(message.chat.id,
"Выберите кейс:", reply_markup=get_cases_keyboard())

    elif text == "□ Назад к урокам":
bot.send_message(message.chat.id, "Разделы:",
reply_markup=get_lessons_keyboard())


    elif text in bot_content['theory_data']:

        bot.send_message(message.chat.id,
bot_content['theory_data'][text], parse_mode="HTML")


    elif text.startswith("□ "):

        case_name = text.replace("□ ", "")

        if case_name in bot_content['cases']:

            cdata = bot_content['cases'][case_name]

            caption = cdata.get('caption', '')


            try:

                if cdata['type'] == 'video' and cdata['file_id']:

                    bot.send_video(message.chat.id, cdata['file_id'],
caption=caption, parse_mode="HTML")

                elif cdata['type'] == 'photo' and cdata['file_id']:

                    bot.send_photo(message.chat.id, cdata['file_id'],
caption=caption, parse_mode="HTML")

                else:

                    bot.send_message(message.chat.id, caption,
parse_mode="HTML")

            except Exception as e:

                bot.send_message(message.chat.id, f"Ошибка
отображения: {e}\n\nТекст: {caption}")


    elif text == "□ Заявки на оплату" and user_id in ADMIN_ID:

        pending = [uid for uid, u in users_db.items() if
u.get('payment_requested') and not u.get('paid')]

        if not pending:

            bot.send_message(message.chat.id, "Заявок нет.")

        else:

            for uid in pending:

                uname = users_db[uid].get('username', 'Unknown')

                mk = types.InlineKeyboardMarkup()

                mk.add(types.InlineKeyboardButton("✅Одобрить",
callback_data=f"approve_{uid}"))

                bot.send_message(message.chat.id, f"Заявка от
@{uname} (ID: {uid})", reply_markup=mk)


    elif text == "□ Сообщения" and user_id in ADMIN_ID:

        fb_list = load_json(FEEDBACK_FILE, [])

        if not fb_list:

            bot.send_message(message.chat.id, "Сообщений нет.")

        else:

            markup = types.InlineKeyboardMarkup()

            for msg in fb_list:

                markup.add(types.InlineKeyboardButton(f"{msg['date']}
| {msg['username']}", callback_data=f"read_{msg['id']}"))

            bot.send_message(message.chat.id, "Входящие
сообщения:", reply_markup=markup)


    elif text == "□ Ред. ТЕОРИЮ" and user_id in ADMIN_ID:

        markup = types.InlineKeyboardMarkup()

        for t in bot_content['theory_data']:

            markup.add(types.InlineKeyboardButton(t,
callback_data=f"selecttheory_{t}"))

        bot.send_message(message.chat.id, "Выберите тему для
редактирования:", reply_markup=markup)


    elif text == "□ Ред. КЕЙСЫ" and user_id in ADMIN_ID:

        markup = types.InlineKeyboardMarkup()

        for c in bot_content['cases']:

            markup.add(types.InlineKeyboardButton(c,
callback_data=f"selectcase_{c}"))

        bot.send_message(message.chat.id, "Выберите кейс для
редактирования:", reply_markup=markup)


@bot.callback_query_handler(func=lambda call: True)

def handle_callbacks(call):

    user_id = call.from_user.id


    if call.data.startswith("approve_") and user_id in ADMIN_ID:

        uid = call.data.split("_")[1]

        if uid in users_db:

            users_db[uid]['paid'] = True

            users_db[uid]['payment_requested'] = False

            save_json(DB_FILE, users_db)

            bot.edit_message_text(f"✅Оплата для {uid}
подтверждена.", call.message.chat.id, call.message.message_id)

            try: bot.send_message(int(uid), "□ Оплата подтверждена!
Вам доступны Тесты и Практикум.\nНажмите /start для
обновления меню.")
```

```
            except: pass


    elif call.data.startswith("read_") and user_id in ADMIN_ID:
        msg_id = call.data.split("_")[1]
        fb_list = load_json(FEEDBACK_FILE, [])
        msg = next((m for m in fb_list if m['id'] == msg_id), None)
        if msg:
            txt = f"□ @{msg['username']}\n□ {msg['date']}\n\n{msg['text']}"
            markup = types.InlineKeyboardMarkup()
            markup.add(types.InlineKeyboardButton("↩ Ответить", callback_data=f"reply_{msg_id}"),
                       types.InlineKeyboardButton("□ Удалить", callback_data=f"del_{msg_id}"))
            bot.send_message(call.message.chat.id, txt, reply_markup=markup)
        else:
            bot.send_message(call.message.chat.id, "Сообщение не найдено.")


    elif call.data.startswith("del_") and user_id in ADMIN_ID:
        msg_id = call.data.split("_")[1]
        fb_list = load_json(FEEDBACK_FILE, [])
        fb_list = [m for m in fb_list if m['id'] != msg_id]
        save_json(FEEDBACK_FILE, fb_list)
        bot.delete_message(call.message.chat.id, call.message.message_id)


    elif call.data.startswith("reply_") and user_id in ADMIN_ID:
        msg_id = call.data.split("_")[1]
        user_states[user_id] = {'mode': 'admin_reply', 'target_id': msg_id}
        bot.send_message(call.message.chat.id, "☒ Напишите текст ответа:")


    elif call.data.startswith("selecttheory_") and user_id in ADMIN_ID:
        topic = call.data.replace("selecttheory_", "")
        markup = types.InlineKeyboardMarkup()
        markup.add(types.InlineKeyboardButton("□ Изменить текст", callback_data=f"edittheory_{topic}"))
        bot.send_message(call.message.chat.id, f"Тема: {topic}", reply_markup=markup)


    elif call.data.startswith("edittheory_"):
        topic = call.data.replace("edittheory_", "")
        user_states[user_id] = {'mode': 'edit_theory_content', 'topic': topic}
        bot.send_message(call.message.chat.id, f"Пришлите новый текст для «{topic}»:")


    elif call.data.startswith("selectcase_") and user_id in ADMIN_ID:
        case_name = call.data.replace("selectcase_", "")
        markup = types.InlineKeyboardMarkup()
        markup.add(types.InlineKeyboardButton("□ Изменить контент", callback_data=f"editcase_{case_name}"))
        bot.send_message(call.message.chat.id, f"Кейс: {case_name}", reply_markup=markup)


    elif call.data.startswith("editcase_"):
        case_name = call.data.replace("editcase_", "")
        user_states[user_id] = {'mode': 'edit_case_content', 'case_name': case_name}
        bot.send_message(call.message.chat.id, f"Пришлите фото/видео/текст для «{case_name}»:")


if __name__ == '__main__':
    print("Бот запущен...")
    bot.infinity_polling(skip_pending=True)
```