

با سلام

پاسخ به تکه های ارسال شده :

(1) به زبان ساده مدل های هوش مصنوعی (AI) مانند یک مغز هستند که فقط وقتی از آنها سوال پرسیم جواب میدهند و برای انجام کارهایی مانند تولید متن یا ترجمه یک متن و پاسخ به سوالات و یا ... استفاده میشوند.

اما AI agent با استفاده از مدل های هوش مصنوعی کاری مانند آن را انجام میدهند با این تفاوت که دارای قابلیت تصمیم گیری و درک هستند و میتوانند چند مرحله را خودش انجام دهد.

مثال فرق بین چت بات ساده با دستیار هوشمند:

برای مثال میخواهیم فروش سالیانه را از مجموعه ای از داده ها به دست آوریم وقتی این کار را با استفاده از یک چت بات ساده انجام میدهیم نمیتواند به طور خودکار و مستقل برای ما داده ها را بخواند یا استخراج کند و مبلغ فروش سالیانه را داخل فایلی ساختار یافته مانند یک فایل اکسل ذخیره کند.
اما یک دستیار هوشمند میتواند این کار را به خوبی انجام دهد.

(2) معماری یک دستیار هوشمند برای کلینیک پزشکی:

- (a) سیستم نوبت دهی به بیماران
- (b) سیستم پاسخ هوشمند به سوالات متداول
- (c) مازول یاد آوری نوبت بیماران
- (d) سیستم مدیریت پرونده بیماران
- (e) سیستم API

لینک گیت هاب برای تکه های عملی:

https://github.com/aradsahaviyeh/my_tasks.git

ربات تلگرامی

```
from telegram import Update
from telegram.ext import (
   ApplicationBuilder,
    ContextTypes,
    MessageHandler,
    CommandHandler,
    filters,
)
from os import environ
from dotenv import load_dotenv

load_dotenv()
TOKEN = environ["TOKEN"]

# for start command
async def start_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Hello welcome to this Bot")

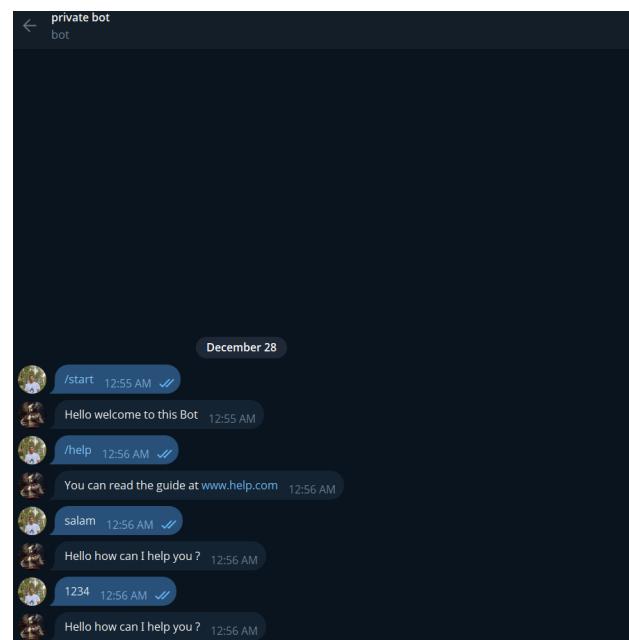
# for help command
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("You can read the guide at www.help.com")

# for any text
async def static_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Hello how can I help you ?")

if __name__ == "__main__":
    app = ApplicationBuilder().token(TOKEN).build()

    # commands
    app.add_handler(CommandHandler("start", start_command))
    app.add_handler(CommandHandler("help", help_command))

    # messages
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, static_message))
```



ساخت API با استفاده از django rest framework

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
In [2]: from django.utils import timezone
In [3]: m = Message()
In [4]: m.message_text = "welcome to my task project"
In [5]: m.status = True
In [6]: m.pub_date = timezone.now()
In [7]: m.save()
In [8]: m
Out[8]: <Message: welcome to my task project>
In [9]: 
```

← → ⌂ 127.0.0.1:8000/web/messages/ Django REST framework

Message

OPTIONS GET

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "message_text": "welcome to my task project",
        "status": true,
        "pub_date": "2025-12-27T21:40:24.253075Z"
    },
    {
        "id": 2,
        "message_text": "Hello this is for test",
        "status": false,
        "pub_date": "2025-12-27T21:41:30.872775Z"
    }
]
```

Media type: application/json

Content:

```
{
    "00858806_text": "this is a test for post data",
    "status": false,
    "pub_date": "2025-12-27T22:41:30.872775Z"
}
```

Message

OPTIONS GET

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 3,
        "message_text": "this is a test for post data",
        "status": false,
        "pub_date": "2025-12-27T22:41:30.872775Z"
    }
]
```

Media type: application/json

Content:

تنظیمات فنی ربات تلگرامی:

برای ساخت ربات تلگرامی، ابتدا با استفاده از **BotFather** یک ربات ایجاد و توکن مربوطه دریافت شد. جهت افزایش امنیت، توکن در فایل **env** ذخیره گردید.
سپس پک **virtual environment** ایجاد و کتابخانه‌های **python-dotenv** و **python-telegram-bot** نصب شدند. در ادامه، توابع لازم برای پاسخ به پیام‌ها و درخواست‌های کاربران پیاده‌سازی شد.

تنظیمات فنی DRF:

برای پیاده‌سازی API با **django rest framework**، ابتدا یک **virtual environment** برای ایزوولم‌سازی پروژه ایجاد و فعال شد. سپس **django** نصب و پروژه با دستور **django-admin startproject** ساخته شد. بعد از آن یک اپلیکیشن با نام **web** ایجاد گردید.

برای ساخت **django rest framework** نصب و در فایل **settings.py** به لیست اپلیکیشن‌ها اضافه شد. مسیرهای مربوط به اپلیکیشن در فایل **urls.py** پروژه و اپ مدیریت شدند.

برای ساخت API‌ها، فایل **serializers.py** ایجاد و **serializer** های موردنیاز پیاده‌سازی شد و در فایل **views.py** مورد استفاده قرار گرفت. با استفاده از متدهای GET و POST امکان دریافت داده‌ها به صورت JSON و ارسال اطلاعات معتبر به دیتابیس فراهم شد.

با تشکر

آزاد ساھویہ