

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра  
інформатики та програмної інженерії  
(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з «Основи програмування — 2. Методології програмування»

(назва дисципліни)

на тему: Створення програми доставки їжі «Green Food»

Студентка 1-го курсу, групи ІІІ-22

Андреева Уляна Андріївна

Спеціальності 121 «Інженерія програмного  
забезпечення»

Керівник ст. вик. Головченко М. М.

(посада, вчене звання, науковий  
ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_

Національна оцінка \_\_\_\_\_

Члени комісії

\_\_\_\_\_

(підпис)

к. т. н., доц. Муха І. П.

(вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_

(підпис)

асистент Вовк Є. А.

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2023 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

---

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрямок "ІІЗ"

Курс 1

Група

ІІ-22

Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Андреєвої Уляни Андріївни

---

(прізвище, ім'я, по батькові)

1. Тема роботи Створення програми доставки їжі «GreenFood»

2. Строк здачі студентом закінченої роботи 31.05.2023

3. Вихідні дані до роботи Додаток «А» Технічне завдання

---

---

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Постановка задачі, теоретичні відомості, опис алгоритмів, опис програмного забезпечення, тестування програмного забезпечення, інструкція користувача.

5. Перелік графічного матеріалу ( з точним зазначенням обов'язкових креслень )

---

---

6. Дата видачі завдання 12.02.2023

## КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	12.02.2023	
2.	Підготовка ТЗ	20.02.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	21.02.2023	
4.	Розробка сценарію роботи програми	03.03.2023	
5.	Узгодження сценарію роботи програми з керівником	05.03.2023	
6.	Розробка (вибір) алгоритму рішення задачі	10.03.2023	
7.	Узгодження алгоритму з керівником	13.03.2023	
8.	Узгодження з керівником інтерфейсу користувача	27.03.2023	
9.	Розробка програмного забезпечення	03.04.2023	
10.	Налагодження розрахункової частини програми	07.04.2023	
11.	Розробка та налагодження інтерфейсної частини програми	11.04.2023	
12.	Узгодження з керівником набору тестів для контрольного прикладу	17.04.2023	
13.	Тестування програми	21.04.2023	
14.	Підготовка пояснювальної записки	16.05.2023	
15.	Здача курсової роботи на перевірку	31.05.2023	
16.	Захист курсової роботи	06.06.2023	

Студент \_\_\_\_\_  
(підпис)

Керівник \_\_\_\_\_  
(підпис)

Головченко М. М.  
(прізвище, ім'я, по батькові)

"12" лютого 2023 р.

## АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 91 сторінка, 14 рисунків, 16 таблиць, 4 посилання.

Мета роботи: підвищення зручності процесу замовлення їжі через мобільний застосунок «GreenFood».

Вивчено методи роботи з базою даних Firebase та методи підходів до створення графічного інтерфейсу в мобільних застосунках.

Виконана програмна реалізація алгоритму бінарного пошуку, фільтрації та лінійного пошуку страв. Розроблений мобільний застосунок для взаємодії з користувачем через базу даних, яка виконує функцію сховища інформації замовлення.

СИСТЕМА ДОСТАВКИ, ЗАМОВЛЕННЯ, СТРАВА, ГЕОПОЗИЦІЯ,  
БАЗА ДАНИХ, ВИБІР, МОБІЛЬНИЙ ЗАСТОСУНОК.

## ЗМІСТ

ВСТУП.....	5
1 ПОСТАНОВКА ЗАДАЧІ.....	6
2 ТЕОРЕТИЧНІ ВІДОМОСТІ.....	7
3 ОПИС АЛГОРИТМІВ.....	9
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	12
4.1. Діаграма класів програмного забезпечення.....	12
4.2. Опис методів частин програмного забезпечення.....	13
4.2.1. Стандартні методи.....	13
4.2.2. Користувацькі методи.....	22
У таблиці 4.2 наведено користувацькі методи, створені в.....	22
5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	31
5.1 План тестування.....	31
5.2 Приклади тестування.....	32
6 ІНСТРУКЦІЯ КОРИСТУВАЧА.....	41
6.1. Робота з програмою.....	41
6.2. Формат вхідних та вихідних даних.....	53
6.3. Системні вимоги.....	54
ВИСНОВКИ.....	55
ПЕРЕЛІК ПОСИЛАНЬ.....	56
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ.....	57
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ.....	61

## ВСТУП

У минулі роки додатки для доставки їжі стали надзвичайно відомими, особливо під час пандемії COVID-19, коли люди були обмежені у своїй можливості відвідувати ресторани та кафе. Це призвело до стрімкого зростання ринку доставляння їжі, і цей тренд очікується і надалі. Додаток "Green Food" має потенціал стати популярним серед тих, хто бажає замовляти здорову їжу з можливістю постачання. Він спрощує вибір для користувачів та допомагає заощадити час.

Програми доставки їжі найкраще розробляти у формі мобільних застосунків з кількох причин. По-перше, мобільні пристрої, такі як смартфони і планшети, стали неодмінною частиною життя для багатьох людей. Вони завжди з ними, доступні в будь-який час і дозволяють отримувати інформацію та здійснювати операції на ходу.

По-друге, мобільні застосунки забезпечують зручність і швидкість використання. Вони можуть працювати в автономному режимі і не вимагають постійного підключення до Інтернету, що дозволяє користувачам замовляти їжу навіть за умов відсутності стабільного зв'язку.

Отже, мобільні додатки для доставки їжі є ідеальним варіантом, оскільки вони забезпечують доступність, зручність, інтерактивність і персоналізацію. Вони стають підходящим вибором для користувачів, які шукають здорову їжу та зручний спосіб її замовлення.

## 1 ПОСТАНОВКА ЗАДАЧІ

Розробити програму для доставки їжі, яка буде підвищувати зручність процесу через мобільний застосунок «Green Food».

Вхідними даними для даної роботи є база даних меню, яка знаходиться в Firebase: *dish\_description* – опис страви, *dish\_ingredient* – перелік інгредієнтів, *dish\_name* – назва страви, *dish\_photo* – фото страви, *dish\_price* – ціна страви, *dish\_rank* – кількість зірок страви – це все міститься у колекції Items. Програмне забезпечення повинно обробляти запити користувача по замовленню та передавати всю інформацію до бази даних.

Вихідними даними для даної роботи являється сукупність інформації, яка є динамічною у базі даних та містить у собі інформацію про користувача та його замовлення. Програмне забезпечення також повинно зчитувати геолокацію користувача за його згодою. Якщо користувач не надав дозвіл на використання геолокації, програма повинна вивести повідомлення про неможливість використання цієї функції та вимкнути її у налаштуваннях для даного додатку.

## 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Додатки для доставки їжі стали надзвичайно популярними в останні роки, особливо під час пандемії COVID-19, коли люди були обмежені в своїй здатності відвідувати ресторани і кафе. Це зробило ринок доставки їжі одним з найбільш швидко зростаючих ринків, і цей тренд очікується зберегтися в майбутньому. Додаток «Green Food» має потенціал стати популярним серед тих, хто хоче замовляти здорову їжу з можливістю доставки. Він полегшує вибір користувача та дозволяє зекономити час.

Всього за кілька кліків, покупець може обрати бажану страву, вказати адресу доставки і зробити замовлення. Завдяки цьому додатку, зекономлений час можна витратити на інші важливі справи або просто на відпочинок.

Робота програми починається після дозволу користувача на встановлення географічного місцезнаходження.

Після цього, програмне забезпечення пропонує користувачу ознайомитися з меню ресторану – цінами, рейтингом, інгредієнтами, описом та світлинами, які першочергово знаходяться в базі даних та є зручними для зміни у будь-який момент. Для реалізації пошуку страв в меню можна використати пошукове поле. Також присутнє сортування страв за алфавітним порядком та в протилежному.

Наступним кроком є обрання страв за вподобаннями користувача та доданням їх у корзину.

Коли користувач обрав усе бажане з меню, він може ознайомитися зі своїм замовленням, фінальною ціною, а також змінити кількість страв або ж навіть видалити те, що вважає непотрібним. Після виконання цієї дії потрібно підтвердити своє замовлення й тоді інформація про покупку автоматично передається до бази даних та з'являється сповіщення, що замовлення було оброблене коректно, якщо ж клієнт змінив свою думку, то можна просто скасувати замовлення, що призведе до автоматичного видалення з бази даних. На даному етапі програма завершує свою роботу.



Додатковими функціями є:

- Перегляд сторінки ресторану в соціальній мережі Instagram.
- Ознайомлення з контактною інформацією та геолокацією ресторану на карті.
- Перегляд сторінки зі знижками, промо-роліками та корисною інформацією ресторану.

### 3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних і їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 — Основні змінні та їхні призначення

Змінна	Призначення
<code>orderedDishes</code>	Булева змінна, яка показує, чи були замовлені страви
<code>cartItems</code>	Масив об'єктів типу <code>Cart</code> , який містить дані про продукти, що додані до кошика
<code>search</code>	Містить текст для пошуку певних елементів
<code>filteredData</code>	Містить дані про продукти, що відповідають критеріям пошуку
<code>items</code>	Масив об'єктів типу <code>Item</code> , який містить дані про продукти
<code>isCartIndex</code>	Ознака знаходження елемента з заданим <code>id</code> у масиві (повернення 0 – елемент не знайдений, повернення <code>index</code> – елемент знайдений)

#### 3.1. Алгоритм бінарного пошуку

##### 1. ПОЧАТОК

##### 2. Ініціалізація змінних *index* та *cartIndex*:

2.1. *index* = *nil*.

2.2. *cartIndex* = *nil*.

##### 3. ЦИКЛ проходу по масиву *items*:

3.1. ЯКЩО *items[i].id* == *item.id*, ТО *index* = *i*.

3.2. Вийти з циклу.

##### 4. ЦИКЛ проходу по масиву *cartItems*:

4.1. ЯКЩО *cartItems[i].item.id* == *item.id*, ТО *cartIndex* = *i*.

4.2. Вийти з циклу.

##### 5. ПОВЕРНУТИ *isCartIndex?* *cartIndex* : *index*.

##### 6. КІНЕЦЬ

### 3.2. Алгоритм фільтрації

#### 1. ПОЧАТОК

2. Виклик функції `filteringData()` з параметром *search* - рядком, за яким будемо шукати елементи в масиві *items*.

3. Створення порожнього масиву *filteredData*.

4. Виклик методу `filter()` на масиві *items* з передачею замикання (*item*) -> Bool, яке відповідає за фільтрацію. Результат фільтрування зберігається у змінну *filteredItems*.

5. Ініціалізація змінної *filteredItems* = 0.

6. ЦИКЛ проходу по масиву *items*.

6.1. ЯКЩО рядок *dish\_name* поточного елемента масиву *items* містить рядок *search*, ТО повернути true, що означає, що цей елемент має бути доданий до масиву *filteredItems*.

6.2. ІНАКШЕ повернути false.

7. ЯКЩО результат фільтрування *filteredItems* не порожній, ТО ініціалізація змінної *i* зі значенням 0.

8. ЦИКЛ проходу по масиву *filteredItems*.

8.1. Додати поточний елемент масиву *filteredItems* до масиву *filteredData* за допомогою методу `append()`.

8.2. Збільшити значення змінної *i* на 1.

9. ПОВЕРНУТИ масив *filteredData* з відфільтрованими елементами.

#### 10. КІНЕЦЬ

### 3.3. Алгоритм сортування бульбашкою

#### 1. ПОЧАТОК

2. Ініціалізувати змінну *n*, що містить кількість елементів у списку, який потрібно відсортувати.

3. ЦИКЛ від 0 до *n*-1 з кроком 1:

3.1. ЦИКЛ від 0 до *n*-2 з кроком 1:

3.1.1. ЯКЩО поточний елемент більший за наступний елемент, ТО поміняти їх місцями.

4. КІНЕЦЬ

3.4. Алгоритм лінійного пошуку

1. ПОЧАТОК

2. Ініціалізація змінної  $index = -1$ .

3. ЦИКЛ проходу по масиву  $arr$ :

3.1. ЯКЩО  $arr[i] == key$ , ТО  $index = i$ .

3.2. Вийти з циклу.

4. ПОВЕРНУТИ  $index$ .

5. КІНЕЦЬ

## 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1. Діаграма класів програмного забезпечення

На рисунку 4.1 зображена діаграма класів програми. На ній зображені наступні класи:

- NSObject – функціональні можливості для об'єктно-орієнтованого програмування.
- UIApplicationDelegate – виступає в ролі делегата (delegate) додатку і отримує повідомлення про різні події, які стосуються стану додатку.
- AppDelegate – відповідає за керування головним циклом життя додатка, обробку системних подій і взаємодію з основними об'єктами додатка.
- App – ініціалізація додатка, обробка подій системи та управління станом додатка.
- GreenFoodOrderApp – виконує протокол App.
- HomeViewModel – включає в себе функціональність для управління домашньою сторінкою додатка.
- Cart – включає в себе протоколи товарів доданих у кошик.
- Item – пункт меню з відповідними властивостями.
- ContentView – описує вміст сторінки додатку.
- Home – описує головний екран застосунку.
- ItemView – відображає деталі страв на екрані.
- CartView – відображає замовлення корзини.
- MainMenu – описує вміст декількох екранів програми.

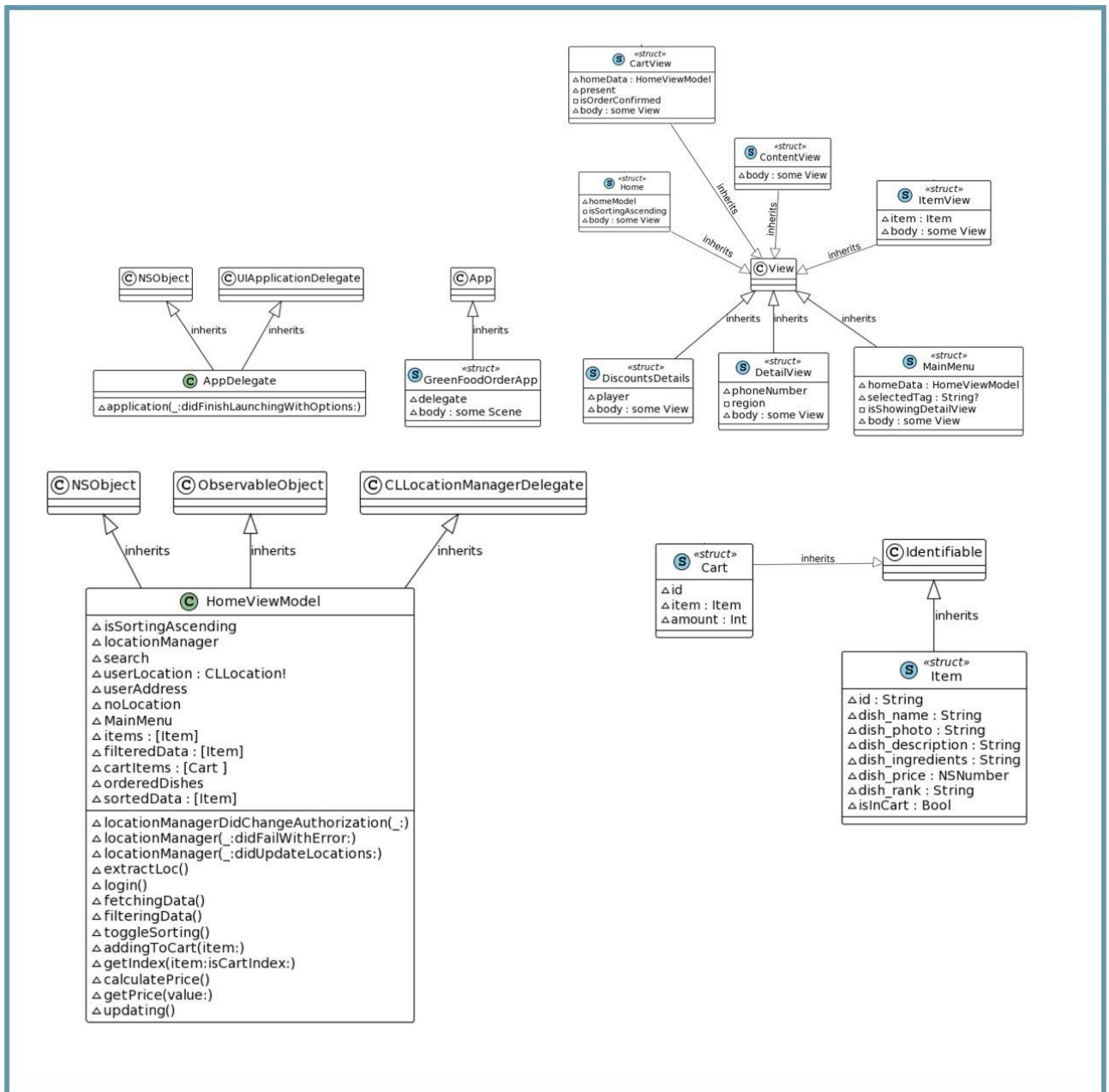


Рисунок 4.1 — Діаграма класів

## 4.2. Опис методів частин програмного забезпечення

### 4.2.1. Стандартні методи

У таблиці 4.1 наведено стандартні методи, створені в програмному забезпеченні.

Таблиця 4.1 — Стандартні методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	View	padding	Створення проміжку між вмістом представлення та його рамкою або іншими представлення ми.	Параметри розміщення об'єктів програми.	Не повертає вихідних параметрів.
2	View	shadow	Встановлення тіней.	Параметри стилю об'єктів програми.	Не повертає вихідних параметрів.
3	View	font	Встановлення шрифту.	Параметри стилю оформлення строки.	Не повертає вихідних параметрів.
4	View	fontWeight	Встановлення товщини шрифту.	Параметри стилю оформлення строки.	Не повертає вихідних параметрів.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
5	View	frame	Створення рамки.	Параметри стилю обрамлення об'єктів програми.	Не повертає вихідних параметрів.
6	View	foregroundColor	Встановлення кольору.	Параметри стилю оформлення строки.	Не повертає вихідних параметрів.
7	View	background	Встановлення фону.	Об'єкти програми.	Не повертає вихідних параметрів.
8	View	cornerRadius	Встановлення заокруглення на кнопки.	Параметри стилю кнопки.	Не повертає вихідних параметрів.
9	View	Spacer	Автоматичне заповнення доступного простору в контейнері.	Параметри розміщення об'єктів програми.	Не повертає вихідних параметрів.



Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
10	View	NavigationLink	Створення навігаційних посилань між різними виглядами (вікнами) в додатку.	destination – вигляд до якого користувач буде перенаправлений.	Navigation Link повертає представлення (View), яке створює посилання.
11	UIApplication	UIApplication.shared.open	Відкриття зовнішнього URL-адреси в додатку iOS.	Відповідна URL-адреса.	Не повертає вихідних параметрів.
12	AVFoundation	AVFoundation	Відтворення відео.	Відповідна URL-адреса.	Не повертає вихідних параметрів.
13	View	ScrollView	Прокрутка вмісту на екрані.	Content – дочірній вид, що буде прокручуватися.	Повертає дочірній вид (View), який вміщений всередині нього.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
14	—	DispatchQueue. main.async After	Асинхронна функція затримки, яка виконується на черзі (dispatch queue) головного потoku (main thread) після певного проміжку часу.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
15	View	ignoresSafeArea	Ігнорування безпечної області екрана.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
16	View	withAnimation	Використовується для анімації змін в інтерфейсі користувача.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
17	View	bold	Встановлення товстого шрифту.	Параметри стилю оформлення строки.	Не повертає вихідних параметрів.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	View	Divider	Створює роздільну лінію для візуального розділення елементів інтерфейсу користувача.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
19	View	onTapGesture	Обробник події при торканні елементу.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
20	—	offset	Зміщує позицію елементу.	х, у: значення типу CGFloat (вказують зміщення по горизонтальній та вертикальній вісях).	Не повертає вихідних параметрів.
21	View	border	Встановлення рамки.	Параметри стилю оформлення кнопки.	Не повертає вихідних параметрів.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	View	lineLimit	Обмеження кількості ліній тексту, що можуть бути відображені.	Рядок.	Обмежени й рядок.
19	View	multilineText Alignment	Встановлення вирівнювання багаторівнево го тексту.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
20	View	navigationBar Hidden	Приховує навігаційну панель.	Булеве значення.	Не повертає вихідних параметрів.
21	View	navigationBar BackButtonHi dden	Приховання кнопки «Назад» у модифікаційні й моделі.	Булеве значення.	Не повертає вихідних параметрів.
22	View	Alert	Відображення спливаючого вікна.	Повідомлення й заголовок повідомлення.	Повертає попередже ння для клієнта.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
23	View	WebImage	Автоматичне завантаження зображення з вказаної URL-адреси та відображення його.	URL-адреса зображення, яке потрібно завантажити.	завантаження та відображення зображення з вказаної URL-адреси.
24	View	overlay	Додає додатковий контент або розміщує інший вид (View) поверх вихідного виду.	Вид, який буде розміщено поверх вихідного файлу.	Не повертає вихідних параметрів.
25	View	mask	Вирізання або обмеження видимої області.	Вид для вирізання точної області.	Не повертає вихідних параметрів.
26	Image	Image	Створює зображення.	Системний символ або ж власне зображення.	Не повертає вихідних параметрів.

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
27	ViewModi fier	resizable	Дозволяє змінювати розмір віджету за допомогою маніпуляцій користувача.	Не приймає вхідних параметрів.	Не повертає вихідних параметрів.
28	ViewModi fier	onChange	Визначення дій, які мають відбутися, коли відбувається зміна певного значення або стану.	Значення, яке потрібно змінити.	Не повертає вихідних параметрів.
29	ViewModi fier	onAppear	Виконання певних дій, коли вигляд (view) з'являється на екрані.	Блок коду, який ви хочете виконати при з'яві вигляду.	Повертає модифіков аний вигляд (some View)
30	ViewModi fier	environmentO bject	Передачі об'єкту в середовище вигляду(view).	Об'єкт, який потрібно передати до середовища.	Повертає модифіков аний View.

#### 4.2.2. Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, створені в програмному забезпеченні.

Таблиця 4.2 — Користувацькі методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	ContentView	body	Опис вигляду та інтерфейсу користувацької сторінки.	Не має вхідних параметрів.	Повертає об'єкт типу some View, який представляє вигляд сторінки.
2	ContentView _Previews	previews	Перегляд попереднього відображення сторінки.	Не має вхідних параметрів.	Повертає Об'єкт типу some View.

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
3	AppDelegate	application	Конфігурування Firebase та ініціалізація додатку при запуску.	Application: об'єкт типу UIApplication, який представляє додаток, launchOptions — опціональний словник, що містить параметри запуску додатку.	Повертає булеве значення, що вказує, чи було успішно завершено запуск додатку.
4	DiscountsDetails	body	Опис вигляду та інтерфейсу сторінки деталей знижок.	Не має вхідних параметрів.	Повертає об'єкт типу some View, який представляє вигляд сторінки знижок.



Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
5	DetailView	body	Опис вигляду та інтерфейсу сторінки з контактною інформацією.	Не має вхідних параметрів.	Повертає об'єкт типу some View, який представляє вигляд сторінки з контактною інформацією.
6	MainMenu	body	Опис вигляду та інтерфейсу головного меню.	«homeData» — об'єкт типу «HomeView Model», який використов ується в головному меню.	Повертає об'єкт типу «some View», який представляє вигляд головного меню.

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
7	Home	body	Опис вигляду та інтерфейсу домашньої сторінки.	Не має вхідних параметрів.	Не повертає вихідних параметрів.
8	ItemView	body	Опис вигляду окремого елемента страви.	Параметр item типу Item, який представляє об'єкт страви.	Повертає об'єкт типу some View, який представляє вигляд елемента страви.

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
9	CartView	body	Опис вигляду корзини замовлень.	Параметри «homeData» типу «HomeView Model» (об'єкт моделі домашньої сторінки) і «present» типу «Binding» до «presentation Mode» (параметр оточення для управління відображенн ям).	Повертає об'єкт типу some View, який представля є вигляд корзини замовлень.
10	HomeViewModel	sortedData	Повертає відсортований масив елементів (Item).	Не має вхідних параметрів.	Повертає відсортова ний масив елементів ([Item]).

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
11	HomeViewModel	toggleSorting	Змінює напрямок сортування (зі зростання на спадання або навпаки).	Не має вхідних параметрів.	Не має вихідних параметрів.
12	HomeViewModel	addingToCart	Додає або видаляє елемент (Item) з кошика покупок. Оновлює дані про елементи у списку та відфільтрован ому списку.	item: елемент (Item), який додається або видаляється з кошика.	Повертає відсортован ий масив елементів ([Item]).
13	HomeViewModel	getIndex	Повертає індекс елемента (Item) у списку items або у списку cartItems.	Елемент (Item), індекс. isCartIndex: булеве значення, чи потрібно шукати індекс.	Повертає індекс елемента (Int).

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
14	HomeViewModel	calculatePrice	Обчислює загальну вартість елементів у кошику покупок.	Не має вхідних параметрів.	Повертає вартість у вигляді рядка (String).
15	HomeViewModel	getPrice	Форматує числове значення (Float) у формат вартості (курсу).	value: числове значення, яке потрібно отримати у форматі вартості.	Повертає вартість у вигляді рядка (String).
16	HomeViewModel	updating	Оновлює дані у базі даних.	Не має вхідних параметрів.	Не має вихідних параметрів.
17	User	locationManager DidChangeAuthorization	Обробляє зміну статусу авторизації локації. Виконує дії залежно від статусу авторизації.	manager: об'єкт CLLocationManager, який управляє локацією.	Не має вихідних параметрів.

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	User	locationManager	Обробляє помилки, пов'язані з локацією.	manager: об'єкт CLLocation Manager, який управляє локацією.  error: помилка, пов'язана з локацією.	Не має вихідних параметрів.
19	User	extractLoc	Отримує адресу на основі отриманої локації користувача.	Не має вхідних параметрів.	Не має вихідних параметрів.
20	User	login	Авторизується анонімно в системі.	Не має вхідних параметрів.	Не має вихідних параметрів.

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
21	User	fetchingData	Отримує дані з бази даних Firebase та ініціалізує об'єкти типу Item.	Не має вхідних параметрів.	Не має вихідних параметрів.
22	User	filteringData	Фільтрує дані за певним критерієм.	Не має вхідних параметрів.	Не має вихідних параметрів.
23	GreenFoodOrderApp	body	Описує вигляд додатку та інтерфейс.	Не має вхідних параметрів.	Об'єкт типу Scene, який є сценою додатку.

## 5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 5.1 План тестування

Для подальшого проведення тестування розробимо план, за яким буде протестовано основний функціонал програмного забезпечення.

а) Тестування входу за допомогою власної геолокації.

- 1) Тестування при наданні дозволу використання геопозиції одноразово.
- 2) Тестування при наданні дозволу використання геопозиції багаторазово.
- 3) Тестування при не наданні дозволу використання геопозиції.

б) Тестування пошуку страв у базі даних.

- 1) Тестування пошуку страв у базі даних.

в) Тестування кнопки «Our instagram».

- 1) Тестування кнопки «Our instagram».

г) Тестування кнопки «Contact information».

- 1) Тестування кнопки «Contact information».

д) Тестування кнопки «Call manager».

- 1) Тестування кнопки «Call manager».

е) Тестування кнопки «Discounts».

- 1) Тестування кнопки «Discounts».

е) Тестування відправки замовлення в базу даних.

- 1) Тестування кнопки «Confirm the order».
- 2) Тестування кнопки «Deny Order».

є) Тестування кнопки «Cart».

- 1) Тестування кнопки «Cart».

ж) Тестування кнопки «Sort».



## 5.2 Приклади тестування

Протестуємо основний функціонал програмного забезпечення за планом, наведеним у пункті 5.1.

У таблиці 5.1 наведено приклад роботи програми при наданні дозволу використання геопозиції одноразово.

Таблиця 5.1 — Приклад роботи програми при наданні дозволу використання геопозиції одноразово.

Мета тесту	Перевірити можливість входу за допомогою геолокації користувача одноразово
Початковий стан програми	Відкрите вікно вибору місцезнаходження
Вхідні дані	Місцезнаходження (широта та довгота)
Схема проведення тесту	Вибір місцезнаходження на карті та надання дозволу на використання додатком власної геопозиції
Очікуваний результат	Зберігання координат геопозиції користувача
Стан програми після проведення випробувань	Програма почне свою роботу

У таблиці 5.2 наведено приклад роботи програми при наданні дозволу використання геопозиції багаторазово.

Таблиця 5.2 — Приклад роботи програми при наданні дозволу використання геопозиції багаторазово.

Мета тесту	Перевірити можливість входу за допомогою геолокації користувача багаторазово
Початковий стан програми	Відкрите вікно вибору місцезнаходження
Вхідні дані	Місцезнаходження (широта та довгота)
Схема проведення тесту	Вибір місцезнаходження на карті та надання дозволу на використання додатком власної геопозиції назавжди
Очікуваний результат	Зберігання координат геопозиції користувача
Стан програми після проведення випробувань	Програма почне свою роботу

У таблиці 5.3 наведено приклад роботи програми при не наданні дозволу використання геопозиції.

Таблиця 5.3 — Приклад роботи програми при не наданні дозволу використання геопозиції.

Мета тесту	Перевірити можливість входу, коли користувач відмовився надавати геопозицію
Початковий стан програми	Відкрите вікно вибору місцезнаходження
Вхідні дані	Немає
Схема проведення тесту	Відмова від надання дозволу використання місцезнаходження
Очікуваний результат	Вивід попередження про необхідність надати доступ до геолокації конкретного додатку в налаштування приватності телефону
Стан програми після проведення випробувань	Програма не почне свою роботу

У таблиці 5.4 наведено приклад роботи програми при виконанні процесу пошуку страв у базі даних.

Таблиця 5.4 — Приклад роботи програми при виконанні процесу пошуку страв у базі даних.

Мета тесту	Перевірити фільтрацію у базі даних за вводом користувача
Початковий стан програми	Відкрите вікно пошукового поля
Вхідні дані	Ввід користувача
Схема проведення тесту	Введення назв страв
Очікуваний результат	Фільтрація меню під вимоги користувача
Стан програми після проведення випробувань	Програма знайде і відфільтрує меню за заданим запитом

У таблиці 5.5 наведено приклад роботи програми при тестуванні кнопки «Our instagram».

Таблиця 5.5 — Приклад роботи програми при тестуванні кнопки «Our instagram».

Мета тесту	Перевірити переадресацію на інстаграм-сторінку
Початковий стан програми	Відкрите бокове вікно сторінки інстаграму
Вхідні дані	Немає
Схема проведення тесту	Натискання на кнопку «Our instagram»
Очікуваний результат	Переадресація на інстаграм-сторінку
Стан програми після проведення випробувань	Програма вийде з додатку та перейде у середовище «Instagram»

У таблиці 5.6 наведено приклад роботи програми при тестуванні кнопки «Contact information».

Таблиця 5.6 — Приклад роботи програми при тестуванні кнопки «Contact information».

Мета тесту	Перехід у меню контактної інформації
Початковий стан програми	Відкрите бокове контактне меню
Вхідні дані	Немає
Схема проведення тесту	Натискання на кнопку «Contact information»
Очікуваний результат	Вивід – мінікарти та кнопки «Call manager»
Стан програми після проведення випробувань	Програма виведе мінікарту та кнопку «Call manager»

У таблиці 5.7 наведено приклад роботи програми при тестуванні кнопки «Call manager».

Таблиця 5.7 — Приклад роботи програми при тестуванні кнопки «Call manager».

Мета тесту	Дзвінок менеджеру
Початковий стан програми	Відкрите бокове меню та натистута кнопка «Contact information»
Вхідні дані	Немає
Схема проведення тесту	Натискання на кнопку «Call manager»
Очікуваний результат	Дзвінок менеджеру
Стан програми після проведення випробувань	Програма набере номер менеджера

У таблиці 5.8 наведено приклад роботи програми при тестуванні кнопки «Discounts».

Таблиця 5.8 — Приклад роботи програми при тестуванні кнопки «Discounts».

Мета тесту	Перегляд промо-ролика та інформаційних постерів
Початковий стан програми	Відкрите бокове меню та натистута кнопка «Discounts»
Вхідні дані	Немає
Схема проведення тесту	Натискання на кнопку «Discounts»
Очікуваний результат	Перегляд промо-ролика та інформаційних постерів
Стан програми після проведення випробувань	Програма виведе промо-ролик, інформаційні постери та додаткові дані

У таблиці 5.9 наведено приклад роботи програми при тестуванні кнопки «Confirm the order».

Таблиця 5.9 — Приклад роботи програми при тестуванні кнопки «Confirm the order».

Мета тесту	Відправлення замовлення користувача та його додаткової інформації до бази даних
Початковий стан програми	Відкрите меню корзини та натистута кнопка «Confirm the order»
Вхідні дані	Готове замовлення користувача
Схема проведення тесту	Натискання на кнопку «Confirm the order»
Очікуваний результат	Відправлення інформації по замовленню до бази даних
Стан програми після проведення випробувань	Програма виведе повідомлення про успішне підтвердження замовлення та відправить інформацію по замовленню користувача до бази даних

У таблиці 5.10 наведено приклад роботи програми при тестуванні кнопки «Deny Order».

Таблиця 5.10 — Приклад роботи програми при тестуванні кнопки «Deny Order».

Мета тесту	Скасування замовлення користувача
Початковий стан програми	Відкрите меню корзини та натистута кнопка «Deny Order»
Вхідні дані	Готове замовлення користувача
Схема проведення тесту	Натискання на кнопку «Deny Order»
Очікуваний результат	Скасування інформації по замовленню в базі даних
Стан програми після проведення випробувань	Програма скасує замовлення та видалить його з бази даних

У таблиці 5.11 наведено приклад роботи програми при тестуванні кнопки «Cart».

Таблиця 5.11 — Приклад роботи програми при тестуванні кнопки «Cart».

Мета тесту	Ознайомлення з власним вибором меню користувача: обрання к-сть страв, підрахунок остаточної ціни
Початковий стан програми	Відкрите меню корзини та натистута кнопка «Cart»
Вхідні дані	Обрані страви користувача
Схема проведення тесту	Натискання на кнопку «Cart»
Очікуваний результат	Відображення обраних страв із зазначеною кількістю та порахованою сумою замовлення
Стан програми після проведення випробувань	Програма відобразить замовлення користувача з порахованою сумою



У таблиці 5.11 наведено приклад роботи програми при тестуванні кнопки «Sort».

Таблиця 5.11 — Приклад роботи програми при тестуванні кнопки «Sort».

Мета тесту	Сортування страв за алфавітом у зростаючому та спадаючому порядку
Початковий стан програми	Відкрите головне меню та натистута кнопка «Sort»
Вхідні дані	Обрані страви користувача
Схема проведення тесту	Натискання на кнопку «Sort»
Очікуваний результат	Відображення обраних страв за алфавітним порядком залежно від вибору користувача
Стан програми після проведення випробувань	Програма відобразить відсортовані списки страв

## 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

### 6.1. Робота з програмою

Після запуску мобільного застосунку, відкривається вікно дозволу користувача на використання геолокації додатком «Green Food» (рисунок 6.1).

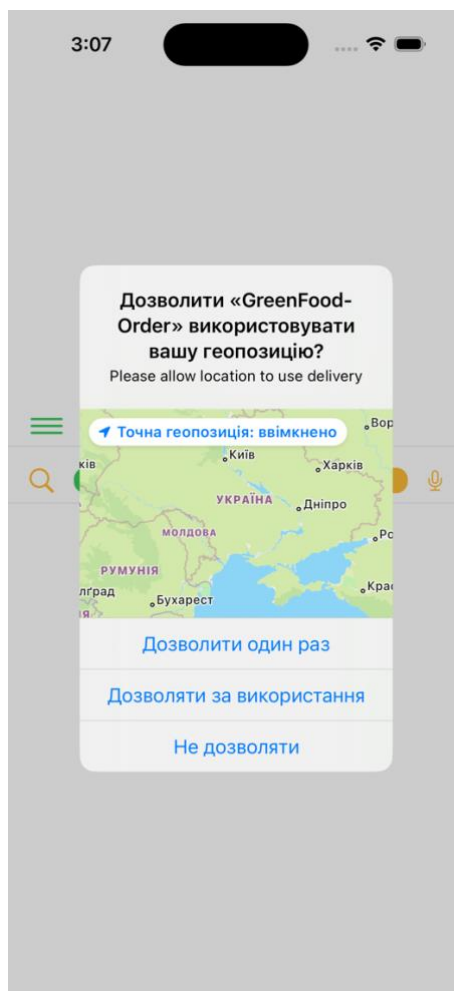


Рисунок 6.1 — Вікно дозволу користувача на надання геопозиції

За дозволом користувача програма продовжує свою роботу та відкриває головну сторінку програми, інакше доступ до програми блокується і може бути поновлений лише в налаштуваннях самого додатку всередині телефону. Отож, якщо дозвіл був наданий, відкриється головна сторінка мобільного застосунку (рисунок 6.2).

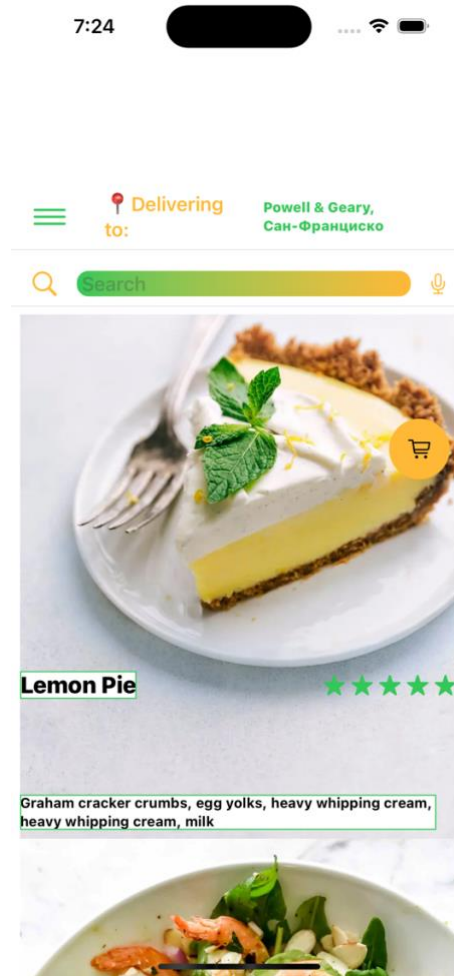


Рисунок 6.2 — Вікно головної сторінки мобільного застосунку

Вікно головної сторінки застосунку містить у собі меню закладу «Green Food» та пошукове поле через яке здійснюється фільтрація страв за запитом користувача. Якщо користувач натисне на нього, то програма перейде в режим пошуку (рисунок 6.3).

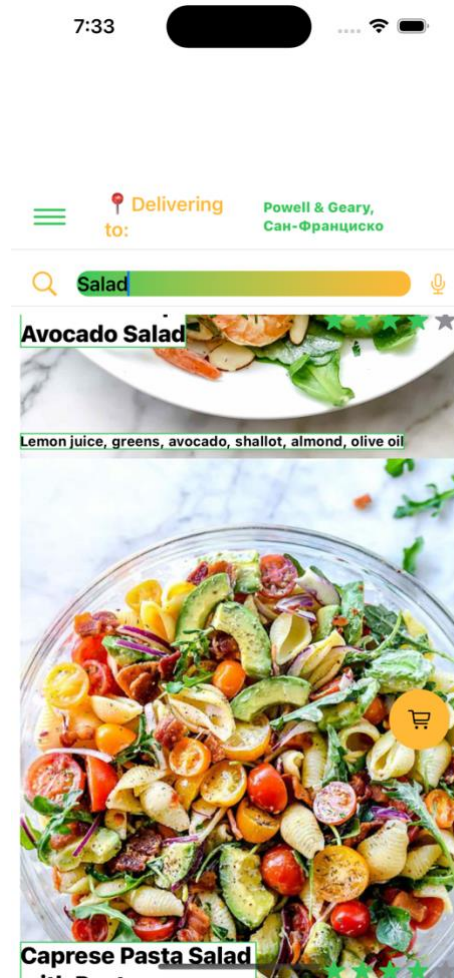


Рисунок 6.3 — Режим пошуку страв у меню

На цьому етапі користувач має право обрати страви за своїм вподобанням обравши бажанні ним позиції. Реалізувати це можна натисканням на кнопку справа від фото страви (рисунок 6.4).

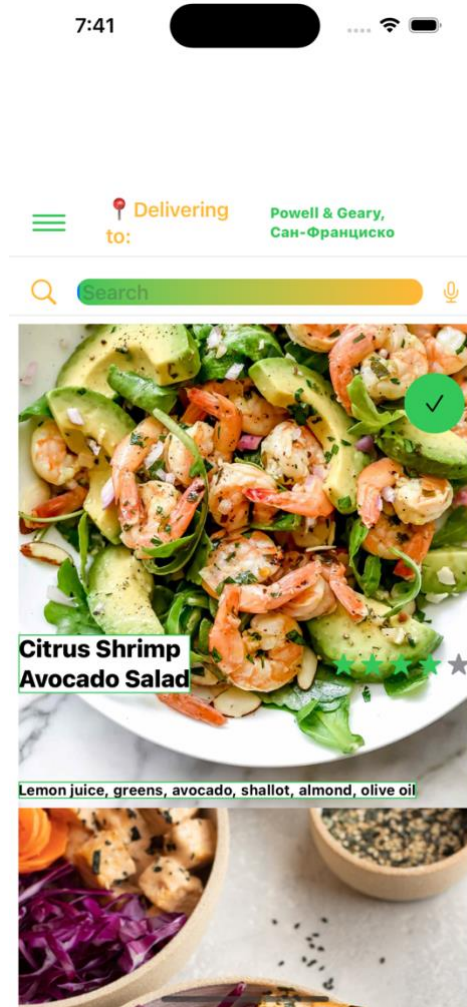


Рисунок 6.4 — Демонстрування додання страви у корзину

Після того як користувач обрав страви, він може перейти до бокового меню натиснувши на іконку навігації. Після натиснення кнопки користувач може побачити бокове меню (рисунок 6.5).

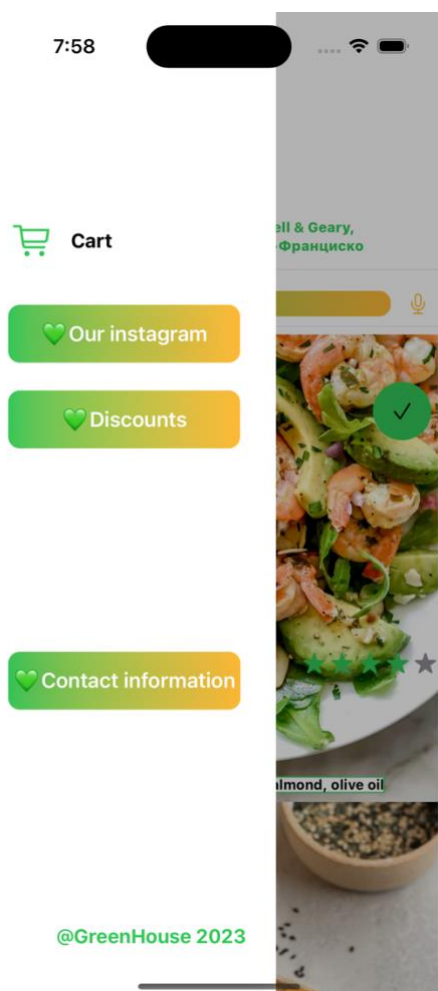


Рисунок 6.5 — Вікно бокового меню мобільного застосунку

Щоб перевірити своє замовлення, обрати точну кількість страв, розрахувати остаточну ціну своє замовлення потрібно натиснути кнопку «Cart» та перейти на сторінку корзини користувача (рисунок 6.6).

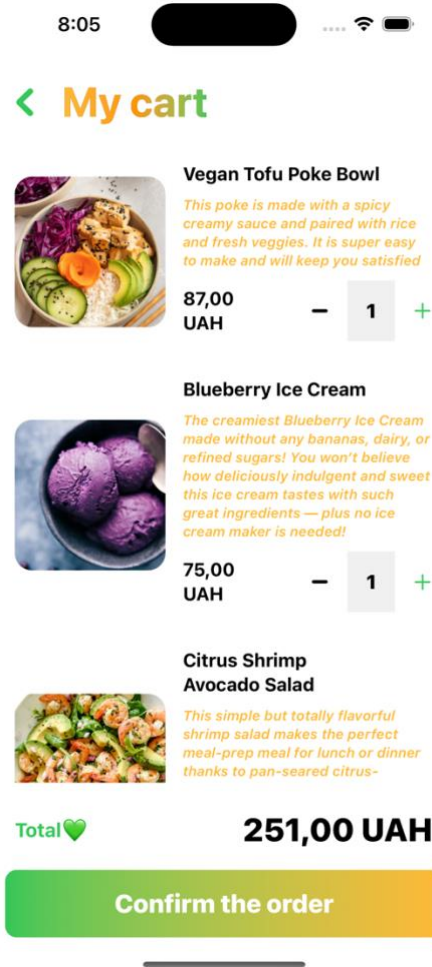


Рисунок 6.6 — Сторінка корзини мобільного застосунку

Щоб видалити певну позицію, користувач повинен натиснути на конкретну страву, що призведе до появи кнопки видалення (рисунок 6.7).

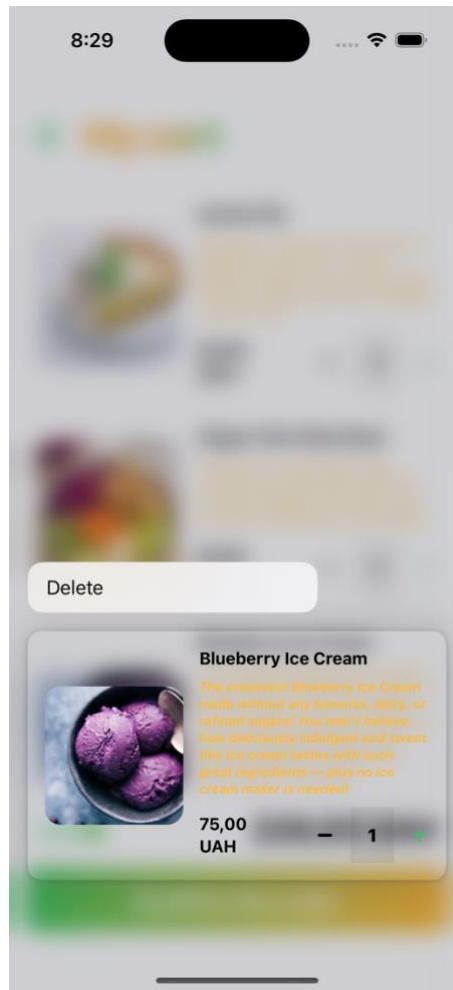


Рисунок 6.7 — Вікно видалення страв

Щоб підтвердити своє замовлення потрібно скористатися кнопкою «Confirm the order», після цього замовлення буде передане до бази даних та користувачу прийде повідомлення про успішність підтвердження замовлення (рисунок 6.8).



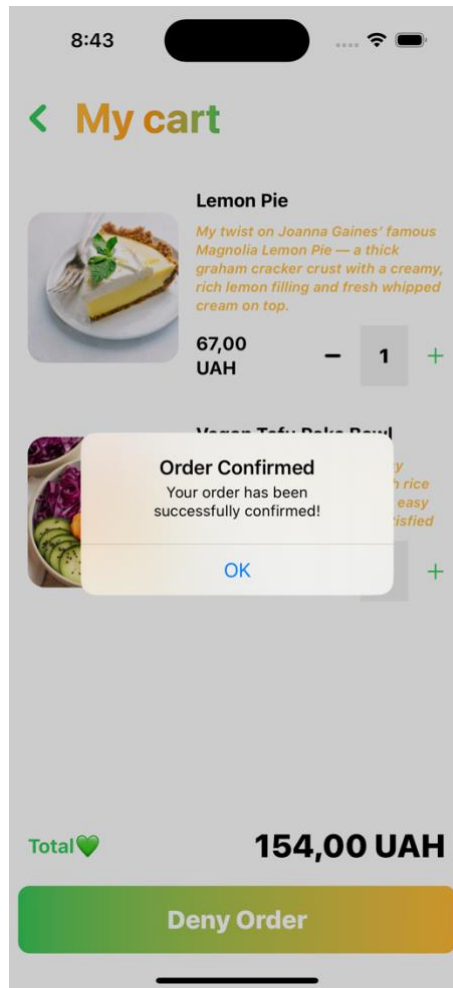


Рисунок 6.8 — Вікно успішного підтвердженого замовлення

Демонстрація бази даних, яка містить отриману інформацію про замовлення користувача (рисунок 6.9).

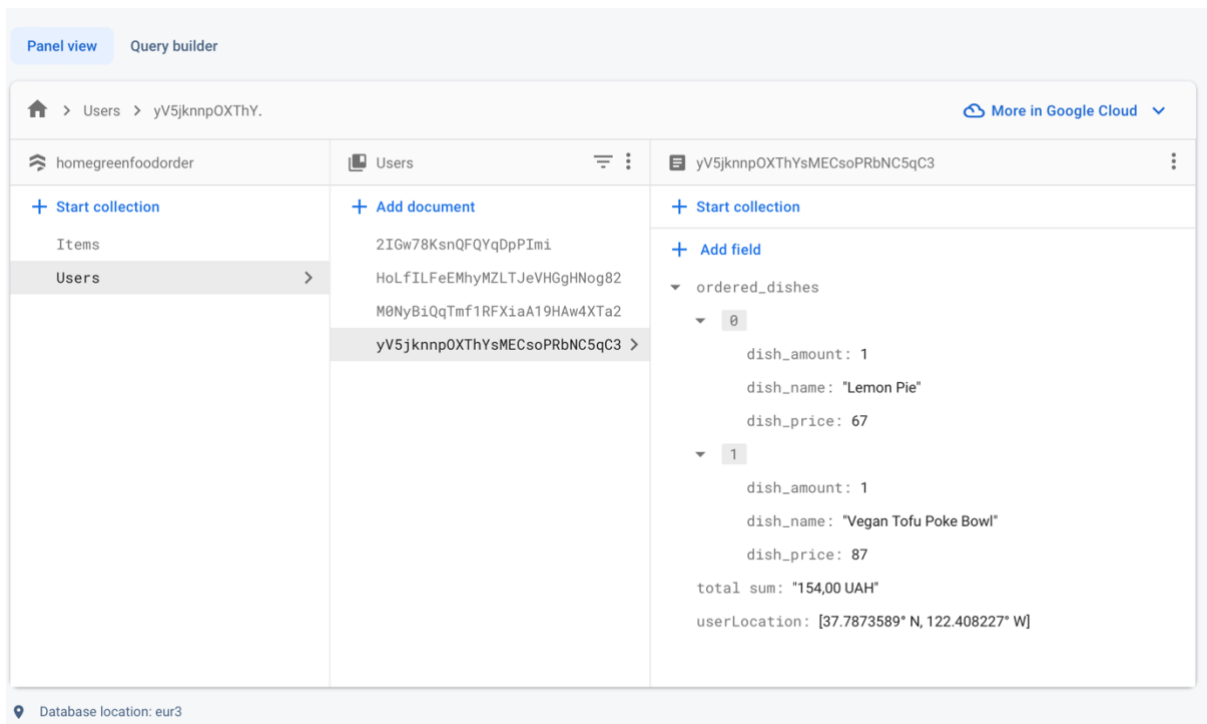


Рисунок 6.9 — Вікно успішного відправлення замовлення у базу даних

В боковому меню програми можна ознайомитися з контактною інформацією мобільного застосунку (мінікартою геолокації закладу та скористатися дзвінком менеджера в разі виникнення питань). Вікно з контактною інформацією можна побачити при натисненні кнопки «Contact information» (рисунок 6.10).

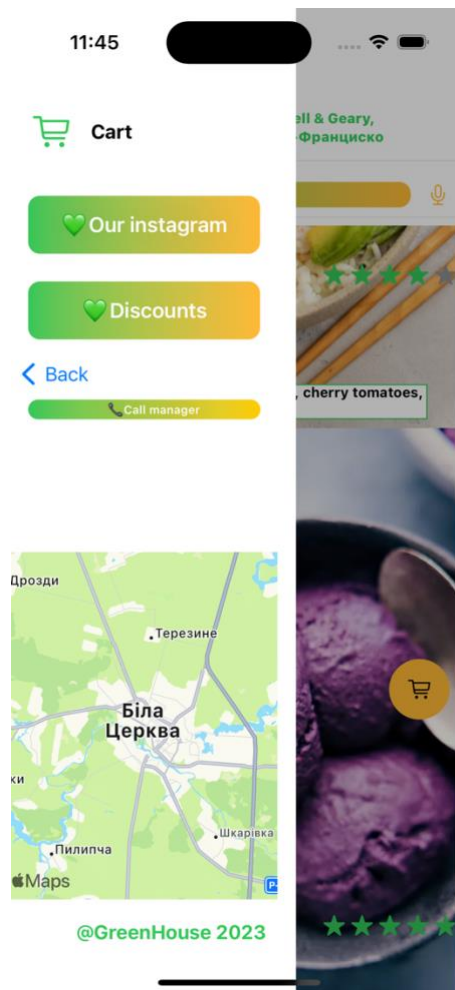


Рисунок 6.10 — Вікно з контактною інформацією

В боковому меню програми можна ознайомитися з інформацією знижок та акцій (промо-роликом та постерами). Вікно зі знижками можна побачити при натисненні кнопки «Discounts» (рисунок 6.11).

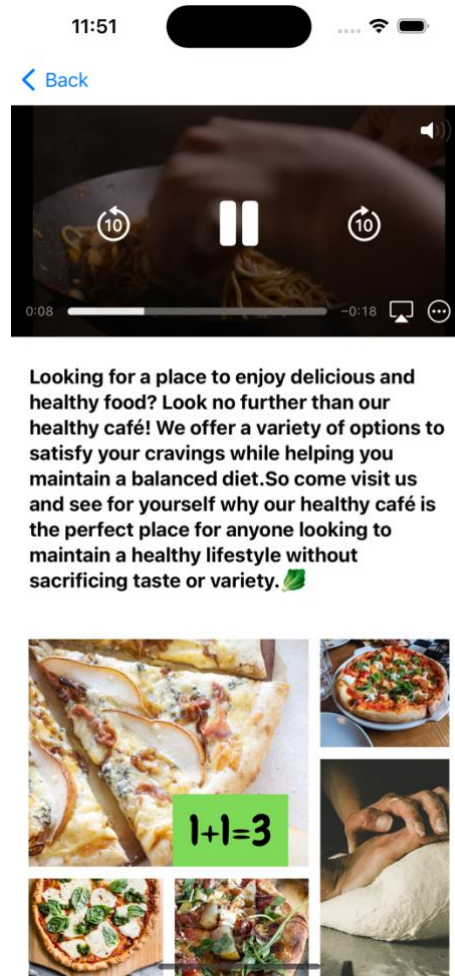


Рисунок 6.11 — Вікно з інформацією знижок та акцій

В боковому меню програми також присутня кнопка «My instagram», яка переадресовує користувача на сторінку закладу (рисунок 6.12).



Рисунок 6.12 — Вікно переадресації на інстаграм-сторінку

Додатковою опцією в меню програми наявна кнопка «Sort», яка відсортовує страви за алфавітним або ж протилежним порядком (рисунок 6.13).

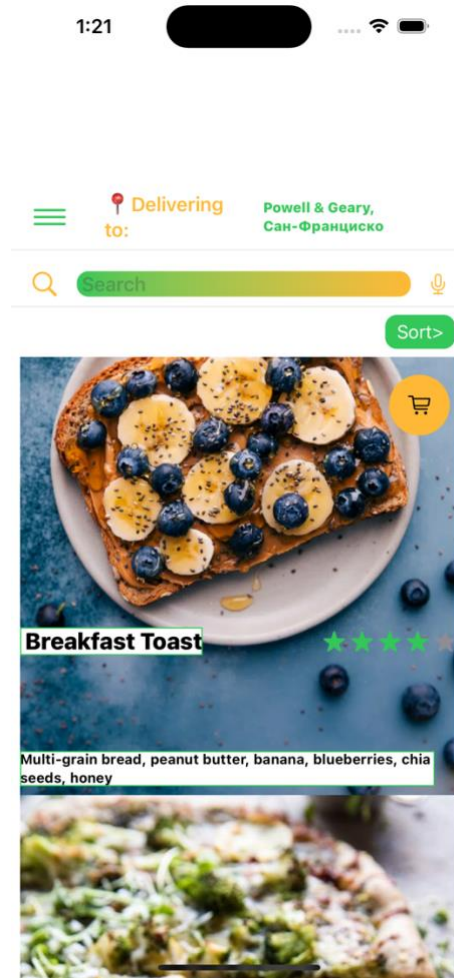


Рисунок 6.13 — Функція сортування за алфавітом

## 6.2. Формат вхідних та вихідних даних

Вхідними даними для програми є геолокація користувача та меню закладу, яке знаходиться у базі даних.

Результатом виконання програми є успішно відправлене замовлення у базу даних.

### 6.3. Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 — Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	iOS12/iOS13/iOS14/iOS15/iOS16	iOS16
Чіпсети	Apple A12 Bionic/Apple A13 Bionic/Apple A14 Bionic	Apple A14 Bionic
Оперативна пам'ять	300 MB RAM	350 MB RAM
Дисплей	1792 x 828px	2532 x 1170px
Додаткове програмне забезпечення	Firebase 9.6.0, SDWebImageSwiftUI 2.2.2, SDWebImage 5.15.0	

## ВИСНОВКИ

У розділі «Постановка задачі» визначено вхідні та вихідні дані мобільного застосунку, принципи роботи додатка «Green Food» та визначено вимоги до користувача для маніпулювання програмою.

У розділі «Теоретичні відомості» підкреслена актуальність даного застосунку та алгоритм користування програмою.

У розділі «Опис алгоритмів» визначено алгоритми додатку «Green Food», які виконують важливі функції: сортування, пошук тощо.

У розділі «Опис програмного забезпечення» розроблено діаграму класів UML, яка продемонструвала логіку програми, визначено найважливіші методи та їх призначення.

У розділі «Тестування програмного забезпечення» наведено план тестування та протестовано основний функціонал програмного забезпечення за цим планом.

У розділі «Інструкція користувача» наведено головний підхід до використання програми, визначено можливості та системні вимоги для користувача. Загалом було продемонстровано всі етапи роботи програми.



## ПЕРЕЛІК ПОСИЛАНЬ

- Документація до мови програмування Swift [hackingwithswift.com](https://www.hackingwithswift.com). 2023. *SwiftUI by Example*. URL: <https://www.hackingwithswift.com/quick-start/swiftui> (дата звернення: 27.02.2023).
- Документація до мови програмування Swift [developer.apple.com](https://developer.apple.com). 2023. *Swift / Apple Developer Documentation*. URL: <https://developer.apple.com/documentation/swift> (дата звернення: 03.03.2023).
- Документація до фреймворку SwiftUI [getstream.io](https://getstream.io). 2023. Learn SwiftUI: 24 Essential Tutorials for Beginners. URL: <https://getstream.io/blog/learn-swiftui/> (дата звернення: 17.03.2023).
- Документація до роботи з базою даних Firebase [firebase.google.com](https://firebase.google.com). 2023. *Firebase Realtime Database*. URL: <https://firebase.google.com/docs/database> (дата звернення: 07.04.2023).

**ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**  
**КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського**

Кафедра  
інформатики та програмної інженерії

Затвердив  
Керівник Головченко Максим Миколайович  
«03» березня 2023 р.

Виконавець:  
Студент Андрєєва Уляна Андріївна  
«03» березня 2023 р.

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на виконання курсової роботи  
на тему: Створення програми доставки їжі «GreenFood»  
з дисципліни:  
«Основи програмування»

Київ 2023

1. *Мета:* Метою курсової роботи є підвищення зручності процесу замовлення їжі через мобільний застосунок «GreenFood».
2. *Дата початку роботи:* «03» березня 2023 р.
3. *Дата закінчення роботи:* «31» травня 2023 р.
4. *Вимоги до програмного забезпечення.*

1) Функціональні вимоги:

- Можливість автоматично визначити геолокацію користувача (не на симуляторі).
- Можливість погодитися на поширення геолокації один раз, завжди або ж повністю заборонити поширення.
- Можливість обрання страв з бази даних Firebase, яка знаходиться на хмарі.
- Можливість пошуку страв, які містяться в меню.
- Можливість додавання в корзину страв, які користувач бажає замовити.
- Можливість відобразити інформацію про замовлені страви в корзині.
- Можливість маніпуляцій у корзині: обирання точної кількості страв, видалення страв та автоматичне рахування суми за обраними позиціями.
- Можливість підтвердження замовлення або ж навпаки скасування.
- Можливість перегляду координат покупця та його замовлення в базі даних Firebase, яка знаходиться на хмарі.
- Можливість переходу до Інстаграм сторінки завдяки натисканню на кнопку «Our instagram».

- Можливість перегляду промо – ролика доставки їжі та декількох банерів зі знижками завдяки натисканню на кнопку «Discounts».

- Можливість перегляду контактної інформації (геолокацію служби доставки та їх контактний номер телефону, на який можна подзвонити за кліком(не на симуляторі).

## 2) Нефункціональні вимоги:

- Можливість запуску програмного забезпечення на комп'ютерах із операційною системою macOS Ventura 13, macOS Ventura 14, macOS Ventura 15, macOS Ventura 16.

- Для роботи програмного забезпечення на операційних системах macOS Ventura 13, macOS Ventura 14, macOS Ventura 15, macOS.

Ventura 16 в папці повинні знаходитись функціональні файли (GoogleService-Info.plist для під'єднання Firebase та папки з ресурсами для програми, які будуть встановлені за допомогою інсталятора).

- Можливість запуску програмного забезпечення на пристроях із операційною системою IOS із версією SDK 6.1 і більше (Це версії IOS 12.5.7, IOS 13.7, IOS 14.8.1, IOS 15.7.3, IOS 16.3.1, IOS 16.4 beta).

- Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

*5. Стадії та етапи розробки:*

- 1) Об'єктно-орієнтований аналіз предметної області задачі (до 09.03.2023 р.).
- 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 27.03.2023 р.).
- 3) Розробка програмного забезпечення (до 17.04.2023 р.).
- 4) Тестування розробленої програми (до 30.04.2023 р.).
- 5) Розробка пояснювальної записки (до 29.05.2023 р.).
- 6) Захист курсової роботи (до 06.06.2023 р.).

*6. Порядок контролю та приймання.* Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

## ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

*Тексти програмного коду мобільного застосунку «Green  
Food»*

---

(Найменування програми (документа))

*Електронний носій*

---

(Вид носія даних)

*30 арк, 200 Мб*

---

(Обсяг програми (документа), арк.,

*студентки групи ІП-22 І курсу*

*Андрєвої У. А.*

**GreenFoodOrderApp.swift**

```

import SwiftUI
import FirebaseCore
import Firebase

class AppDelegate: NSObject, UIApplicationDelegate {
    func application(_ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        FirebaseApp.configure()
        return true
    }
}

@main
struct GreenFoodOrderApp: App {
    @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate
    var body: some Scene {
        WindowGroup {
            NavigationView {
                ContentView()
            }
        }
    }
}

```

**ContentView.swift**

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        NavigationView{
            Home()
                .navigationBarHidden(true)
                .navigationBarBackButtonHidden(true)
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```



**Item.swift**

```
import SwiftUI

struct Item : Identifiable{
    var id: String
    var dish_name: String
    var dish_photo: String
    var dish_description: String
    var dish_ingredients: String
    var dish_price: NSNumber
    var dish_rank: String
    var isInCart: Bool = false}
```

**Cart.swift**

```
import SwiftUI

struct Cart: Identifiable {
    var id = UUID().uuidString
    var item: Item
    var amount: Int
}
```

**MainMenu.swift**

```

import SwiftUI
import MessageUI
import MapKit
import MediaPlayer
import CoreVideo
import AVKit

struct DiscountsDetails: View{
    @State var player = AVPlayer(url: URL(string:
"https://joy1.videvo.net/videvo_files/video/free/2016-
12/large_watermarked/FoodPack1_14_Videvo_preview.mp4")!) // 1

    var body: some View {

        VideoPlayer(player: player)
            .frame(width: 400,
                height: 200,
                alignment: .center)

        ScrollView(.vertical,showsIndicators: false){
            LazyVStack(spacing: 0){
                Text("Looking for a place to enjoy delicious and healthy food? Look
no further than our healthy café! We offer a variety of options to satisfy your cravings
while helping you maintain a balanced diet.So come visit us and see for yourself why
our healthy café is the perfect place for anyone looking to maintain a healthy lifestyle
without sacrificing taste or variety. 🥦")

                .padding()
                .fontWeight(.bold)
                .frame(maxWidth: .infinity)
                .foregroundColor(Color.black)
            }
        }
    }
}

```

```

        .multilineTextAlignment(.leading)}
        ZStack {

            VStack(alignment: .center) {

                Image("Image 3").resizable()
                    .frame(width: UIScreen.main.bounds.width -
30,height: 300)

                    .padding()
                Image("Image 4").resizable()
                    .frame(width: UIScreen.main.bounds.width -
30,height: 300)

                    .padding()
                Spacer(minLength: 0)
            }
        }
    }
}

struct DetailView: View {
    var phoneNumber = "380677393876" //49.806698703333666,
30.10266722386009

    @State private var region = MKCoordinateRegion(center:
CLLocationCoordinate2D(latitude: 49.806698703333666 , longitude:
30.10266722386009), span: MKCoordinateSpan(latitudeDelta: 0.2, longitudeDelta:
0.2))

    var body: some View {
        VStack {
            Button(action: {

```

```

        let phone = "tel://"
        let phoneNumberformatted = phone + phoneNumber
        guard let url = URL(string: phoneNumberformatted) else { return }
        UIApplication.shared.open(url)
    }) {
        Text("📞 Phone number:" + phoneNumber)
    }
}

.padding(.leading)
.shadow(radius: 10)
.font(.system(size:10))
.fontWeight(.semibold)
.frame(width: 200,height:17)
.foregroundColor(.white)
.background(LinearGradient(gradient: Gradient(colors: [Color.green,
Color.yellow])), startPoint: .leading, endPoint: .trailing))
.cornerRadius(10)
Spacer(minLength: 0)
Map(coordinateRegion: $region)
    .frame(width: 400, height: 300)
}
}

struct MainMenu: View {
    @ObservedObject var homeData : HomeViewModel
    @State var selectedTag: String?
    @State private var isShowingDetailView = false

    var body: some View {
        VStack{

```

```

        NavigationLink(destination: CartView(homeData: homeData)){
            HStack(spacing: 15){
                Image(systemName: "cart")
                    .font(.title)
                    .foregroundColor(.green)

                Text("Cart")
                    .fontWeight(.bold )
                    .foregroundColor(.black)
                Spacer(minLength: 0)
            }
            .padding()
        }
        HStack(spacing: 15){
            Button(action: {
                if let url = URL(string:
"https://www.instagram.com/greenhousefoodss/") {
                    UIApplication.shared.open(url)

                }

            }) {
                Text("❤️ Our instagram")
                    .shadow(radius: 21)
                    .font(.system(size:18))
                    .fontWeight(.semibold)
                    .frame(width: 200,height: 50)
                    .foregroundColor(.white)
                    .background(LinearGradient(gradient: Gradient(colors:
[Color.green, Color.init("Color")]), startPoint: .leading, endPoint: .trailing))

```

```

        .cornerRadius(10)
        Spacer(minLength: 0)

    }
    .padding()
}

HStack(spacing: 15){
    Button(action: {
        self.selectedTag = "xx"
    }, label: {
        Text("❤️ Discounts")

        .font(.system(size:18))
        .fontWeight(.semibold)
        .frame(width: 200,height: 50)
        .foregroundColor(.white)
        .background(LinearGradient(gradient: Gradient(colors:
[Color.green, Color.init("Color")]), startPoint: .leading, endPoint: .trailing))
        .cornerRadius(10)
    })

    .background(
        NavigationLink(
            destination:
            DiscountsDetails(),
            tag: "xx",
            selection: $selectedTag,
            label: { EmptyView() }
        )
    )
}

```

```
)}
```

```
HStack(spacing: 15){
  NavigationView {
    NavigationLink(destination: DetailView()) {
      Text("💚 Contact information")
        .shadow(radius: 21)
        .font(.system(size:18))
        .fontWeight(.semibold)
        .frame(width: 200,height: 50)
        .foregroundColor(.white)
        .background(LinearGradient(gradient: Gradient(colors:
[Color.green, Color.init("Color")]), startPoint: .leading, endPoint: .trailing))
        .cornerRadius(10)
    }
  }.padding()
}
```

```
Spacer()
HStack{
  Spacer()
  Text("@GreenHouse 2023")
    .fontWeight(.bold)
    .foregroundColor(.green )
}
.padding(10)
}
```

```
.padding([.top,.trailing])
```



```
.frame(width: UIScreen.main.bounds.width / 1.6)
.background(Color.white.ignoresSafeArea())
}
}
```

**Home.swift**

```

import SwiftUI

struct Home: View {
    @StateObject var HomeModel = HomeViewModel()

    var body: some View {

        GeometryReader{ geometry in
            VStack(spacing: 7){

                Spacer(minLength: 0)
                HStack(spacing: 30){
                    Button(action: {
                        withAnimation(.easeIn){HomeModel.MainMenu.toggle()}
                    }, label: {

                        Image(systemName: "line.horizontal.3")
                            .font(.title)
                            .foregroundColor(Color.green)
                    })
                    Text(HomeModel.userLocation == nil ? "Locating... " : "📍 Delivering
to:")

                    .foregroundColor(Color.init("Color"))
                    .bold()
                    Text(HomeModel.userAddress)
                        .font(.caption)
                        .fontWeight(.heavy)
                        .foregroundColor(.green)
                }
            }
        }
    }
}

```

```

        Spacer(minLength: 0 )
    }

    .padding([.horizontal,.top])
    Divider()
    HStack(spacing: 15){
        Image(systemName: "magnifyingglass")
            .font(.title2)
            .foregroundColor(Color.init("Color"))
        TextField( " Search", text: $HomeModel.search)
            .background(LinearGradient(gradient: Gradient(colors: [Color.green,
Color.init("Color")]), startPoint: .leading, endPoint: .trailing))
            .foregroundColor(Color.black)
            .fontWeight(.semibold)
            .cornerRadius(10)

        Image(systemName: "mic")
            .renderingMode(.template)
            .foregroundColor(Color.init("Color"))
    }

    .padding(.horizontal)
    .padding(.top,10)
    Divider()
    if HomeModel.items.isEmpty{
        Spacer()
        ProgressView()
        Spacer()
    }

```

```

    }
    else{
        ScrollView(.vertical, showsIndicators: false,content: {

            VStack(spacing:80) {
                ForEach(HomeModel.filteredData){item in

                    ZStack(alignment: Alignment(horizontal:
.listRowSeparatorTrailing, vertical: .top), content: {
                        ItemView(item:item)
                        HStack{
                            Button(action: {
                                HomeModel.addToCart(item: item)
                            }, label: {
                                Image(systemName: item.isInCart ? "checkmark" : "cart")
                                    .foregroundColor(.black)
                                    .padding(17.0)
                                    .background(item.isInCart ? Color.green :
Color.init("Color"))
                                    .clipShape(Circle())

                            })
                        }
                        .padding( .trailing ,5.0)
                        .padding( .top,50 )
                    })
                .frame(width: UIScreen.main.bounds.width - 15,
                    height: UIScreen.main.bounds.height - 450)
                Spacer(minLength: 0)
            }
        }
    }

```



```

    }
}

.onAppear(perform: {
    HomeModel.locationManager.delegate = HomeModel
})
.onChange(of: HomeModel.search, perform: { value in
    DispatchQueue.main.asyncAfter(deadline: .now() + 0.3){
        if value == HomeModel.search && HomeModel.search != "" {
            HomeModel.filteringData()
        }
    }
    if HomeModel.search == ""{
        withAnimation(.linear ){
            HomeModel.filteredData = HomeModel.items
        }
    }
})
}
}

```

**HomeViewModel.swift**

```

import SwiftUI
import CoreLocation
import Firebase

class HomeViewModel:NSObject ,ObservableObject, CLLocationManagerDelegate{

    @Published var locationManager = CLLocationManager()
    @Published var search = ""
    @Published var userLocation: CLLocation!
    @Published var userAddress = ""
    @Published var noLocation = false
    @Published var MainMenu = false
    @Published var items: [Item] = []
    @Published var filteredData: [Item] = []
    @Published var cartItems: [Cart] = []
    @Published var orderedDishes = false

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
        switch manager.authorizationStatus{
        case .authorizedWhenInUse:
            print("You have been authorized")
            manager.requestLocation()
            self.noLocation = false

        case.denied:
            print("You have been denied")
            self.noLocation = true
        default:
            print("Not defined")
            self.noLocation = false
        }
    }
}

```

```

        //Direct call
        locationManager.requestWhenInUseAuthorization()
    }
}

func locationManager(_ manager: CLLocationManager, didFailWithError error:
Error) {
    print(error.localizedDescription)
}

func locationManager(_ manager: CLLocationManager, didUpdateLocations
locations: [CLLocation]){
    self.userLocation = locations.last //reading user location
    self.extractLoc()
    self.login()
}

func extractLoc(){
    CLGeocoder().reverseGeocodeLocation(self.userLocation) { (res,err) in
        guard let safeData = res else{return}
        var adress = ""

        adress += safeData.first?.name ?? ""
        adress += ", "
        adress += safeData.first?.locality ?? ""
        self.userAddress = adress
    }
}

func login(){ //anonymus login
    Auth.auth().signInAnonymously{(res,err) in
        if err != nil{
            print(err!.localizedDescription )
        }
    }
}

```



```

        return
    }
    print("Success = \$(res! .user.uid)")

    self.fetchingData()
}

func fetchingData(){ //adding data from firebase
    let db = Firestore.firestore()

    db.collection("Items").getDocuments { (snap,err) in

        guard let itemData = snap else{return}

        self.items = itemData.documents.compactMap({ (doc) -> Item? in
            let id = doc.documentID
            let name = doc.get("dish_name") as! String
            let photo = doc.get("dish_photo") as! String
            let description = doc.get("dish_description") as! String
            let ingredients = doc.get("dish_ingredients") as! String
            let price = doc.get("dish_price") as! NSNumber
            let rank = doc.get("dish_rank") as! String
            return Item(id: id, dish_name: name, dish_photo: photo, dish_description:
description, dish_ingredients: ingredients, dish_price: price, dish_rank: rank)
        })
        self.filteredData = self.items
    }
}

func filteringData(){
    withAnimation(.linear){

```

```

        self.filteredData = self.items.filter{
            return $0.dish_name.lowercased().contains(self.search.lowercased())
//searching some information
        }
    }
}

func addToCart(item: Item){
    self.items[getIndex(item: item, isCartIndex: false)].isInCart = !item.isInCart //
check for adding
    let filterindex = self.filteredData.firstIndex{ (item1) -> Bool in

        return item.id == item1.id
    } ?? 0
    self.filteredData[filterindex].isInCart = !item.isInCart // updating filtered Data,
also for search
    if item.isInCart {
        self.cartItems.remove(at: getIndex(item: item, isCartIndex: true)) //remove
        return
    }
    self.cartItems.append(Cart(item: item, amount: 1))// adding dishes
}

func getIndex(item: Item, isCartIndex: Bool) -> Int{
    let index = self.items.firstIndex{ (item1) -> Bool in
        return item.id == item1.id
    } ?? 0
    let cartIndex = self.cartItems.firstIndex{ (item1) -> Bool in
        return item.id == item1.item.id
    } ?? 0
    return isCartIndex ? cartIndex : index
}

```

```

}
func calculatePrice()->String{
    var price : Float = 0
    cartItems.forEach{(item) in
        price += Float(item.amount) * Float(truncating: item.item.dish_price)
    }
    return getPrice(value: price)
}
func getPrice(value: Float)->String{
    let format = NumberFormatter()
    format.numberStyle = .currency
    return format.string(from: NSNumber(value: value)) ?? ""
}
func updating(){
    let db = Firestore.firestore()
    if orderedDishes{
        orderedDishes = false
        db.collection("Users").document(Auth.auth().currentUser!.uid).delete{
            (err) in
            if err != nil{
                self.orderedDishes = true
            }
        }
    }
    return
}
var data : [[String: Any]] = []
cartItems.forEach{(cart) in
    data.append([
        "dish_name" : cart.item.dish_name,
        "dish_amount" : cart.amount,

```

```

        "dish_price" : cart.item.dish_price

    })
}
orderedDishes = true
db.collection("Users").document(Auth.auth().currentUser!.uid).setData([
    "ordered_dishes" : data,
    "total sum" : calculatePrice(),
    "userLocation" : GeoPoint(latitude: userLocation.coordinate.latitude,
longitude: userLocation.coordinate.longitude)
]) { (err) in
    if err != nil{
        self.orderedDishes = false
        return
    }
    print("success")
}
}
}

```

**ItemView.swift**

```

import SwiftUI
import SDWebImageSwiftUI
struct ItemView: View {
    var item: Item
    var body: some View {
        VStack{
            //Photos
            WebImage(url: URL(string: item.dish_photo))
                .resizable()
                .aspectRatio( contentMode: .fill)
                .frame(height: 268)

            Spacer(minLength: 0)

            //Dish names
            HStack(spacing: 2){
                Text(item.dish_name )

                .font(/*@START_MENU_TOKEN@*/.title3/*@END_MENU_TOKEN@*/)
                .background(.white)
                .border(.green)
                .fontWeight(.heavy)
                .foregroundColor(Color.black)
                Spacer(minLength: 0)

                //Ratings

```

```

    ForEach(1...5,id: \.self){index in
        Image(systemName: "star.fill")
            .foregroundColor(index <= Int(item.dish_rank) ?? 0 ?
                Color.green : .gray )

    }
}

//Dish ingredients
HStack{
    Spacer(minLength: 0)
    Text(item.dish_ingredients)
        .multilineTextAlignment(.leading)
        .background(.white)
        .font(.caption)
        .fontWeight(.bold)
        .foregroundColor(.primary)
        .border(.green)
        .lineLimit(4)
        .frame(
            maxWidth: .infinity,
            maxHeight: .infinity,
            alignment: .leadingLastTextBaseline)

    Spacer(minLength: 0)
}
}
}
}

```

**CartView.swift**

```

import SwiftUI
import SDWebImage
import SDWebImageSwiftUI

struct CartView: View {
    @ObservedObject var homeData: HomeViewModel
    @Environment(\.presentationMode) var present

    @State private var isOrderConfirmed = false

    var body: some View {
        GeometryReader { geometry in
            VStack {
                HStack(spacing: 20) {
                    Button(action: { present.wrappedValue.dismiss() }) {
                        Image(systemName: "chevron.left")
                            .font(.system(size: 26, weight: .heavy))
                            .foregroundColor(.green)
                    }

                    Text("My cart")
                        .font(Font.system(size: 36, weight: .bold))
                        .multilineTextAlignment(.center)
                        .overlay {
                            LinearGradient(
                                colors: [.init("Color"), .orange, .green],
                                startPoint: .leading,
                                endPoint: .trailing
                            )
                        }
                }
            }
        }
    }
}

```

```

        .mask(
            Text("My cart")
                .font(Font.system(size: 36, weight: .bold))
                .multilineTextAlignment(.center)
        )
    }

    Spacer()
}

.padding()

ScrollView(.vertical,showsIndicators: false){

    LazyVStack(spacing: 0){

        ForEach(homeData.cartItems){ cart in
            HStack(spacing: 15){
                WebImage(url: URL(string: cart.item.dish_photo))
                    .resizable()
                    .aspectRatio(contentMode: .fill)
                    .frame(width: 130,height: 130)
                    .cornerRadius(15)
                VStack(alignment: .leading, spacing: 10){
                    Text(cart.item.dish_name)
                        .font(.callout)
                        .fontWeight(.bold)
                        .foregroundColor(.black)
                    Text(cart.item.dish_description)
                        .font(.caption)
                        .fontWeight(.semibold)
                }
            }
        }
    }
}

```



```

        .foregroundColor(Color.init("Color"))
        .italic()
    HStack(spacing: 15){
        Text(homeData.getPrice(value: Float(truncating:
cart.item.dish_price) ))

        .font(.callout)
        .fontWeight(.bold)
        .foregroundColor(.black)
    Spacer(minLength: 0)
    Button(action: {
        if cart.amount > 1{
            homeData.cartItems[homeData.getIndex(item:
cart.item, isCartIndex: true)].amount -= 1
        }
    }){
        Image(systemName: "minus")
        .font(.system(size: 16,weight: .heavy))
        .foregroundColor(.black)
    }
    Text("\$(cart.amount)")
        .fontWeight(.heavy)
        .foregroundColor(.black)
        .padding(.vertical)
        .padding(.horizontal)
        .background(Color.black.opacity(0.06))
    Button(action: {
        homeData.cartItems[homeData.getIndex(item:
cart.item, isCartIndex: true)].amount += 1

    })

```

```

        {
            Image(systemName: "plus")
                .foregroundColor(.green)
        }
    }
}
}
.padding()
.contextMenu{
    Button(action:{
        let index = homeData.getIndex(item: cart.item,
isCartIndex: true)


        let itemIndex = homeData.getIndex(item: cart.item,
isCartIndex: false)

        homeData.items[itemIndex].isInCart = false
        homeData.filteredData[itemIndex].isInCart = false
        homeData.cartItems.remove(at: index)

    }){
        Text("Delete")
            .fontWeight(.semibold)
            .foregroundColor(.black)
            .background(Color.init("Color"))
    }
}
}
}
}
}

```

```

VStack {
  HStack {
    Text("Total 

```

```

        .alert(isPresented: $isOrderConfirmed) {
            Alert(
                title: Text("Order Confirmed"),
                message: Text("Your order has been successfully confirmed!"),
                dismissButton: .default(Text("OK"))
            )
        }
    }

    Spacer(minLength: 0)
}

.navigationBarHidden(true)
.navigationBarBackButtonHidden(true)

Spacer(minLength: 0)
}
}
}

```