**Умная кухня**

Разработать прототип интеллектуальной системы приготовления пищи (завтрак, обед, ужин, вечеринка) и управления запасами холодильников в студенческом общежитии. В зависимости от следующих параметров:

- количества порций (планируемого количества человек);

- наличие и количества продуктов (быстро портящихся (в холодильнике), продуктов с большим сроком хранения в разных холодильниках;

- имеющейся посуды и приборов.

Холодильники объединены между собой сетью передачи данных и могут обмениваться информацией.

При решении задачи выдачи рекомендаций на вход подается информация, какие холодильники/владельцы участвуют в кооперации при приготовлении пищи.

Если каких-то продуктов не хватает, то система также оформляет заказ на поставку/приобретение.

В части анализа текста:

В части анализа текста на основе НС-методов подготовить наборы текстов и произвести извлечение существенной информации и знаний для базы знаний рецептов и технологических карт блюд.

Требуется для задачи автоматизированного наполнения базы знаний (из входного текста извлечение правил для базы знаний) разработать метод и его программную реализацию.

Грубые шаги алгоритма при анализе входного фрагмента текста (очередностью надо поэкспериментировать ...):

1)? отбрасывание несущественной информации;

2) выявление сущностей и намерений;

3) выявление причинно-следственных связей для формулирования частей Если (...) и То (...);

4) определение противоречий (расхождения фактов) для отсеивания при построении БЗ.

**1- пользовательский интерфейс**

**Html:**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Умная Кухня</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<h1>Умная Кухня</h1>

</header>

<main>

<section id="product-section">

<h2>Добавление продуктов</h2>

<form id="fruit-form">

<input type="text" id="fruit-name" placeholder="Название фрукта">

<select id="fruit-type">

<option value="переходящий">Переходящий</option>

<option value="не-переходящий">Не-переходящий</option>

</select>

<button type="submit">Добавить фрукт</button>

</form>

<form id="grain-form">

<input type="text" id="grain-name" placeholder="Название зерна">

<select id="grain-type">

<option value="переходящий">Переходящий</option>

<option value="не-переходящий">Не-переходящий</option>

</select>

<button type="submit">Добавить зерно</button>

</form>

<form id="kitchen-item-form">

<input type="text" id="kitchen-item-name" placeholder="Название кухонного предмета">

<input type="number" id="kitchen-item-quantity" min="1" max="100" value="1">

<button type="submit">Добавить кухонный предмет</button>

</form>

<div id="product-list"></div>

</section>

<section id="analyze-section">

<h2>Анализ пользовательского ввода</h2>

<textarea id="user-input" rows="4" cols="50" placeholder="Введите запрос пользователя..."></textarea>

<button onclick="analyzeInput()">Анализировать</button>

<pre id="analysis-result"></pre>

</section>

<section id="suggest-products">

<h2>Предложенные продукты</h2>

<ul id="suggested-products"></ul>

</section>

<section id="check-kitchen-items">

<h2>Проверка кухонных предметов</h2>

<textarea id="recipe-text" rows="4" cols="50" placeholder="Введите рецепт..."></textarea>

<button onclick="checkKitchenItems()">Проверить</button>

<pre id="missing-items"></pre>

</section>

<section id="iot-data">

<h2>Обработка данных IoT</h2>

<table id="iot-table">

<thead>

<tr>

<th>Сенсор ID</th>

<th>Тип сенсора</th>

<th>Значение</th>

</tr>

</thead>

<tbody id="iot-data-body"></tbody>

</table>

</section>

<section id="order-section">

<h2>Отправка заказа</h2>

<input type="number" id="order-id" min="1" max="1000000" value="1">

<input type="text" id="delivery-address" placeholder="Адрес доставки">

<button onclick="sendOrder()">Отправить заказ</button>

<p id="order-status"></p>

</section>

</main>

<footer>

<p>© 2023 Умная Кухня</p>

</footer>

<script src="script.js"></script>

</body>

</html>

**Css:**

body {

font-family: Arial, sans-serif;

line-height: 1.6;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

header {

background-color: #333;

color: #fff;

text-align: center;

padding: 1rem;

}

main {

display: flex;

flex-direction: column;

align-items: center;

padding: 2rem;

}

section {

background-color: #fff;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

margin-bottom: 2rem;

padding: 1.5rem;

width: 80%;

}

h1, h2 {

color: #333;

}

form {

display: flex;

flex-direction: column;

gap: 1rem;

}

input, select, textarea {

padding: 0.5rem;

border: none;

border-radius: 3px;

}

button {

padding: 0.5rem;

background-color: #333;

color: #fff;

border: none;

border-radius: 3px;

cursor: pointer;

}

button:hover {

background-color: #555;

}

#product-list, #suggested-products, #missing-items {

list-style-type: none;

padding-left: 0;

}

pre {

white-space: pre-wrap;

word-wrap: break-word;

}

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 1rem;

}

th, td {

border: 1px solid #ddd;

padding: 0.5rem;

text-align: left;

}

th {

background-color: #f2f2f2;

}

footer {

background-color: #333;

color: #fff;

text-align: center;

padding: 1rem;

position: fixed;

bottom: 0;

left: 0;

right: 0;

}

**Js:**

function showMenu(menuType) {

document.addEventListener('DOMContentLoaded', function() {

let db = new DatabaseManager();

let engine = new AlgorithmEngine();

function addProduct(type) {

let form = document.getElementById(`product-form-${type}`);

let nameInput = form.elements['name'];

let typeSelect = form.elements['type'];

db.add\_product(nameInput.value, typeSelect.value);

updateProductList(type);

nameInput.value = '';

typeSelect.value = '';

}

function updateProductList(type) {

let listElement = document.getElementById(`product-list-${type}`);

listElement.innerHTML = '';

switch(type) {

case 'fruit':

fruits.forEach(fruit => {

let li = document.createElement('li');

li.textContent = `${fruit.name} (${fruit.type}, ${fruit.expiration\_date || 'Без срока'}>`;

listElement.appendChild(li);

});

break;

case 'grain':

grains.forEach(grain => {

let li = document.createElement('li');

li.textContent = `${grain.name} (${grain.type})`;

listElement.appendChild(li);

});

break;

case 'kitchen-item':

kitchenItems.forEach(item => {

let li = document.createElement('li');

li.textContent = `${item.name} (Количество: ${item.quantity})`;

listElement.appendChild(li);

});

break;

}

}

function analyzeInput() {

let input = document.getElementById('user-input').value;

let analysis = engine.analyze\_user\_input(input);

document.getElementById('analysis-result').textContent = JSON.stringify(analysis, null, 2);

}

function suggestProducts() {

let products = engine.suggest\_products(engine.analyze\_user\_input(document.getElementById('user-input').value));

let listElement = document.getElementById('suggested-products');

listElement.innerHTML = '';

products.forEach(product => {

let li = document.createElement('li');

li.textContent = `${product.name} (${product.type}, ${product.expiration\_date || 'Без срока'}`;

listElement.appendChild(li);

});

}

function checkKitchenItems() {

let recipeText = document.getElementById('recipe-text').value;

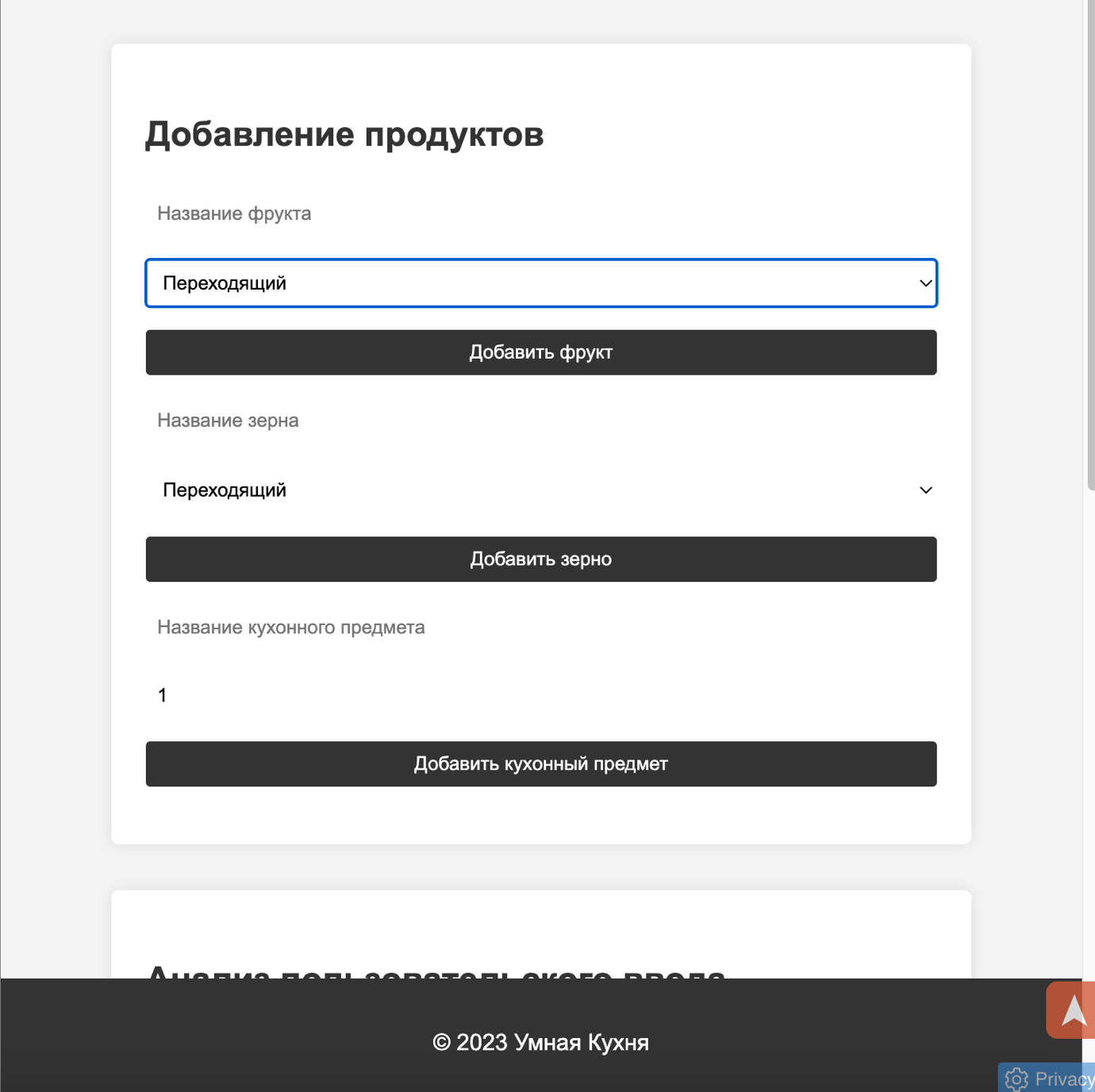
let missingItems = engine.check\_kitchen\_items(recipeText);

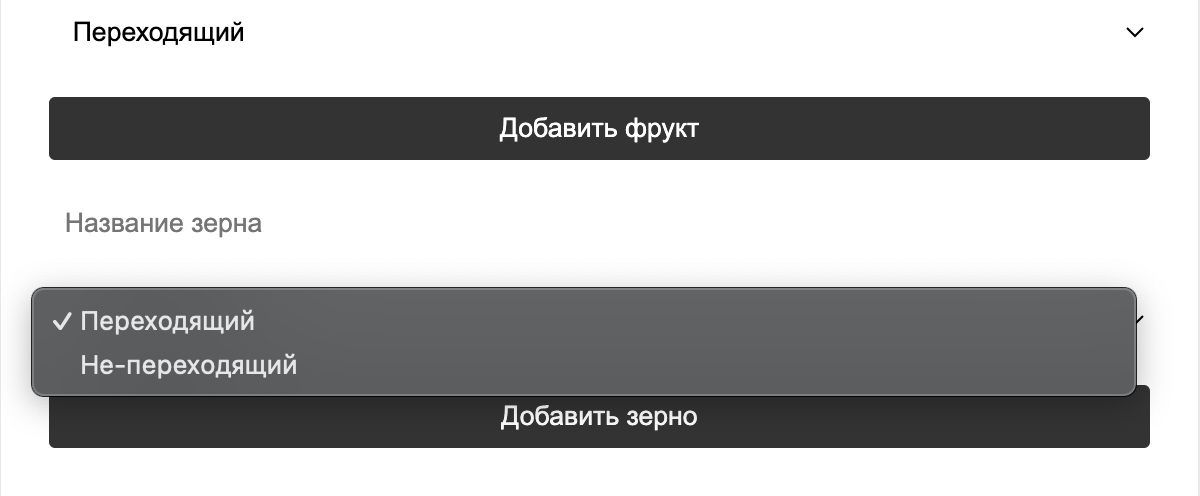
document.getElementById('missing-items').textContent = JSON.stringify(missingItems, null, 2);

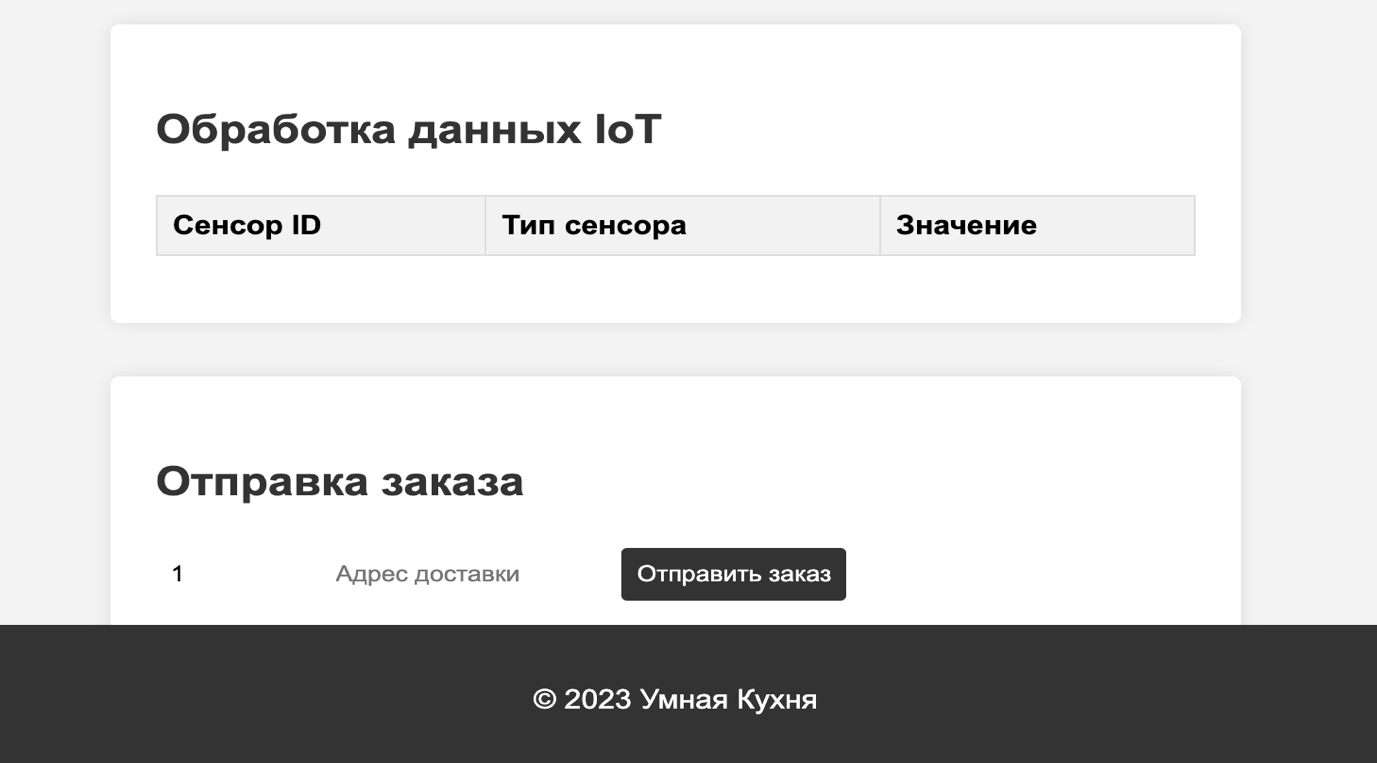
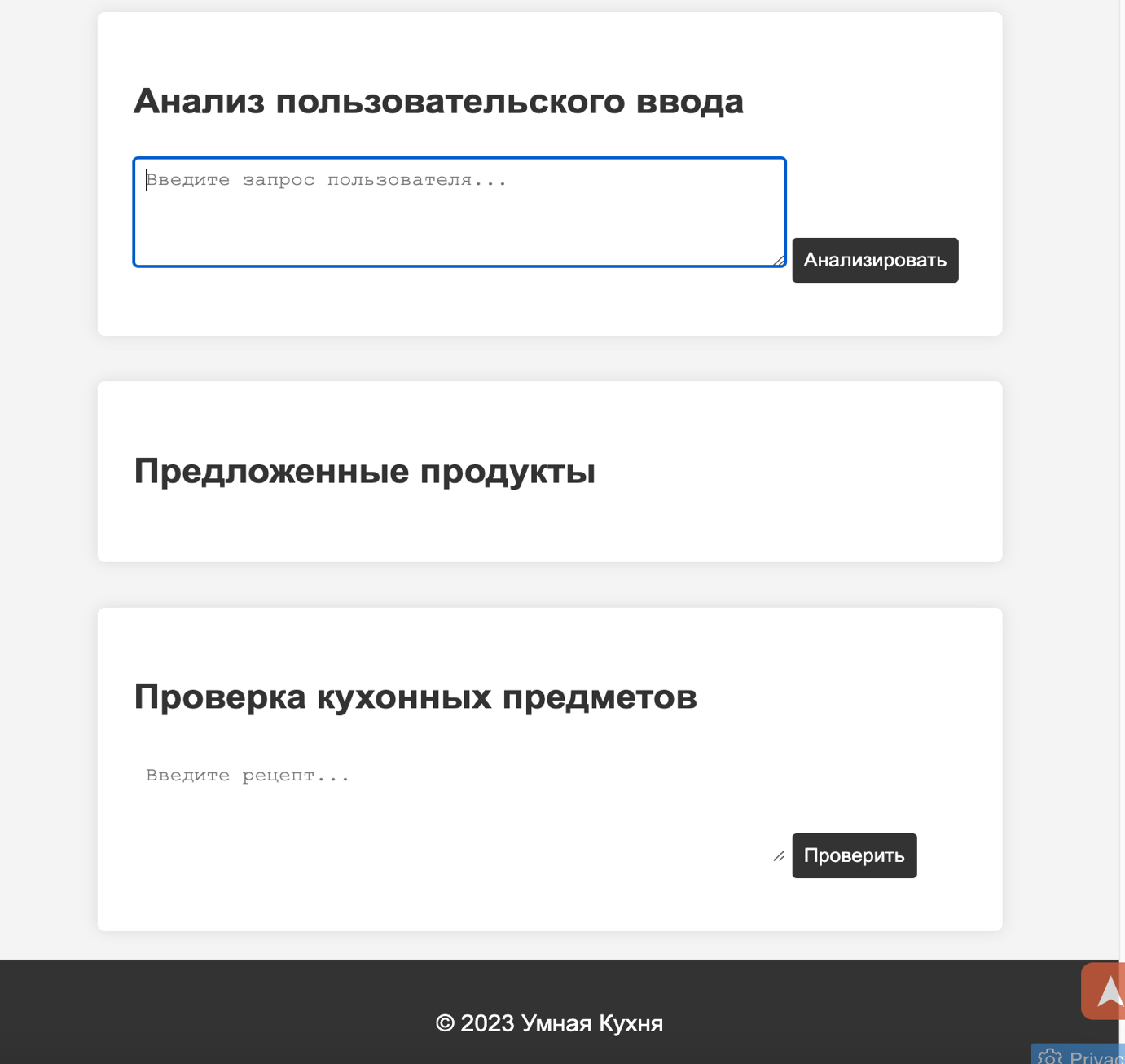
}

function processIo```

**результаты:**







Структура Главной Страницы:

1. Заголовок (Header): В нем написано "Умная Кухня" (Akıllı Mutfak).
2. Основное Содержание (Main): Здесь расположены все важные разделы.
3. Подвал (Footer): Содержит информацию об сайте.

Основные Разделы

1. Добавление Продуктов:
   * Можно добавить фрукты, зёрна и кухонные принадлежности.
   * Для каждого типа есть отдельный формуляр.
2. Анализ Ввода Пользователя:
   * Ожидает ввод от пользователя.
   * Анализирует введенную информацию и показывает результаты.
3. Предложенные Продукты:
   * Показывает продукты, предложенные на основе анализа ввода пользователя.
4. Проверка Кухонских Предметов:
   * Позволяет ввести рецепт для поиска отсутствующих кухонных предметов.
5. Обработка IoT Данных:
   * Можно ввести данные сенсоров и обработать их.
6. Отправка Заказа:
   * Вводится ID продукта и адрес доставки.
   * Показывается статус заказа.

Принцип Работы

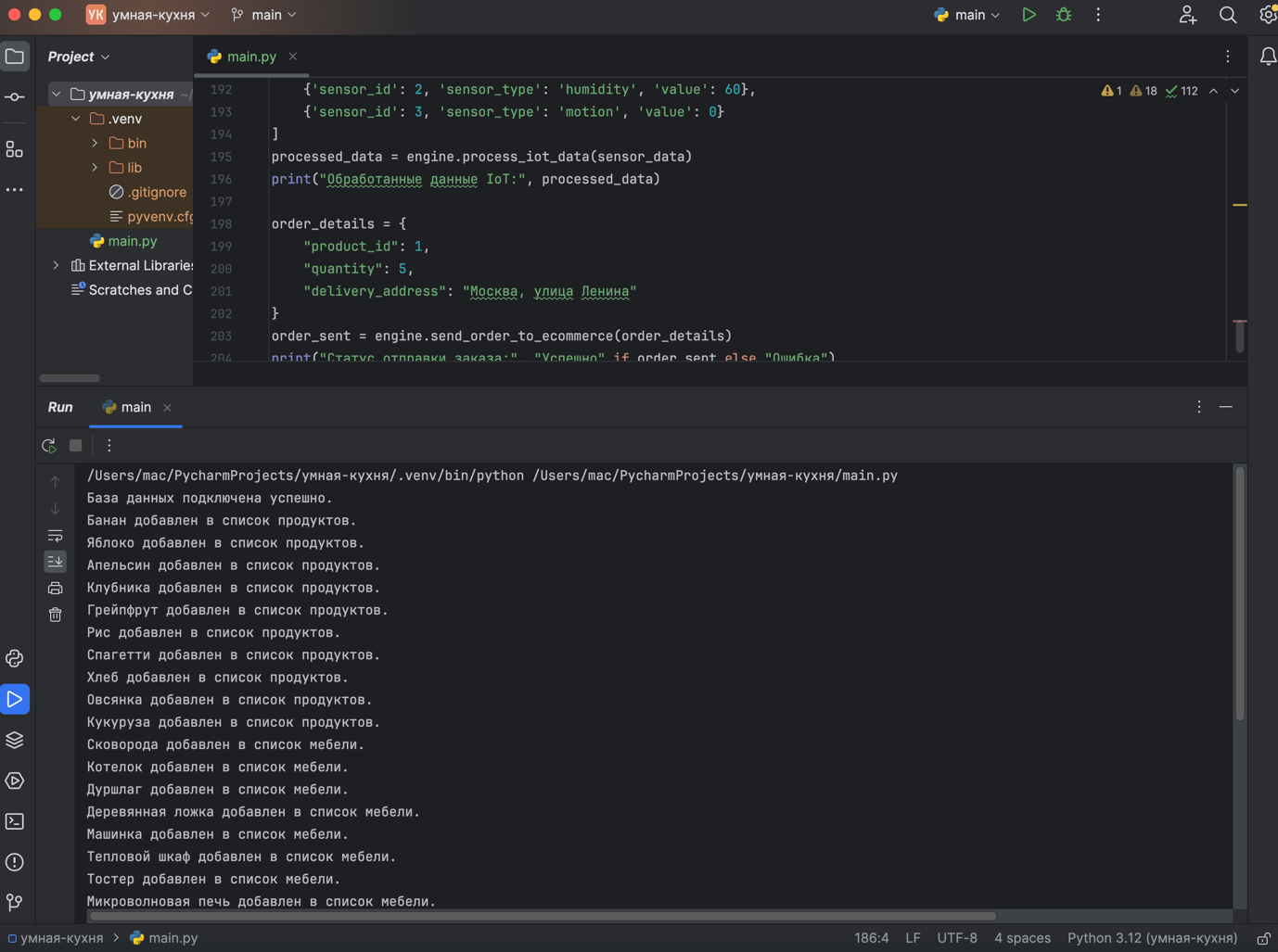
1. Пользователь заполняет формы в различных разделах для ввода данных.
2. JavaScript связывает каждое действие с Python кодом.
3. Вызываются определенные функции в Python, и результаты отображаются в HTML.
4. Пользователь может видеть полученные результаты и выполнять новые действия.

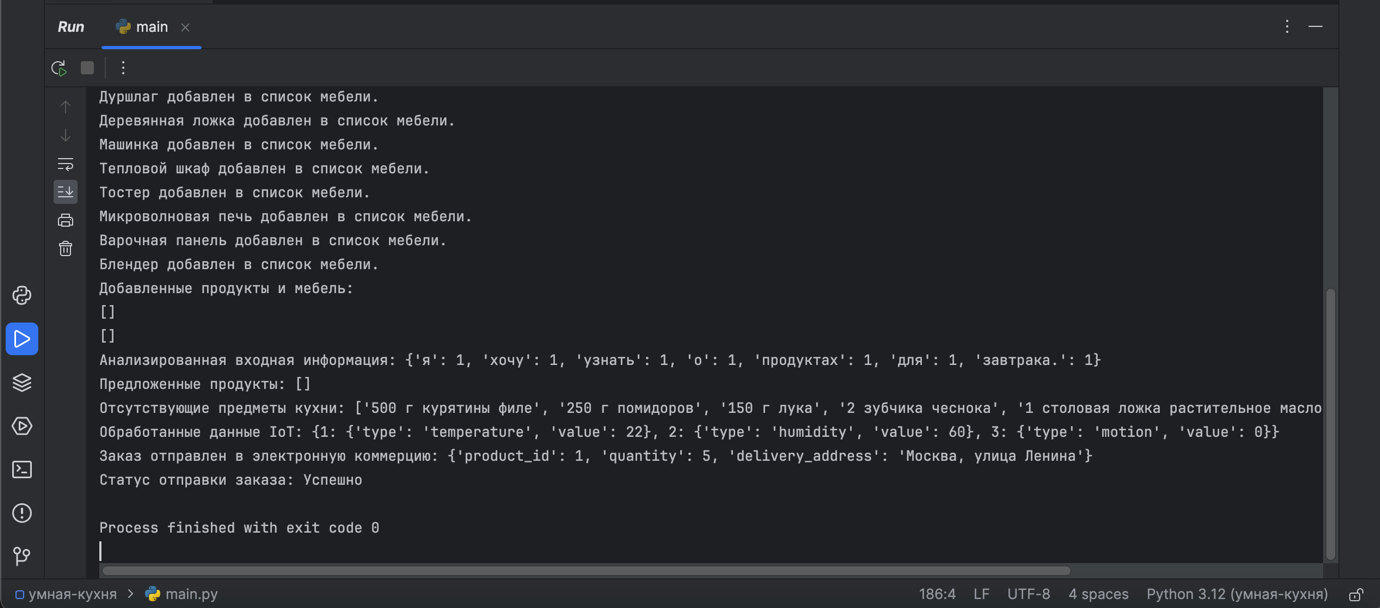
Этот интерфейс позволяет пользователям взаимодействовать с системой умной кухни в интерактивном режиме. Пользователи могут добавлять продукты, анализировать, просматривать предложенные продукты, проверять наличие кухонных предметов, обрабатывать данные IoT и отправлять заказы.

**2- база данных**

import datetime  
  
  
class DatabaseManager:  
 def \_\_init\_\_(self):  
 self.products = {}  
 self.kitchen\_items = {}  
  
 def connect(self):  
 print("База данных подключена успешно.")  
  
 def disconnect(self):  
 print("Подключение к базе данных закрыто.")  
  
 def create\_user(self, username, email, password):  
 # Этот метод не используется в текущей версии  
 pass  
  
 def get\_all\_users(self):  
 # Этот метод не используется в текущей версии  
 return []  
  
 def add\_product(self, name, type, expiration\_date=None):  
 self.products[name] = {  
 'type': type,  
 'expiration\_date': expiration\_date  
 }  
 print(f"{name} добавлен в список продуктов.")  
  
 def get\_all\_products(self):  
 valid\_products = []  
 for name, details in self.products.items():  
 if 'type' in details and 'expiration\_date' in details:  
 valid\_products.append({'name': name, \*\*details})  
 else:  
 print(f"Ошибка: Недостаточно данных для продукта {name}")  
 return valid\_products  
  
 def add\_kitchen\_item(self, name, quantity=1):  
 self.kitchen\_items[name] = {  
 'quantity': quantity  
 }  
 print(f"{name} добавлен в список мебели.")  
  
 def get\_all\_kitchen\_items(self):  
 return list(self.kitchen\_items.values())  
  
  
class AlgorithmEngine:  
 def \_\_init\_\_(self):  
 self.db = DatabaseManager()  
 self.db.connect()  
  
 def analyze\_user\_input(self, input\_text):  
 words = input\_text.lower().split()  
 word\_counts = {}  
 for word in words:  
 if word in word\_counts:  
 word\_counts[word] += 1  
 else:  
 word\_counts[word] = 1  
  
 return word\_counts  
  
 def suggest\_products(self, user\_needs):  
 suggested\_products = []  
  
 all\_products = self.db.get\_all\_products()  
 for product in all\_products:  
 if 'type' not in product or 'expiration\_date' not in product:  
 print(f"Ошибка: Недостаточно данных для продукта {product['name']}")  
 continue  
  
 if product['type'] == 'perishable':  
 if not product['expiration\_date']:  
 suggested\_products.append(product)  
 elif datetime.now() > product['expiration\_date']:  
 suggested\_products.append(product)  
  
 return suggested\_products  
  
 def check\_kitchen\_items(self, recipe):  
 required\_items = self.parse\_recipe(recipe)  
 available\_items = self.db.get\_all\_kitchen\_items()  
  
 missing\_items = [item for item in required\_items if item not in [i['name'] for i in available\_items]]  
  
 return missing\_items  
  
 def parse\_recipe(self, recipe\_text):  
 ingredients = recipe\_text.split('\n')  
 parsed\_ingredients = [ingredient.strip() for ingredient in ingredients if ingredient.strip()]  
 return parsed\_ingredients  
  
 def process\_iot\_data(self, sensor\_data):  
 processed\_data = {}  
 for sensor in sensor\_data:  
 sensor\_id = sensor['sensor\_id']  
 sensor\_type = sensor['sensor\_type']  
 value = sensor['value']  
  
 if sensor\_type == 'temperature':  
 processed\_data[sensor\_id] = {'type': 'temperature', 'value': value}  
 elif sensor\_type == 'humidity':  
 processed\_data[sensor\_id] = {'type': 'humidity', 'value': value}  
 elif sensor\_type == 'motion':  
 processed\_data[sensor\_id] = {'type': 'motion', 'value': value}  
  
 return processed\_data  
  
 def send\_order\_to\_ecommerce(self, order\_details):  
 print(f"Заказ отправлен в электронную коммерцию: {order\_details}")  
 return True  
  
  
def add\_fruits(db):  
 fruits = [  
 ("Банан", "переходящий", None),  
 ("Яблоко", "переходящий", None),  
 ("Апельсин", "переходящий", None),  
 ("Клубника", "переходящий", None),  
 ("Грейпфрут", "переходящий", None)  
 ]  
  
 for fruit in fruits:  
 db.add\_product(fruit[0], fruit[1], fruit[2] if len(fruit) == 3 else None)  
  
  
def add\_grains(db):  
 grains = [  
 ("Рис", "не-переходящий"),  
 ("Спагетти", "не-переходящий"),  
 ("Хлеб", "переходящий", None),  
 ("Овсянка", "не-переходящий"),  
 ("Кукуруза", "не-переходящий")  
 ]  
  
 for grain in grains:  
 db.add\_product(grain[0], grain[1], grain[2] if len(grain) == 3 else None)  
  
  
def add\_kitchen\_items(db):  
 kitchen\_items = [  
 ("Сковорода", 1),  
 ("Котелок", 1),  
 ("Дуршлаг", 1),  
 ("Деревянная ложка", 1),  
 ("Машинка", 1),  
 ("Тепловой шкаф", 1),  
 ("Тостер", 1),  
 ("Микроволновая печь", 1),  
 ("Варочная панель", 1),  
 ("Блендер", 1)  
 ]  
  
 for item in kitchen\_items:  
 db.add\_kitchen\_item(item[0], item[1])  
  
  
# Использование примера  
db = DatabaseManager()  
engine = AlgorithmEngine()  
  
add\_fruits(db)  
add\_grains(db)  
add\_kitchen\_items(db)  
  
print("Добавленные продукты и мебель:")  
print(engine.db.get\_all\_products())  
print(engine.db.get\_all\_kitchen\_items())  
  
# Пример использования  
user\_input = "Я хочу узнать о продуктах для завтрака."  
analyzed\_input = engine.analyze\_user\_input(user\_input)  
print("Анализированная входная информация:", analyzed\_input)  
suggested\_products = engine.suggest\_products(analyzed\_input)  
print("Предложенные продукты:", suggested\_products)  
  
recipe = """  
500 г курятины филе  
250 г помидоров  
150 г лука  
2 зубчика чеснока  
1 столовая ложка растительное масло  
1 столовая ложка томатной пасты  
"""  
missing\_items = engine.check\_kitchen\_items(recipe)  
print("Отсутствующие предметы кухни:", missing\_items)  
  
sensor\_data = [  
 {'sensor\_id': 1, 'sensor\_type': 'temperature', 'value': 22},  
 {'sensor\_id': 2, 'sensor\_type': 'humidity', 'value': 60},  
 {'sensor\_id': 3, 'sensor\_type': 'motion', 'value': 0}  
]  
processed\_data = engine.process\_iot\_data(sensor\_data)  
print("Обработанные данные IoT:", processed\_data)  
  
order\_details = {  
 "product\_id": 1,  
 "quantity": 5,  
 "delivery\_address": "Москва, улица Ленина"  
}  
order\_sent = engine.send\_order\_to\_ecommerce(order\_details)  
print("Статус отправки заказа:", "Успешно" if order\_sent else "Ошибка")

**результаты:**





Код Проекта Умной Кухни:

Этот код написан для проекта "Умная Кухня" . Основные характеристики проекта следующие:

Управление Базой Данных:

Класс DatabaseManager управляет базой данных продуктов и кухонных предметов. Добавляет продукты в базу данных. Извлекает все продукты из базы данных. Извлекает кухонные предметы из базы данных.

Мотор Алгоритма:

Класс AlgorithmEngine анализирует входящие данные пользователя и предлагает рекомендации. Анализирует запросы пользователя. Определяет предложенные продукты. Проверяет наличие недостающих кухонных предметов. Обрабатывает данные сенсоров IoT. Отправляет заказ в систему электронной торговли.

Функции Добавления Продуктов:

Функции add\_fruits, add\_grains и add\_kitchen\_items добавляют различные виды товаров (фрукты, зёрна, кухонные предметы) в базу данных.

Системы Анализа и Предложений:

Анализирует входящие данные пользователя и предлагает соответствующие товары. Находит недостающие кухонные предметы по рецепту.

Интеграция IoT:

Обрабатывает данные сенсоров и интерпретирует их.

Обработка Заказов:

Отправляет заказ в систему электронной торговли.

Цель всего кода - создание интегрированного умного управления кухней, который предлагает рекомендации в соответствии с потребностями пользователя, интегрируется с IoT и работает с системой электронной торговли. Пользователи могут получать предложения на основе своих запросов или текущего состояния, находить недостающие кухонные предметы и отправлять необходимые заказы.

ВЫХОД:

Проект Разработки Результаты и Оценка

Цель проекта и Основные Структуры

Целью нашего проекта было создание системы управления умной кухней. Эта система должна была предлагать рекомендации в соответствии с потребностями пользователя, интегрироваться с IoT и работать с электронным магазином.

В процессе разработки мы создали следующие основные компоненты:

1. Система управления базой данных
2. Мотор алгоритма
3. Интерфейс пользователя

Эти компоненты образовали интегрированную систему.

Шаги разработки и Прогресс

1. Создание системы управления базой данных:
   * Был разработан класс DatabaseManager.
   * Реализована конфигурация для хранения информации о продуктах и кухонных предметах.
2. Разработка мотора алгоритма:
   * Создан класс AlgorithmEngine.
   * Написаны функции для анализа входных данных и предложения рекомендаций.
   * Добавлены модули для обработки данных IoT и интеграции с системой электронной торговли.
3. Разработка интерфейса пользователя:
   * Создан интерактивный интерфейс с использованием HTML, CSS и JavaScript.
   * Пользователи теперь могут взаимодействовать с системой через интерфейс.
4. Добавление функциональности:
   * Добавлена возможность добавления продуктов, анализ, предложение рекомендаций, проверка кухонных предметов и отправка заказов.
5. Интеграция:
   * Функциональность Python, созданная для обработки данных, была подключена к веб-интерфейсу.

Результаты и Будущие Разработки

Проект достиг всех запланированных целей и стал функциональным. Пользователи теперь имеют возможность взаимодействовать с системой умной кухни через интерактивный интерфейс.

Для будущих разработок мы можем рассмотреть следующие направления:

1. Улучшение пользовательского опыта
2. Оптимизация производительности
3. Интеграция с дополнительными сценариями (например, распознавание голоса, интеграция мобильного приложения)
4. Усиление мер безопасности

Этот проект является хорошим примером практического применения технологий умного дома и создал прочную основу для будущих разработок.

АЙЙЫЛДЫЗ ГЮЛИН

РИ-320942