**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

«Бот: кулинарный помощник»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Сомов Кирилл |  | Гапанюк Ю. Е. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2023 г.

# Описание проекта:

Разработка Telegram-бота, который будет предоставлять пользователю кулинарные рецепты на основе выбранных им ингредиентов.

# Функциональные требования:

/start:

Приветственное сообщение при старте бота.

/help:

Предоставление справочной информации о возможностях бота.

/ingredients:

Вывод списка категорий продуктов для выбора.

При выборе категории - вывод доступных ингредиентов из этой категории.

Добавление ингредиентов:

После выбора ингредиента пользователем, он добавляется в список.

/view\_ingredients:

Просмотр текущего списка добавленных ингредиентов.

/remove\_ingredient:

Удаление последнего добавленного ингредиента из списка.

/clear:

Очистка списка добавленных ингредиентов.

/cook:

Поиск подходящего рецепта на основе списка ингредиентов пользователя.

Вывод наилучшего рецепта и списка недостающих ингредиентов, если такие есть.

# Требования к базе данных:

Использование JSON-файлов для хранения данных о категориях ингредиентов и рецептах.

# Логика работы бота:

При выборе категории или ингредиента - формирование клавиатуры с вариантами выбора.

Проверка наличия ингредиента в списке пользователя перед добавлением.

Поиск рецепта осуществляется на основе пересечения ингредиентов пользователя с рецептами.

Обработка ошибок и неопределенных команд:

В случае неопределенной команды выводить сообщение об ошибке и предложение воспользоваться /help для получения справки.

# Технологии:

Использование библиотеки Telebot для работы с Telegram API.

Язык программирования: Python.

Код программы

import telebot  
import json  
from collections import Counter  
  
TOKEN = '6325303584:AAEx2PeNZBpPnmdfXTFzNyXna-VyxbQNfzg'  
bot = telebot.TeleBot(TOKEN)  
  
# Загрузка данных о ингредиентах из файла  
with open('ingredients.json', 'r', encoding='utf-8') as file:  
 ingredients\_data = json.load(file)  
  
# Загрузка данных о рецептах из файла  
with open('recipes.json', 'r', encoding='utf-8') as file:  
 recipes\_data = json.load(file)  
  
user\_ingredients = []  
  
  
# Обработчик команды /start  
@bot.message\_handler(commands=['start'])  
def handle\_start(message):  
 bot.send\_message(message.chat.id,  
 "Привет! Я кулинарный помощник. Давай выберем блюдо! Напиши /help, чтобы узнать больше.")  
  
  
# Обработчик команды /help  
@bot.message\_handler(commands=['help'])  
def handle\_help(message):  
 help\_message = (  
 "Привет! Я кулинарный помощник.\n"  
 "Для начала напиши /ingredients и выбери категорию продуктов.\n"  
 "Затем добавляй продукты в свой список. Когда закончишь, напиши /cook, чтобы я нашел подходящий рецепт.\n"  
 "Если хочешь посмотреть эту справку, напиши /help.\n\n"  
 "Дополнительные команды:\n"  
 "/view\_ingredients - посмотреть текущие ингредиенты в списке\n"  
 "/remove\_ingredient - удалить последний добавленный ингредиент из списка\n"  
 )  
 bot.send\_message(message.chat.id, help\_message)  
  
  
# Обработчик команды /ingredients  
@bot.message\_handler(commands=['ingredients'])  
def handle\_ingredients(message):  
 markup = create\_categories\_markup()  
  
 bot.send\_message(message.chat.id, "Выбери категорию продуктов:", reply\_markup=markup)  
  
  
# Обработчик выбора категории продуктов  
@bot.message\_handler(func=lambda message: message.text in ingredients\_data)  
def handle\_category(message):  
 category = message.text  
  
 # Запрос ингредиентов из выбранной категории  
 markup = telebot.types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 for ingredient in ingredients\_data[category]:  
 markup.add(telebot.types.KeyboardButton(ingredient))  
  
 # Добавление клавиши "Назад"  
 markup.add(telebot.types.KeyboardButton("Назад"))  
  
 bot.send\_message(message.chat.id, f"Добавь в свои продукты из категории '{category}':", reply\_markup=markup)  
  
  
# Функция для создания клавиатуры с категориями продуктов  
def create\_categories\_markup():  
 markup = telebot.types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 for category in ingredients\_data:  
 markup.add(telebot.types.KeyboardButton(category))  
  
 # Добавление кнопок "Выбрать блюдо" и "Показать ингредиенты"  
 markup.row(telebot.types.KeyboardButton("Выбрать блюдо"))  
 if user\_ingredients:  
 markup.row(telebot.types.KeyboardButton("Показать ингредиенты"))  
  
 return markup  
  
  
# Обработчик выбора ингредиентов  
@bot.message\_handler(func=lambda message: message.text in sum(ingredients\_data.values(), []) or message.text == "Назад")  
def handle\_ingredient\_selection(message):  
 if message.text == "Назад":  
 bot.send\_message(message.chat.id, "Выбери категорию продуктов:", reply\_markup=create\_categories\_markup())  
 return  
  
 ingredient = message.text  
  
 # Проверка наличия ингредиента в списке пользователя  
 if ingredient in user\_ingredients:  
 bot.send\_message(message.chat.id, f"Ингредиент '{ingredient}' уже есть в вашем списке.")  
 else:  
 # Добавление выбранного ингредиента в список пользователя  
 user\_ingredients.append(ingredient)  
 bot.send\_message(message.chat.id, f"Ингредиент '{ingredient}' добавлен в список.")  
  
  
# Функция для поиска подходящего рецепта  
def find\_best\_recipe(user\_ingredients):  
 best\_recipe = None  
 max\_common\_ingredients = 0  
  
 for recipe in recipes\_data:  
 common\_ingredients = len(set(user\_ingredients) & set(recipe['ingredients'])) # & множеств  
  
 if common\_ingredients > max\_common\_ingredients:  
 max\_common\_ingredients = common\_ingredients  
 best\_recipe = recipe  
  
 return best\_recipe  
  
  
# Функция для поиска недостающих ингредиентов  
def find\_missing\_ingredients(user\_ingredients, recipe\_ingredients):  
 user\_ingredients\_counter = Counter(user\_ingredients)  
 recipe\_ingredients\_counter = Counter(recipe\_ingredients)  
  
 missing\_ingredients = (recipe\_ingredients\_counter - user\_ingredients\_counter).elements()  
 return list(missing\_ingredients)  
  
  
# Обработчик команды /view\_ingredients  
@bot.message\_handler(commands=['view\_ingredients'])  
def handle\_view\_ingredients(message):  
 if not user\_ingredients:  
 bot.send\_message(message.chat.id,  
 "Ваш список продуктов пока пуст. Добавьте ингредиенты с помощью /ingredients.")  
 else:  
 ingredients\_str = ", ".join(user\_ingredients)  
 bot.send\_message(message.chat.id, f"Ваши текущие ингредиенты: {ingredients\_str}")  
  
  
# Обработчик команды /remove\_ingredient  
@bot.message\_handler(commands=['remove\_ingredient'])  
def handle\_remove\_ingredient(message):  
 if not user\_ingredients:  
 bot.send\_message(message.chat.id,  
 "Ваш список продуктов пока пуст. Добавьте ингредиенты с помощью /ingredients.")  
 else:  
 removed\_ingredient = user\_ingredients.pop()  
 bot.send\_message(message.chat.id, f"Ингредиент '{removed\_ingredient}' удален из вашего списка.")  
  
  
# Обработчик команды "Показать ингредиенты"  
@bot.message\_handler(func=lambda message: message.text == "Показать ингредиенты" and user\_ingredients)  
def handle\_show\_ingredients(message):  
 handle\_view\_ingredients(message)  
  
  
# Обработчик команды "Выбрать блюдо"  
@bot.message\_handler(func=lambda message: message.text == "Выбрать блюдо")  
def handle\_cook(message):  
 bot.send\_message(message.chat.id, "Ищем подходящий рецепт...")  
  
 # Находим подходящий рецепт  
 best\_recipe = find\_best\_recipe(user\_ingredients)  
  
 if best\_recipe:  
 bot.send\_message(message.chat.id, f"Предлагаю приготовить блюдо: {best\_recipe['name']}.")  
 missing\_ingredients = find\_missing\_ingredients(user\_ingredients, best\_recipe['ingredients'])  
  
 if missing\_ingredients:  
 bot.send\_message(message.chat.id, f"Не хватает следующих ингредиентов: {', '.join(missing\_ingredients)}.")  
 else:  
 bot.send\_message(message.chat.id, f"Ингредиентов достаточно, можем приступать!")  
 else:  
 bot.send\_message(message.chat.id, "Извините, не удалось найти подходящий рецепт.")  
  
  
# Обработчик команды /clear  
@bot.message\_handler(commands=['clear'])  
def handle\_clear(message):  
 # Очистка списка user\_ingredients  
 user\_ingredients.clear()  
 bot.send\_message(message.chat.id, "Список ингредиентов очищен.")  
  
  
# Обработчик команды /cook  
@bot.message\_handler(commands=['cook'])  
def handle\_cook(message):  
 bot.send\_message(message.chat.id, "Ищем подходящий рецепт...")  
  
 # Находим подходящий рецепт  
 best\_recipe = find\_best\_recipe(user\_ingredients)  
  
 if best\_recipe:  
 bot.send\_message(message.chat.id, f"Предлагаю приготовить блюдо: {best\_recipe['name']}.")  
 missing\_ingredients = find\_missing\_ingredients(user\_ingredients, best\_recipe['ingredients'])  
  
 if missing\_ingredients:  
 bot.send\_message(message.chat.id, f"Не хватает следующих ингредиентов: {', '.join(missing\_ingredients)}.")  
 else:  
 bot.send\_message(message.chat.id, "Извините, не удалось найти подходящий рецепт.")  
  
  
# Обработчик неизвестных команд  
@bot.message\_handler(func=lambda message: True)  
def handle\_unknown(message):  
 bot.send\_message(message.chat.id, "Извините, я не понимаю ваш запрос. Для справки напишите /help.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 bot.polling(none\_stop=True)

Пример работы























