НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»
на тему: «Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів»

Виконала:
студентка IV курсу, групи КП-52
Яшан Ольга Сергіївна _____

Керівник:
старший викладач кафедри ПЗКС, к.т.н.,
Хіцко Я.В. _____

Консультант з нормоконтролю:
доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
доцент кафедри СПСКС, к.т.н.,
Боярінова Ю.Є. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студентці

Яшан Ользі Сергіївні

1. Тема проекту «Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів», керівник проекту Хіцко Яна Володимирівна, к.т.н., старший викладача кафедри ПЗКС, затверджені наказом по університету від «22» травня 2019 р. № 1331-С

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз мов та технологій розробки веб-сайтів та мобільних додатків;
   - розробка інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів;
   - опис розроблених підпрограм та алгоритмів;
   - аналіз розробленої системи.

5. Перелік обов'язкового графічного матеріалу:
   - схема алгоритму пошуку спожитих продуктів (креслення);
   - схема алгоритму обрахунку спожитих цукоровмісних продуктів(креслення);
   - прототип сторінки (плакат);
   - діаграма переходів станів (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розробка та узгодження технічного завдання	28.11.2018	
3.	Розробка структури веб-ресурсу та мобільного додатку	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розробка дизайну сторінок та графічних елементів веб-сайту та екранів мобільного додатку	14.01.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	31.01.2019	
7.	Програмна реалізація веб-ресурсу та мобільного додатку	15.03.2019	
8.	Тестування веб-ресурсу та мобільного додатку	31.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	15.04.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	30.04.2019	
11.	Підготовка графічної частини дипломного проекту	14.05.2019	
12.	Оформлення документації дипломного проекту	31.05.2019	

Студент О. С. Яшан

Керівник проекту Я. В. Хіцко

АНОТАЦІЯ

Даний дипломний проект присвячено розробці інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів. Система має логічний поділ на три частини: серверну та два клієнтських застосунки. Серверна частина обробляє запити клієнтів та надає клієнтам інформацію, яка міститься у БД у форматі JSON.

Веб-сайт – один з клієнтських застосунків інформаційної системи, мета котрого надати можливість адміністрування системи, тобто додавання, видалення чи оновлення інформації про продукти та їх компоненти, а також надати користувачам можливість переглянути інформацію про продукти та компоненти, налаштувати особливості свого раціону, переглянути історію з’їдених продуктів та поглиблено проаналізувати продукти, які вони споживають. Поглиблений аналіз базується на трьох показниках: продукти, які споживаються більш часто; чи споживаються продукти, які містять заборонені для користувача компоненти; як часто вживаються цукоровмісні продукти.

Функціональність мобільного застосунку надає користувачу можливість отримати інформацію про продукти та компоненти, налаштувати заборонені компоненти, проаналізувати продукт у контексті його корисності для споживача.

Результатом роботи над даним дипломним проектом стали: розроблена архітектура системи, дизайн та програмна реалізація. Також були розроблені три алгоритми аналізу корисності продукту для користувача.

ABSTRACT

This diploma project is devoted to the development of an information system for analyzing and forecasting the impact of food products on the health of consumers. The system is logically divided into three parts: server and two client applications. A server side processes the queries of clients and provides information to the clients, which is contained in DB in the JSON format.

Website, which is one of client applications, allows administering the system: add, delete or update information of the products and their components, and allow users to view information about the products and components to customize features of diet, see history of eaten products and deep analysis of the products they consume. The deep analysis is based on three indicators: products that are consumed most often; products with allergic components that are in history list, frequency of consumption sugar-based products.

The functionality of the mobile application gives the user the opportunity to get information about the products and components, to configure the allergic components, to analyze product`s utility for the consumer.

The results of work on this diploma project includes developed architecture of the system, created design and software implementation. In addition, three algorithms, for analyzing the utility of the product for the user, were developed.

ДП.045490-01-90 Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045490-02-91	Інформаційна система для	4	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Технічне завдання		
ДП.045490-03-81	Інформаційна система для	76	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Пояснювальна записка		
ДП.045490-04-51	Інформаційна система для	4	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Програма та методика		
	тестування		
ДП.045490-05-34	Інформаційна система для	17	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045490-06-99	Інформаційна система для	1	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Пошук серед спожитих		
	продуктів таких, що		
	містять заборонені		
	компоненти. Схема		
	алгоритму		
ДП.045490-07-99	Інформаційна система для	1	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Обрахунок кількості		
	спожитих цукоровмісних		
	продуктів. Схема		
	алгоритму		
ДП.045490-08-98	Інформаційна система для	1	
	аналізу та прогнозування		
	впливу продуктів		
	харчування на		
	здоров'я споживачів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ'Я СПОЖИВАЧІВ

Технічне завдання

ДП.045490-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.С. Яшан

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для органів якості контролю продуктів, з метою надання ресурсів для зберігання та оновлення даних про продукти та компоненти, а також для споживачів для швидкого доступу до даних у цій системі та аналізу складу продуктів харчування.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-ресурс повинен забезпечувати такі основні функції:

1) перегляд інформації про продукт;
2) перегляд інформації про компонент;
3) оновлення інформації про продукт;
4) оновлення інформації про компонент;
5) налаштування особливостей раціону користувача;
6) аналіз впливу спожитих продуктів на здоров’я споживача.

Мобільний додаток повинен забезпечувати такі основні функції:

1) перегляд інформації про продукт;
2) перегляд інформації про компонент;
3) налаштування особливостей раціону користувача;
4) зчитування штрих-коду продукту.

Розробку веб-ресурсу та мобільного додатку виконати за допомогою технології React.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Архітектура веб-ресурсу. Діаграма розгортання»;
- «Структура веб-сайту. Схема сторінок та переходів веб-сайту».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2018

Розробка та узгодження технічного завдання…….....28.11.2018

Розробка структури веб-ресурсу та мобільного додатку.....15.12.2018

Розробка дизайну веб-сайту та мобільного додатку……...14.01.2019

Програмна реалізація веб-ресурсу та мобільного додатку.....15.03.2019

Тестування веб-ресурсу та мобільного додатку.……....31.03.2019

Підготовка матеріалів текстової частини проекту……..….30.04.2019

Підготовка матеріалів графічної частини проекту……..…14.05.2019

Оформлення технічної документації проекту……..…31.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ'Я СПОЖИВАЧІВ

Пояснювальна записка

ДП.045490-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.С. Яшан

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Авторизація – керування рівнями та засобами доступу до певного захищеного ресурсу.

Атомарність – характеристика, яка означає неперервність та цілісність.

Браузер – програмне забезпечення для електронного пристрою, під'єднаного до Інтернету, що дає можливість взаємодіяти з текстом, малюнками або іншою інформацією на гіпертекстовій веб-сторінці.

Валідація – процес підтвердження відповідності або надання законної сили.

Верифікація – доказ того, що вірогідний факт або твердження є істинним.

Віджет – підпрограма, що вбудовується у веб-сторінку або у браузер.

Драйвер – комп'ютерна програма, за допомогою якої операційна система отримує доступ до приладу апаратного забезпечення.

Інтерпретатор – програма чи технічні засоби, необхідні для виконання інших програм.

Колбек-функція, callback-функція – є частиною виконуваного коду, що передається як аргумент до іншого коду, який має викликати цей код у відповідь.

Компіляція – трансляція вихідного коду програми в бінарний код.

Конфігурація – сукупність налаштувань програми, що задається користувачем, а також процес зміни цих налаштувань відповідно до потреб користувача.

Кросплатформність – властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі.

Метрика – міра, що дозволяє отримати чисельне значення деякої властивості.

Модальне вікно – вікно, що блокує роботу користувача з батьківським застосунком доти, доки користувач це вікно не закриє.

Нативний додаток – це програма, розроблена для використання на певному пристрої чи платформі.

Офлайн – стан, у якому пристрій не має з'єднання з мережею зв'язку.

Персоналізація – налаштування з урахуванням потреб конкретного користувача.

Рендеринг – отримання зображення за моделлю з допомогою комп'ютерної програми.

Реплікація – це механізм розподілу даних за вузлами, що дозволяє зберігати копії тих самих даних на різних вузлах мережі.

Роутинг, маршрутизація, навігація – процес визначення шляху.

Скрипт – послідовність команд.

Тест-кейс – це ситуація, яка перевіряє конкретну умова з вимог.

Фреймворк – заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи.

Хостинг – ресурс, який надає послуги надання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі.

API – опис способів, якими одна комп'ютерна програма може взаємодіяти з іншою програмою.

Cookie – інформації у вигляді текстових або бінарних даних, отриманих від веб-сайту, яка зберігається у клієнта, тобто браузера, а потім відправляється на той самий сайт, якщо його буде повторно відвідано.

DOM, DOM-дерево – це представлення документа у вигляді дерева об'єктів, доступне для зміни за допомогою мови програмування JavaScript.

Drag-n-drop (перетягування) – форма виконання певних дій у графічних інтерфейсах користувача, що передбачає використання комп'ютерної миші або сенсорного екрана.

ВСТУП

Здоров’я, незаперечно, є найбільшою цінністю кожної людини. Однак ми сьогодні вимушені жити в умовах несприятливих для життєдіяльності, адже протягом десятиріч руйнували комфорт та безпеку нашого перебування на планеті Земля. Шкідливі випари, глобальне забруднення, надмір хімічних речовин у середовищі – чинники неухильного зростання числа хворих, що страждають від астми, різних алергійних захворювань та раку.

Одним з ключових негативних наслідків розвитку і прогресу – є постійне виробництво шкідливі речовин – хімікатів, котрими виробники продуктів заміняють натуральні компоненти з метою заробітку. Завданням людства загалом і насамперед є виховання відповідального ставлення до співмешканців нашої планети, адже цінність здоров’я кожного повинна стояти вище ніж вигода одного з нас. Програмні інженери у свою чергу можуть вплинути на вирішення даної проблеми створенням допоміжних програмних засобів, адже завдяки розвитку інформаційних технологій існує безліч можливостей узяти під контроль усі сфери життя, зокрема, сферу якості продуктів харчування.

Тема харчування та його впливу на здоров’я стає дедалі популярнішою не тільки серед науковців, лікарів, але й серед пересічних громадян, тому також важливим завданням є спростити процес аналізу продуктів харчування та вбудувати його у ледь не щоденний та рутинний процес походу в магазин.

Отже, завданням даної дипломної роботи є реалізація програмного засобу – інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів, яка вирішить наступні завдання:

1) допомогти пересічному споживачу аналізувати якість продуктів харчування;

2) допомогти органам якості контролю продуктів харчування контролювати якість продуктів харчування.

1. ОПИС, АНАЛІЗ ТА ОБГРУНУТВАННЯ ВИБОРУ МОВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-САЙТУ ТА МОБІЛЬНОГО ДОДАТКУ

1.1. Опис та аналіз мов: JavaScript, HTML, CSS

Програмний застосунок має логічний поділ на клієнтську та серверну частину. Браузерний веб-додаток та мобільний додаток є клієнтами. Серверна частина представлена веб-сервером, який «спілкується» з СКБД (Системою керування базами даних). Кожна логічна одиниця застосунку була розроблена з допомогою притаманних саме їй технологій, втім спільною рисою стала мова програмування JavaScript, обрана для розробки.

1.1.1. *Опис та аналіз мови програмування JavaScript*

Програми, написані цією мовою, називаються скриптами – вони приєднуються до елементів сторінки, які описані у файлі з використанням мови розмітки HTML, і виконуються одразу після завантаження сторінки [1]. Програми написані мовою JavaScript можна виконати за допомогою програми, яка має назву «інтерпретатор». Процес виконання програми на мові JavaScript має назву «інтерпретація». Інтерпретатор виконує построковий чи покомандний переклад у машинний код команд, написаних мовою JavaScript. Сучасні інтерпретатори перетворюють JavaScript у машинний код, тоді оптимізують його, а тоді тільки виконують, тому програми написані на мові програмування JavaScript працюють надзвичайно швидко.

Сучасний JavaScript – це «безпечна» мова програмування, адже не надає програмісту можливості керувати пам’яттю чи процесором комп’ютера. Така особливість зумовлена тим, що JavaScript був орієнтований на виконання у браузерах, де такі можливості не були потрібні.

Можливості та переваги JavaScript розкриваються в контексті середовища, в якому виконується код.

У всі основні браузери вбудований інтерпретатор JavaScript, саме тому вони можуть виконувати скрипти на сторінці. В браузері JavaScript може робити усе, що відноситься до маніпуляцій зі сторінкою, взаємодією з користувачем та, певною мірою, із сервером:

- створювати нові HTML-теги, видаляти існуючі, змінювати стилі елементів, ховати та відображати елементи;
- реагувати на дії користувача, обробляти такі події як клік миші, рух вказівника, натискання на клавіші клавіатури;
- відправляти запити на сервер і завантажувати дані без перезавантаження сторінки;
- отримувати та встановлювати cookie, запитувати дані та виводити повідомлення.

Унікальність мови програмування JavaScript для розробки веб-сайтів полягає у трьох особливостях:

1. цілковита інтеграція з HTML/CSS;
2. прості речі робляться просто;
3. підтримується усіма розповсюдженими браузерами.

Ці три речі не містить в собі жодна інша браузерна технологія.

1.1.2. *Опис мови розмітки HTML*

HTML (Hypertext Markup Language) – це стандартна мова розмітки для створення веб-сторінок та веб-додатків. Веб-браузер отримує HTML документ з веб-сервера чи з локального сховища та відображає документ у вигляді мультимедійної веб-сторінки [2]. HTML описує та визначає структуру веб-сторінки за допомогою HTML-елементів. HTML-елементи є будівними блоками для HTML-сторінок. З допомогою семантичних конструкцій HTML, зображення та інші об’єкти, такі як інтерактивні форми можуть бути вбудовані у сторінку. HTML надає засоби для

створення структурованих документів за допомогою семантичних конструкцій для тексту, таких як заголовки, параграфи, списки, посилання, цитати та ін. HTML-елементи виділені тегами, які пишуться у трикутних дужках.

HTML може вбудовувати на сторінки програми, написані скриптовими мовами, такими як JavaScript, які змінюють поведінку та вміст веб-сторінок.

1.1.3. *Опис мови стилів CSS*

CSS (Cascading Style Sheets) – мова стилів, яка використовується для опису зовнішнього вигляду документу, описаного мовою розмітки HTML. CSS є наріжним каменем технологій World Wide Web, поряд з HTML і JavaScript. CSS розроблений для розділення презентаційної та контентної частини веб-сторінки [3]. Таке розділення надає можливість відображення однієї веб-сторінки з різними стилями для різних методів її рендерингу, таких як:

- на екрані;
- у друкованому вигляді;
- голосовому;
- на тактильних екранах на основі шрифту Брайля.

Слово «каскадний» у назві означає те, що є спеціальна схема пріоритетності, яка визначає яке правило застосовувати, якщо більше ніж одне правило належить до одного елемента. Ця каскадна пріоритетна схема однозначно та передбачувано визначає стилі елементів сторінки.

1.2. Опис та аналіз технології для серверної частини Node.js

1.2.1. *Опис технології Node.js*

Серверна частина інформаційної системи була розроблена з допомогою технології Node.js, крос-платформне середовище виконання

JavaScript, яке виконує код поза браузером. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для написання скриптів серверної частини, відтворення скриптів на стороні сервера для створення динамічного вмісту веб-сторінки до того, як сторінка буде надіслана до веб-браузера користувача. В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням [4].

Отже, Node.js сповідує парадигму «JavaScript скрізь», що об'єднує розробку веб-додатків навколо однієї мови програмування, а не написання серверних та клієнтських скриптів різними мовами.

1.2.2. *Порівняльний аналіз технологій Node.js та PHP*

Node.js та PHP є одними з технологій для розробки веб-сайтів, які використовуються найчастіше. PHP – це скриптова мова програмування для написання програм на стороні сервера [5]. Node.js в свою чергу є середовищем виконання програм, написаних мовою JavaScript.

Порівняльний аналіз цих технологій наведений у табл. 1.

Таблиця 1

Порівняльний аналіз Node.js та PHP

	Node.js	PHP
Підтримка	Зрозуміла та повна документація, але молодість технології зумовило менший огляд її особливостей та можливих проблем на тематичних форумах. Велике різноманіття середовищ розробки, редакторів та валідаторів.	Величезне керівництво користувача на спеціалізованому сайті php.net. Технологію активно використовують вже понад 20 років, тому знайти відповідь на будь-яке питання дуже легко.

Продовження табл. 1

	Node.js	PHP
Інструменти розробника	Також наявний зручний інструмент керування пакетами – npm, який до того ж вбудований за замовчування.	Велике різноманіття середовищ розробки, редакторів та валідаторів. Також наявний менеджер пакетів, який був створений під впливом npm – Composer, однак його доведеться вбудовувати самостійно.
Інтеграція	Ця технологія «розуміє» JSON, який є головним форматом обміну даними у мережі Інтернет або взаємодії з NoSQL базами даних. Дані можуть надходити частинами без необхідності переформатування.	Розвиток технології відбувався протягом багатьох років, тому PHP дозволяє інтеграцію з базами даних та драйверами з допомогою API.
Хостинг	Потрібно знайти спеціалізований хостинг.	Будь-який хостинг інтернету підтримує PHP.
Продуктивність	Є продуктивнішим за PHP, адже: 1. потребує менше залежностей; 2. швидше інтерпретується завдяки движку V8 від Google 3. Застосунок працює постійно і його потрібно ініціалізувати тільки 1 раз; 4. Має неблокуючий потік вводу/виводу. Тобто не потрібно чекати завершення процесу, а	Є менш продуктивним за Node.js, адже: 1. потребує багато залежностей; 2. виконує звичайну клієнт-серверну модель. Кожен запит сторінки ініціює новий застосунок, завантаження параметрів підключення до бази даних, отримання інформації та відображення HTML-коду.

Продовження табл. 1

	Node.js	PHP
	можна створити функцію зворотнього виклику, яка буде прослуховувати процес і виконається після завершення дії.	3. Використовує очевидну модель блокування: коли відбувається запит у базу даних, процес запиту виконається і перейде до обробки наступної дії після закінчення поточного процесу.
Підтримка	Зрозуміла та повна документація, але молодість технології зумовило менший огляд її особливостей та можливих проблем на тематичних форумах.	Величезне керівництво користувача на php.net. Технологію активно використовують вже понад 20 років, тому знайти відповідь на будь-яке питання дуже легко.

1.3. Опис та аналіз технології для частини веб-клієнта React.js

1.3.1. *Опис технології React.js*

Браузерна клієнтська частина була реалізована з допомогою технології React.js – це бібліотека, написана мовою JavaScript, для створення користувацьких інтерфейсів [6]. React.js може бути використаний як основа для розробки односторінкових застосунків або мобільних додатків, адже вона є оптимальним способом для отримання та відображення даних, які часто змінюються без перезавантаження сторінки. Вона покликана вирішувати проблему часткового оновлення вмісту веб-сторінки. Однак, для керування станом, роутингу та взаємодії з API React.js потребує додаткові бібліотеки.

React реалізує тільки користувацьку частину застосунку – інтерфейс, а отже відповідає шаблону, який є стандартом у веб-розробці, MVC (модель-вид-контролер) [7].

Особливості технології React:

1. Одностороння передача даних

   Властивості компоненту передаються у його рендер-функцію, змінювати їх компонент може за допомогою колбек-функцій.

2. Віртуальний DOM

   React створює віртуальний DOM, завдяки якому і визначає змінені частини DOM-дерева, порівнюючи із попереднім віртуальним DOM-деревом.

3. JSX

   Компоненти написані з допомогою мови розмітки JSX, яка є дуже схожою на об’єднання XML та JavaScript.

4. Життєвий цикл компонента

   Кожен компонент, написаний з допомогою React, має етапи свого існування і на кожному з них може виконувати відповідні дії. Етапи існування React-компонента:

   - Mount – на цьому етапі відбувається ініціалізація компонента та рендеринг.
   - Update – цей етап починається щоразу, коли змінюється стан компонента або його властивості.
   - Unmount – це останній етап життєвого циклу компонента, на цьому етапі компонент видаляється з DOM-дерева.

1.3.2. *Порівняльний аналіз технологій React та Angular*

На разі у веб-розробці панівними технологіями для створення користувацьких інтерфейсів є бібліотека React.js та фреймворк Angular, тому нижче приведений порівняльний аналіз React.js та Angular.

Переваги бібліотеки React.js:

1. Зручність для користувача

Висока динамічність користувацького інтерфейсу, а саме швидке оновлення частин інтерфейсу, які здійснюються маніпуляціями користувача, завдяки використанню віртуального DOM.

2. Економія часу

   Можливість повторно використовувати частини коду та оновлювати їх без значних змін інших частин, що досягнуто завдяки особливій організації коду.

3. Швидкість розробки

   Швидке перезавантаження додатку після змін, внесених у код, та відображення їх у браузері значно прискорює розробку.

4. Швидкість тестування

   Поряд з бібліотекою React використовується бібліотека Redux, яка не тільки дозволяє налаштувати обмін даними між компонентами завдяки глобальному сховищу даних, а й надає зручні браузерні інструменти для відслідковування змін цього сховища і таким чином можливе швидке та зручне тестування веб-додатку [8].

5. Стабільність досягнута, завдяки однонаправленому зв’язуванню даних

   React.js дозволяє розробникам працювати безпосередньо з компонентами і використовувати зв'язування даних однонаправлено, щоб гарантувати, що батьківські об'єкти не потрапляють під вплив змін у дочірніх об'єктах. Такий підхід робить код стабільним і має потенціал стати панівним у майбутньому.

Переваги фреймворку Angular:

1. Зрозумілий код

   Angular використовує мову програмування TypeScript, яка є покращеною та розширеною версією JavaScript. Він дозволяє

писати код більш зрозумілим чином та уникати помилок, які впливають на якість та продуктивність програми [9].

2. Висока продуктивність

   Angular має ієрархічну залежність ін'єкцій, в якій класи не залежать один від одного, а звертаються до зовнішніх джерел, що забезпечує більш високу продуктивність для мобільних додатків.

3. Дизайн в стилі «Material Design»

   Angular надає заздалегідь реалізовані елементи, такі як елементи навігації, форми, спливаючі вікна, таблиці зі стилем «Material Design», який використовують у застосунках компанії Google, що значно спрощує розуміння інтерфейсу для користувачів [10].

1.4. Опис та аналіз технології для мобільної клієнтської частини React Native

1.4.1. *Опис технології React Native*

Мобільна користувацька частина була розроблена з допомогою технології React Native. React Native – це фреймворк з відкритим кодом для розробки мобільних застосунків. Він використовується для розробки мобільних застосунків з операційною системою Android, iOS та UWP, надаючи розробникам використовувати можливості, які надає React у браузері, а також можливості самої платформи.

1.4.2. *Порівняльний аналіз підходів до розробки мобільних застосунків*

У світі розробки мобільних застосунків панує підхід створення нативних додатків для платформи iOS, використовуючи мову Objective-C чи Swift, а для Android – Java, втім розвиток технологій не стоїть на місці і з появою React Native з'явилася можливість писати застосунки для обидвох

платформ, використовуючи мову JavaScript, відтак нижче я порівняю два підходи до розробки мобільних застосунків у табл. 2.

Таблиця 2

Порівняльний аналіз React Native та Real Native Apps

React Native	Real Native Apps
Плюси	
1. Повторне використання частин коду [11].	1. Відсутні API обмеження, тобто розробник має доступ до всіх можливостей апаратної частини мобільного телефону [14].
2. Це фреймворк, який містить у собі всі необхідні для розробки інструменти: React, допоміжні засоби для Android та iOS, Flexbox для стилізації користувацького інтерфейсу, різноманітні віджети та анімації [12].	2. Різноманіття ресурсів для розробки, тобто велика кількість сторонніх бібліотек доступні для використання під час розробки, що дозволяє створювати більш функціональні додатки [14].
3. Інтегровані нативні компоненти, тобто нативні елементи інтерфейсу платформи	
Мінуси	
1. Наявні API обмеження, тобто певні функції потрібно реалізовувати з допомогою нативної мови платформи та інтегрувати їх в React Native компоненти.	1. Величезний мінус у тому, що відмінність двох платформ: Android та iOS, зумовлює написання різних програм, які неможливо використовувати разом.
2. Відмінності платформ Android та iOS накладають певні додаткові вимоги у спосіб написання коду додатку [13].	

Продовження табл. 2

3. Відносно низька продуктивність, яка не задовольнить усі вимоги, якщо необхідно реалізувати обробку зображень чи відео.	

1.5. Опис та аналіз системи керування базами даних MongoDB

1.5.1. *Опис СКДБ MongoDB*

Сховище даних було реалізовано за допомогою системи керування базами даних MongoDB – це документо-орієнтована система керування базами даних з відкритим вихідним кодом. Належить до класифікації NoSQL, що означає відмінний підхід до керування даними, який використовується у традиційних реляційних СКБД. Документо-орієнтовані СКБД слугують для зберігання ієрархічних структур даних. MongoDB використовує JSON-подібні документи та схему баз даних.

Основні риси нереляційних баз даних:

1. базова доступність – кожен запит гарантовано завершиться (успішно чи не успішно);
2. гнучкий стан – стан системи може змінюватися з часом, навіть без введення нових даних, для досягнення узгодженості даних.
3. узгодженість в результаті – дані можуть бути неузгодженими, але це змінюється з часом;
4. можливість розробки бази даних без знання схеми на початку.
5. лінійна масштабованість – збільшення процесорів збільшує продуктивність [15].

1.5.2. *Аналіз СКДБ MongoDB*

Переваги:

1. Гнучка база даних: MongoDB – це база даних, яка не містить схем. Це означає, що ми можемо мати будь-який тип даних в окремих

документах. Ця особливість дає нам гнучкість і свободу зберігання даних різних типів.

2. Шардінг – можливість зберігати великі данні, поділивши їх між кількома серверами. Якщо один сервер не зможе працювати з певним обсягом даних, він розподілить їх по іншим серверам і це переповнення ніяк не вплине на роботу всього застосунку.
3. Висока швидкість: MongoDB – це документо-орієнтована база даних. Доступ до документів легко здійснюється шляхом індексації. Таким чином, MongoDB забезпечує швидку відповідь на запит. Швидкість MongoDB в 100 разів швидша, ніж реляційних баз даних.
4. Висока доступність: MongoDB надає можливість реплікації, що дозволяє збільшити доступність даних, а отже і продуктивність застосунку. Реплікація також підвищує захищеність даних, адже копії даних зберігаються на декількох серверах [16].

Недоліки:

1. Використання великого обсягу пам’яті за рахунок збереження ключів для кожної пари значень, а також надмірність даних.
2. Обмеження на обсяг документу: не більше 16 МБ.
3. Обмеження на рівень вкладеності: не більше ніж 100 рівнів вкладеності [17].

1.6. Обгрунтування вибору технологій для розробки інформаційної системи

Отже, для розробки інформаційної системи було обрано наступні технології:

- Node.js для реалізації серверної частини

  Вибір технології був зумовлений вибором мови програмування – JavaScript, наявним у Node.js менеджером пакетів – npm, завдяки якому розробка значно спрощується, адже він надає можливість

швидко знайти та завантажити необхідну бібліотеку та встановити її на проект. Останньою та не менш важливою причиною стало, те що Node.js «розуміє» формат даних JSON та надає вичерпну функціональність для обробки даних у такому форматі.

- React.js для реалізації браузерної користувацької частини
  React.js було обрано для розробки, адже ця бібліотека написана на JavaScript. React.js може швидко оновлювати частини інтерфейсу, які змінилися внаслідок дій користувача, що є надзвичайно важливо для інформаційної системи, яка покликана надавати користувачу інформацію на його вимогу. Швидкість розробки за допомогою React.js завдяки компонентам, які можна використовувати повторно, також зіграло роль, адже система має велику частину елементів інтерфейсу, які повторюються на різних сторінка: форми і таблиці.
- React Native для реалізації мобільної користувацької частини
  Вибір даної технології зумовив вибір мови – JavaScript та бібліотеки для розробки браузерної клієнтської частини React.js. React Native в контексті розробки має ті ж переваги, що й React.js.
- MongoDB для керування базою даних
  MongoDB була обрана адже, вона використовує формат даних JSON та є швидшою за реляційні СКБД завдяки індексації, а швидкий доступ до даних є надзвичайно важливою характеристикою для інформаційної системи.

Поряд з вищезгаданими технологіями варто перерахувати інші важливі аспекти розробки, а саме обране середовище розробки Visual Studio Code, систему керування версіями коду Git та хмарну PaaS-платформу Heroku.

2. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ'Я СПОЖИВАЧІВ

2.1. Вимоги до функціональності системи

Розробка вимог – очевидно перший крок створення будь-якого програмного засобу. Це єднальна ланка між потребами зацікавлених сторін та технологічним рішенням. Метою процесу розробки є отримання корисного, зручного, а головне – необхідного користувачам результату. Саме правильні, однозначні і повні вимоги стануть гарантом розуміння наших цілей, як розробників, їх застосування – ґрунт для перевірки готовності рішення та досягнення поставленої мети.

Керування вимогами багатоетапний та складний процес, він вимагає частої комунікації з кожною зацікавленою стороною у розробці технологічного рішення, чіткого і систематизованого ведення реєстру вимог, однозначного і повного їх опису, глибокого аналізу пріоритетності та доцільності, використаних для реалізації, технологій чи ресурсів, широкого розуміння глобальної цілі для пошуку компромісних рішень, щодо суперечливих аспектів вимог, відслідковування дотримання кожної затвердженої вимоги, можливість гнучкої зміни чи коригування процесу розробки, за умови виникнення додаткових вимог чи зміни у початкових.

Складність процесу керування полягає у стримуванні розробки в певних рамках, окреслених вимогами, за для швидшої та більш результативної реалізації проекту. Хоча й на перших етапах роботи над проектом не вдається повноцінно описати усі вимоги, дуже важливо перевірити наявні вимоги та продовжувати розробку нових згідно з такими критеріями:

- єдиність;
- завершеність;
- атомарність;

- можливість відстеження;
- актуальність;
- послідовність (несуперечливість);
- однозначність;
- обов’язковість;
- верифікованість.

Необхідним етапом формування вимог є поділ на різноманітні категорії, які доцільні з огляду програмного засобу, який планується розробити, та процесів розробки, які мають місце у проекті:

- пріоритетність – вказує на важливість та першочерговість реалізації;
- системний рівень – вказує, на якій ітерації розробки та котрому спеціалісту доручено реалізацію;
- специфіка (наприклад вимоги до функціональності, керування, інтерфейсу тощо);
- відповідальні за створення чи реалізацію вимоги.

Такий чіткий та деталізований розгляд вимог допоможе оптимізувати та поліпшити процес розробки, а саме: скоротити обсяг допрацювань, пришвидшити реалізацію програмного засобу, підвищити продуктивність розробників, в наслідок чіткого розуміння поставлених завдань, зменшення ризиків та витрат завдяки оперативному реагуванню на зміни та нові умови.

2.1.1. *Високорівневі бізнес-вимоги*

Веб-ресурс повинен забезпечувати такі основні функції:

1) перегляд інформації про продукт;
2) перегляд інформації про компонент;
3) оновлення інформації про продукт;
4) оновлення інформації про компонент;

5) налаштування особливостей раціону користувача;

6) аналіз впливу спожитих продуктів на здоров’я споживача.

Мобільний додаток повинен забезпечувати такі основні функції:

1) перегляд інформації про продукт;

2) перегляд інформації про компонент;

3) налаштування особливостей раціону користувача;

4) зчитування штрих-коду продукту.

Розробку веб-ресурсу та мобільного додатку виконати за допомогою технології React.

2.1.2. *Вимоги до функціональності серверної частини*

Таблиця 3

Вимоги до функціональності серверної частини

№	Опис	Пріоритет
1.	Створити запис нового користувача у БД	1
2.	Знайти та повернути запис про користувача з БД за такими параметрами: логін, пароль	1
3.	Знайти та оновити запис про користувача у БД за такими параметрами: ім’я	1
4.	Повернути усі записи про продукти з БД	1
5.	Створити запис нового продукту у БД	1
6.	Знайти та повернути запис про продукт з БД за такими параметрами: назва	2
7.	Знайти та повернути запис про продукт з БД за такими параметрами: штрих-код	1
8.	Знайти та оновити запис про продукт у БД за такими параметрами: назва	1
9.	Знайти та видалити запис про продукт з БД за такими параметрами: назва	1

Продовження табл. 3

№	Опис	Пріоритет
10.	Повернути усі записи про компоненти з БД	1
11.	Створити запис нового компоненту у БД	1
12.	Знайти та оновити запис про компонент у БД за такими параметрами: назва	1
13.	Знайти та видалити запис про компонент з БД за такими параметрами: назва	1
14.	Повернути усі записи про дієти з БД	1
15.	Створити запис нової дієти у БД	1
16.	Знайти та оновити запис про дієту у БД за такими параметрами: назва	2
17.	Знайти та видалити запис про дієту з БД за такими параметрами: назва	2

2.1.3. *Вимоги до функціональності частини веб-клієнта*

Таблиця 4

Вимоги до функціональності барузерної користувацької частини

№	Опис	Пріоритет
1.	Здійснити авторизацію у системі	1
2.	Зареєструватися у системі	1
3.	Вийти з облікового запису	1
4.	Переглянути продукти	1
5.	Переглянути інформацію про продукт	1
6.	Переглянути компоненти	1
7.	Переглянути інформацію про компонент	1
8.	Додати продукт	1
9.	Додати компонент	1
10.	Видалити продукт	1
11.	Видалити компонент	1
12.	Здійснити пошук продукту за назвою	2
13.	Додати переглянуті користувачем продукт в розділ сторінки користувача «Останні переглянуті продукти» в послідовності їх перегляду	2
14.	Налаштувати «Заборонені компоненти» для користувача	1
15.	Підсвічувати продукти, які містять заборонені компоненти для користувача, на сторінці користувача у розділі «Останні переглянуті продукти»	2
16.	Налаштувати «Дієту» для користувача	1

Продовження табл. 4

№	Опис	Пріоритет
17.	Переглянути історію з’їдених продуктів користувача	1
18.	Проаналізувати історію з’їдених продуктів користувача за такими параметрами: Продукти, які користувач їсть найчастіше; Чи їсть користувач заборонені для нього продукти і які саме; Яку кількість цукоровмісних продуктів вживає користувач в середньому в день.	1

2.1.4. *Вимоги до функціональності мобільної користувацької частини*

Таблиця 5

Вимоги до функціональності мобільної користувацької частини

№	Опис	Пріоритет
1.	Здійснити авторизацію у системі	1
2.	Вийти з облікового запису	1
3.	Переглянути продукти	1
4.	Переглянути інформацію про продукт	1
5.	Переглянути компоненти	1
6.	Переглянути інформацію про компонент	1
7.	Додати переглянуті користувачем продукт в розділ сторінки користувача «Останні переглянуті продукти» в послідовності їх перегляду	2
8.	Налаштувати «Заборонені компоненти» для користувача	1

Продовження табл. 5

9.	Відсканувати штрих-код продукту та отримати інформацію про продукт	1
10.	Отримати попередження після сканування штрих-коду	1
11.	Додати відсканований продукт в історію	1

2.1.5. *Вимоги до апаратного та програмного забезпечення*

1) Наявність мережі Інтернет у місцевості використання мобільного додатку.
2) Наявність мережі Інтернет у місцевості використання веб-ресурсу.
3) Наявність браузеру на ПК: Google Chrome, Safari, Mozilla Firefox.
4) Обсяг вільної пам'яті сервера: не менше 5 Гб.
5) Обсяг оперативної пам'яті сервера: не менше 500 Мб.
6) Операційна система мобільного телефону: Android версії 4.0.1 і новіших.
7) Обсяг вільної пам’яті мобільного телефону:
   a. фізичної для збереження додатку: 20 Мб,
   b. оперативної для роботи додатку: 500 Мб.
8) Діагональ екрану мобільного телефону від 4 дюймів.
9) Наявність Wi-Fi інтерфейсу в мобільного телефону: підтримувані стандарти зв'язку: 2G/3G/ 4G.
10) Камера мобільного телефону: від 4 Мп.

2.2. Архітектура системи

Проектування архітектури програмного засобу виконується одразу після визначення, аналізу та формулювання вимог. Метою даного проектування є перетворення вимог до системи у вимоги до ПЗ і побудова на цій основі архітектури системи.

2.2.1. *Огляд клієнт-серверної архітектури*

У створенні розподілених мережних застосунках домінуючою є клієнт-серверна архітектура, яка передбачає такі основні компоненти:

- сервери,
- клієнти,
- мережа.

та взаємодію між ними. Метою даного архітектурного рішення є розподіл навантаження між постачальниками послуг – серверами та споживачами послуг – клієнтами. Фактично клієнт та сервер – це програмне забезпечення, зазвичай розташоване на різних обчислюваних машина, яке взаємодіє між собою через мережу за допомогою мережевих протоколів. Програми-сервери очікують від клієнтських програм запити і надають їм свої ресурси у вигляді даних або у вигляді сервісних функцій.

Застосований у даній дипломній роботі різновид клієнт-серверної архітектури – трирівнева архітектура, наведена на рис. 1.

Трирівнева клієнт-серверна архітектура – це модель програмного комплексу, яка складається з трьох компонентів:

- клієнт – це інтерфейсний (графічний) компонент, який представляє перший рівень, власне застосунок для кінцевого користувача. Перший рівень не має прямих зв'язків з базою даних (за вимогами безпеки), не навантажений основною бізнес-логікою (за вимогами масштабованості) і не зберігає стан програми (за

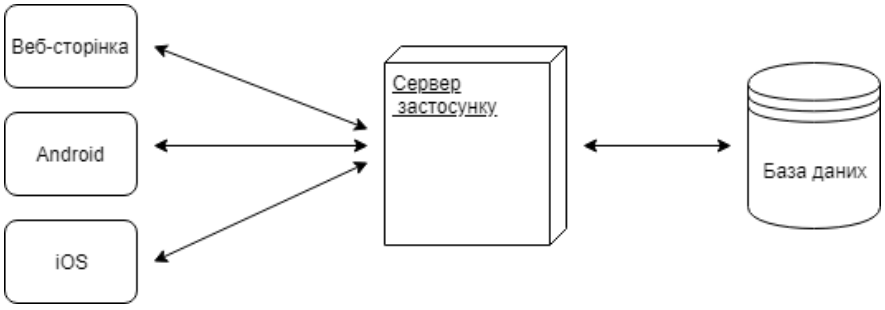


Рис. 1. Схема трирівневої клієнт-серверної архітектури

вимогами надійності). На перший рівень винесена найпростіша бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції (сортування, групування, підрахунок значень) з даними.

- сервер застосунку, до якого підключені клієнтські застосунки – розташований на другому рівні, де зосереджена більша частина бізнес-логіки. Другий рівень – це програмний інтерфейс, що зв'язує клієнтські компоненти з прикладною логікою бази даних.
- сервер бази даних, з яким працює сервер застосунку, забезпечує зберігання даних і виноситься на третій рівень. Зазвичай це реляційна або об'єктно-орієнтована СУБД [18].

Клієнти бувають трьох видів:

1. «Тонкий клієнт» – визначає клієнта, обчислювальних ресурсів якого достатньо лише для запуску необхідного мережевого додатки через інтерфейс. Інтерфейс такого додатка формується засобами статичного HTML (виконання JavaScript не передбачено), вся прикладна логіка виконується на сервері.
2. «Товстий клієнт». Таким є робоча станція або персональний комп'ютер, які працюють під управлінням власної дискової операційної системи і мають необхідний набір програмного забезпечення. До мережевих серверів товсті клієнти звертаються в основному за додатковими послугами (наприклад, доступ до web-серверу або бази даних).
3. «Rich-клієнт». Останнім часом більш часто почав використовуватися ще один термін: rich-клієнт. Rich-клієнт свого роду компроміс між товстим і тонким клієнтом. Як і тонкий клієнт, rich-клієнт також представляє графічний інтерфейс, що описується вже засобами XML і включає деяку функціональність товстих клієнтів (наприклад, інтерфейс drag-and-drop, вкладки,

множинні вікна, що випадають меню і т.п.). Прикладна логіка rich-клієнтів також реалізована на сервері. Дані відправляються в стандартному форматі обміну, на основі того ж XML і інтерпретуються клієнтом.

2.2.2. *Порівняння трирівневої архітектури з дворівневою та файл-серверною архітектурами*

Трирівнева архітектура забезпечує:

- більшу масштабованість, завдяки горизонтальному масштабуванню сервера застосунків і мультиплексуванню з'єднань;
- кращу можливість конфігурування, завдяки ізольованості рівнів один від одного;
- високий рівень надійності;
- високий рівень безпеки;
- низькі вимоги до швидкості каналу (мережі) між клієнтами і сервером застосунків;
- низькі вимоги до продуктивності і технічних характеристик клієнтів, як наслідок зниження їхньої вартості. Клієнтом може виступати не тільки комп'ютер, але і, наприклад, мобільний телефон.

Недоліки трирівневої архітектури:

- вища складність реалізації застосунків;
- складніша у розгортанні і адмініструванні;
- високі вимоги до продуктивності серверів застосунків і сервера бази даних, як наслідок висока вартість серверного обладнання;
- високі вимоги до швидкості каналу (мережі) між сервером бази даних і сервером застосунку.

2.2.3. *Архітектура інформаційної системи*

Архітектура інформаційної системи складається із чотирьох вузлів:

- серверна частина – сервер застосунку;
- браузерна клієнтська частина – клієнт (вид: товстий клієнт);
- мобільна клієнтська частина – клієнт (вид: товстий клієнт);
- сервер даних – сервер бази даних.

Взаємодія відбувається між Клієнтами та Сервером застосунку, між Сервером застосунку та Сервером бази даних, за допомогою набору протоколів TCP/IP [19].

Нижче зображена діаграма розгортання, яка відображає чотири обчислювальні вузли для функціонування інформаційної системи: сервер застосунку, браузерний клієнтський застосунок, мобільний клієнтський застосунок, підключені до сервера застосунків, який в свою чергу підключений до серверу бази даних.

2.3. Дизайн та вміст сторінок веб-сайту та екранів мобільного додатку

Інтерфейс користувача – засіб взаємодії користувача з інформаційною системою. Графічний інтерфейс – один з різновидів інтерфейсів, притаманною ознакою якого є можливість взаємодії з електронними пристроями з допомогою графічних зображень та візуальних вказівок.

2.3.1. *Підхід до розробки дизайну*

Дизайн користувацького інтерфейсу є фактором, який визначає три основних показника якості програмного засобу: його функціональність, естетику та продуктивність [20].

Функціональність – той фактор, який грає для розробників провідну роль, адже вони намагаються створити програму таким чином, щоб користувачі могли виконати свої завдання і їм це було зручно зробити. Безперечно, функціональність – дуже важливий аспект, втім не єдиний.

Естетичність зовнішнього вигляду застосунку і способу його представлення дозволяє користувачу сформувати позитивну думку про використання застосунку. Однак естетичні характеристики вельми суб’єктивні і описати їх здебільшого важче, ніж функціональні вимоги або показники продуктивності. Естетика переважно визначається наступними факторами:

- приємна комбінація кольорів;
- відтворення правильного уявлення про мету елементу інтерфейсу за допомогою його зовнішнього вигляду;
- відчуття користувача під час користування тими чи іншими елементами інтерфейсу і їх успішне застосування в контексті поставленого завдання.

Продуктивність, або надійність, також впливають на перспективу використання застосунку. Якщо застосунок має приємний зовнішній вигляд, простий та зручний спосіб керування, однак повільно промальовує екрани, регулярно припиняє свою роботу на десять чи більше секунд, або зовсім припиняє свою роботу і завершує її критичною помилкою, після певних дій користувача, очевидно, що він навряд зможе претендувати на довготривале використання. З іншого боку, швидка та стабільна робота застосунку може компенсувати не найкращий зовнішній вигляд або відсутність другорядної функціональності.

Отже, забезпечити успішну роботи користувача можна з допомогою дизайну інтерфейсу застосунку, який дотримує баланс вищезгаданих факторів. Цього можна досягти послідовним та ретельним проектуванням, яке містить такі етапи:

1) дослідження;
2) користувацькі сценарії;
3) структура інтерфейсу;
4) прототипування інтерфейсу;
5) визначення стилістики;

6) дизайн концепція;

7) оформлення всіх екранів.

2.3.2. *Розробка дизайну інформаційної системи*

Етап 1: на цьому етапі дослідження була зібрана інформація про інтерфейси аналогів та визначено, що необхідно розробити інтерфейси для веб-сайту та мобільного додатку

Етап 2: були розроблені користувацькі сценарії, наведені у табл. 6 та табл. 7.

Таблиця 6

Користувацькі сценарії для веб-сайту

№	Кроки сценарію
1.	Натиснути кнопку «Вхід» → ввести дані в форму → перейти на сторінку користувача
2.	Натиснути кнопку «Вхід» → ввести дані в форму → отримати повідомлення про помилкові дані → ввести правильні дані → перейти на сторінку користувача
3.	Натиснути кнопку «Вхід» → натиснути кнопку «Зареєструватися» → ввести дані в форму → перейти на сторінку користувача
4.	Натиснути кнопку «Вхід» → натиснути кнопку «Зареєструватися» → ввести дані в форму → отримати повідомлення про помилкові дані → ввести правильні дані → перейти на сторінку користувача
5.	Натиснути кнопку «Почати» → перейти на сторінку «Продукти»
6.	Натиснути кнопку «Продукти» → перейти на сторінку «Продукти»
7.	Натиснути кнопку «Меню» → натиснути кнопку «Компоненти» → перейти на сторінку «Компоненти»
8.	Натиснути кнопку «Меню» → натиснути кнопку «Історія» → перейти на сторінку «Історія»
9.	Перейти на сторінку «Продукти» → натиснути на продукт → отримати інформацію про продукт

Продовження табл. 6

№	Кроки сценарію
10.	Перейти на сторінку «Продукти» → ввести назву продукту, який треба знайти → отримати відфільтрований перелік продуктів
11.	Перейти на сторінку «Продукти» → натиснути кнопку «Додати продукт» → ввести дані → натиснути кнопку «Додати»
12.	Перейти на сторінку «Продукти» → натиснути кнопку «Видалити»
13.	Перейти на сторінку «Продукти» → натиснути кнопку «Оновити» → ввести дані → натиснути кнопку «Оновити»
14.	Перейти на сторінку користувача → натиснути кнопку «Заборонені компоненти» → натиснути на «Компонент» → натиснути кнопку «Додати/Видалити»
15.	Перейти на сторінку користувача → натиснути кнопку «Дієта» → натиснути на «Дієту»
16.	Перейти на сторінку користувача → навести на «Продукт» → отримати повідомлення
17.	Перейти на сторінку користувача → натиснути на «Продукт» → отримати інформацію про продукт
18.	Перейти на сторінку «Компоненти» → переглянути інформацію про компоненти
19.	Перейти на сторінку «Компоненти» → натиснути кнопку «Додати компонент» → ввести дані → натиснути кнопку «Додати»
20.	Перейти на сторінку «Компоненти» → натиснути кнопку «Оновити» → ввести дані → натиснути кнопку «Оновити»
21.	Перейти на сторінку «Компоненти» → натиснути кнопку «Видалити»
22.	Перейти на сторінку «Історія» → переглянути продукти
23.	Перейти на сторінку «Історія» → натиснути кнопку «Проаналізувати та спрогнозувати» → переглянути аналіз та прогноз

Продовження табл. 6

№	Кроки сценарію
24.	Натиснути кнопку «Вихід» → перейти на головну сторінку

Таблиця 7

Користувацькі сценарії для мобільного застосунку

№	Кроки сценарію
1.	Натиснути кнопку «Вхід» → ввести дані в форму → перейти на сторінку користувача
2.	Натиснути кнопку «Вхід» → ввести дані в форму → отримати повідомлення про помилкові дані → ввести правильні дані → перейти на сторінку користувача
3.	Перейти на сторінку «Останні переглянуті продукти» → натиснути кнопку «Склад» → переглянути інформацію про продукт → натиснути кнопку «Назад» → перейти на сторінку «Останні переглянуті продукти»
4.	Перейти на сторінку «Продукти» → натиснути кнопку «Склад» → переглянути інформацію про продукт
5.	Перейти на сторінку «Компоненти» → натиснути кнопку «Більше інформації» → переглянути інформацію про компонент → натиснути кнопку «Видалити із заборонених/Додати у заборонені» → натиснути кнопку «Назад» → перейти на сторінку «Компоненти»
6.	Перейти на сторінку «Відсканувати» → відсканувати штрих-код → отримати інформацію про продукт → натиснути кнопку «Назад» → перейти на сторінку «Відсканувати»
7.	Перейти на сторінку «Відсканувати» → відсканувати штрих-код → отримати повідомлення → натиснути на екран → отримати інформацію про продукт → натиснути кнопку «Назад» → перейти на сторінку «Відсканувати»
8.	Перейти на сторінку «Відсканувати» → натиснути кнопку «Повернутися» → перейти на сторінку «Відсканувати»

Етап 3: На цьому етапі була розроблена структура користувацького інтерфейсу.

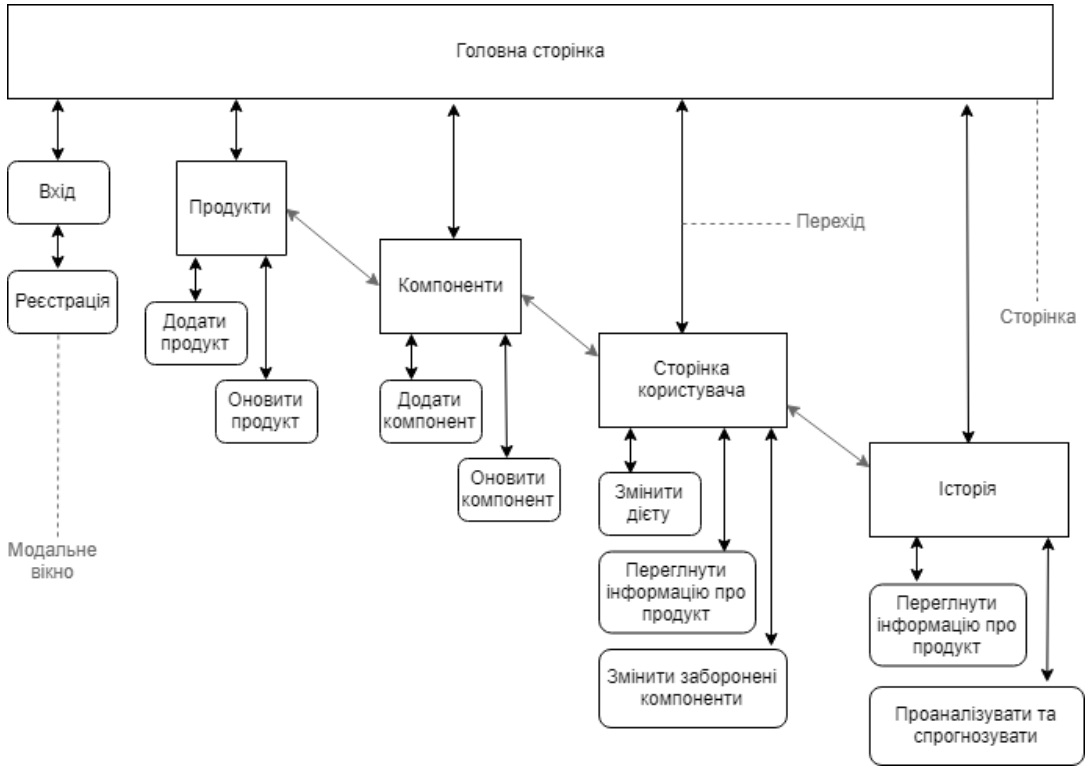


Рис. 3. Схема структури інтерфейсу веб-сайту

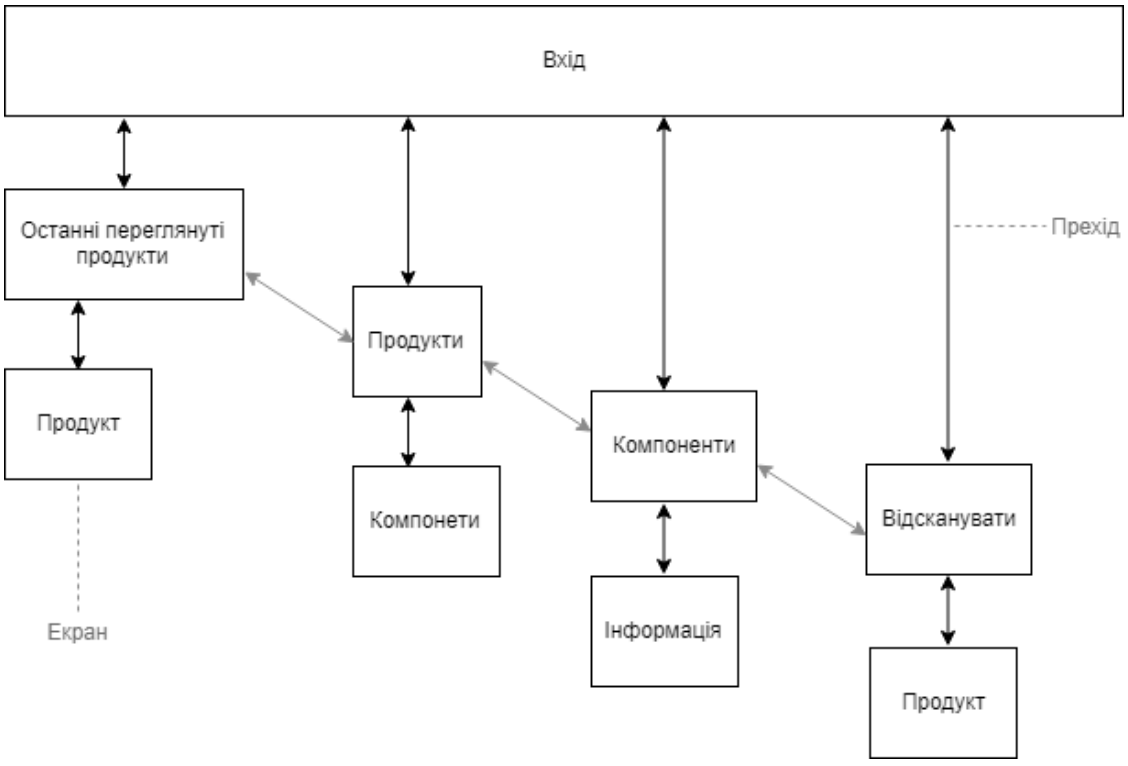


Рис. 4. Схема структури інтерфейсу мобільного застосунку

Етап 4: Прототипи сторінок

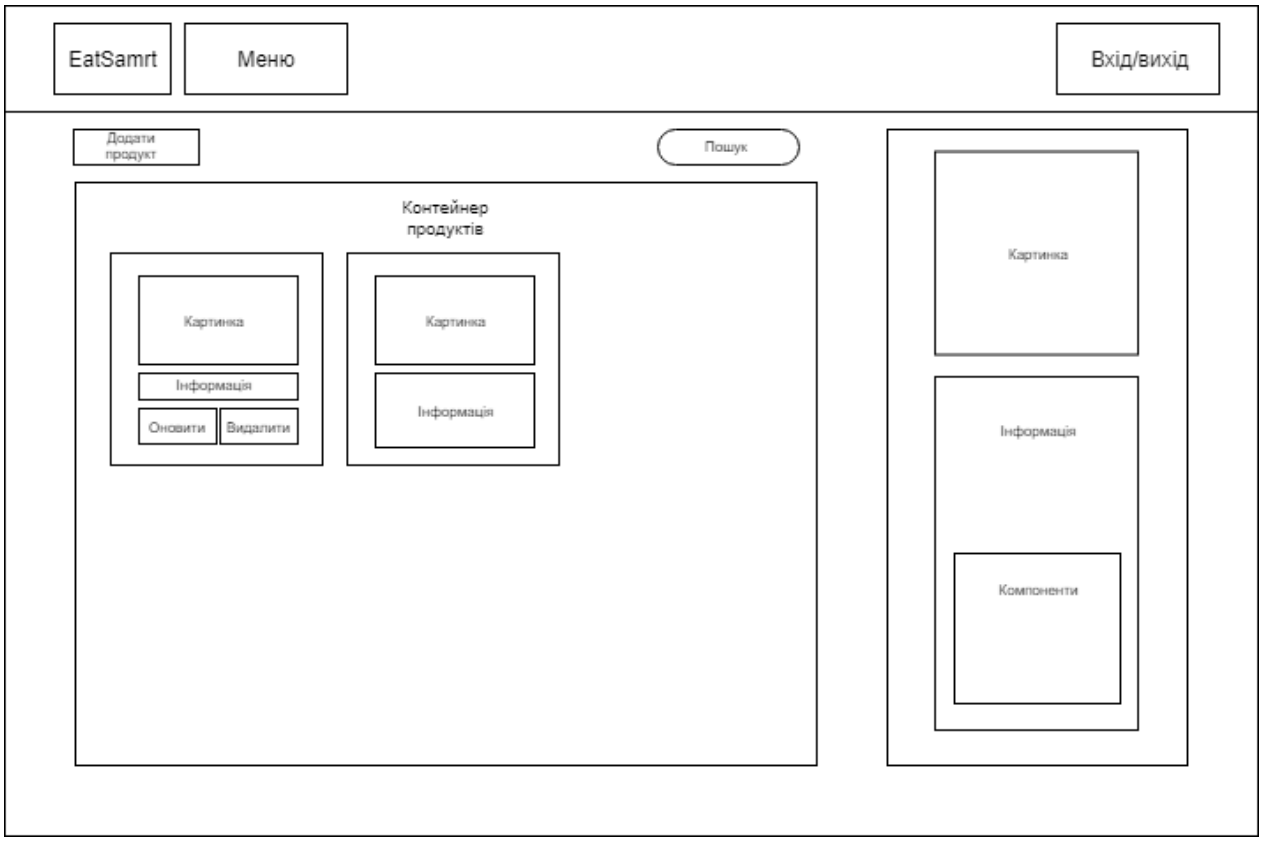


Рис. 5. Прототип сторінки «Продукти» веб-сайту

Етап 5: Стилістика була визначена бібліотекою з готовими елементами користувацького інтерфейсу: для веб-сайту – Semantic UI React, для мобільного застосунку – React Native Elements.

Етап 6: Дизайн концепція полягає у наступних принципах створення дизайну користувацького інтерфейсу:

1. Простий, очевидний, інтуїтивно зрозумілий.
2. Уникнення складної логіки виконання функцій.
3. Тільки справді необхідні елементи інтерфейсу без нагромадження.
4. Зрозумілі і окремі сигнали для різного роду підказок та попереджень.
5. Не більше 5-7 елементів у кожному функціональному блоці.
6. Поділ інформації на сторінках на логічні блоки.

7. Все найважливіше завжди у швидкому доступі.
8. Не більше трьох кроків для переходу між різними розділами програми.
9. Перевірка кожного кроку користувача, з метою не допустити хибних результатів, внаслідок помилки через неуважність.
10. Однакова форма відображення однотипної інформації чи надання реалізації однотипних можливостей у всіх блоках програми.
11. Несхожість на інтерфейси аналогів [21].

Етап 7: Оформлення всіх екранів

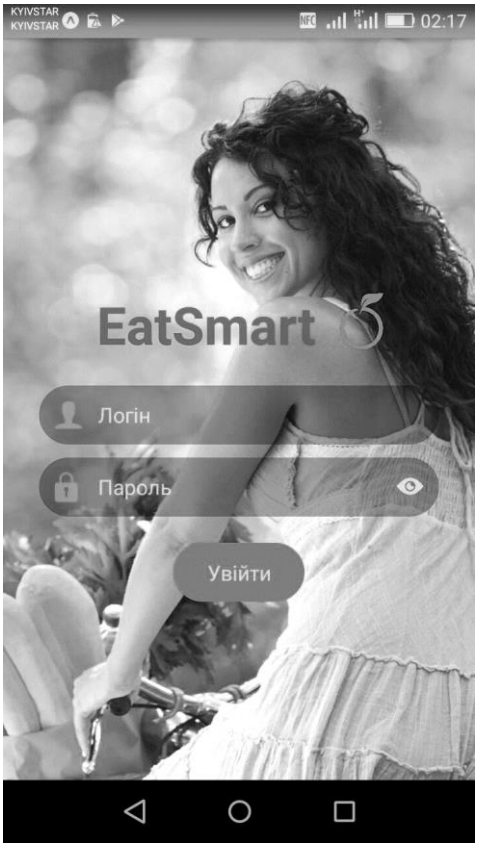


Рис. 6. Екран «Вхід»
мобільного застосунку

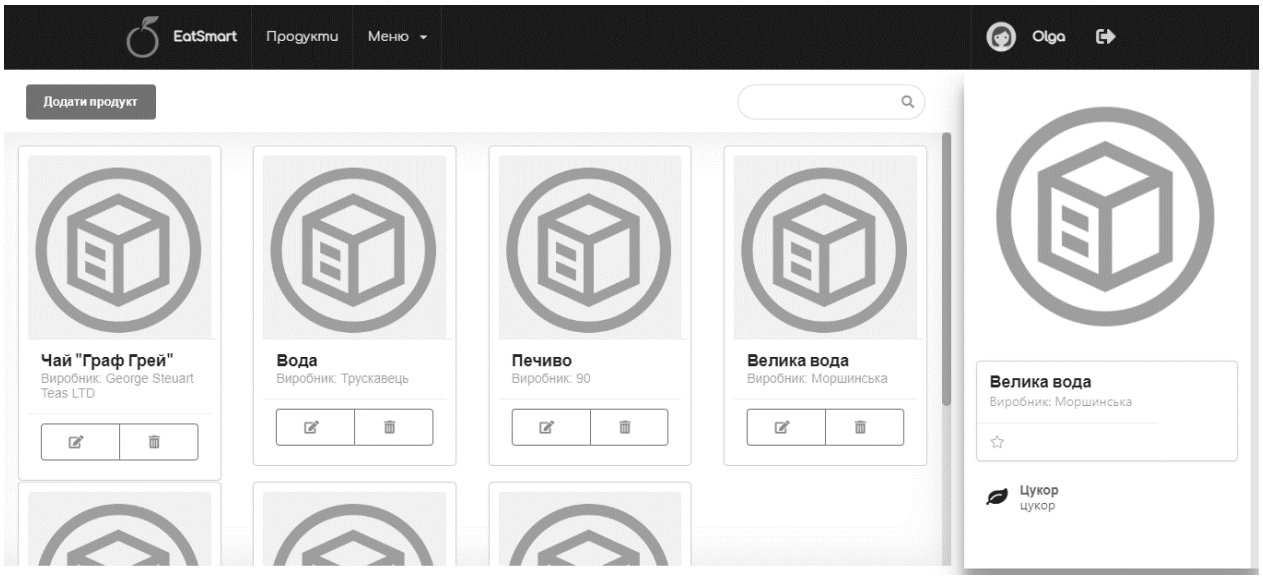


Рис. 7. Сторінка «Продукти» веб-сайту

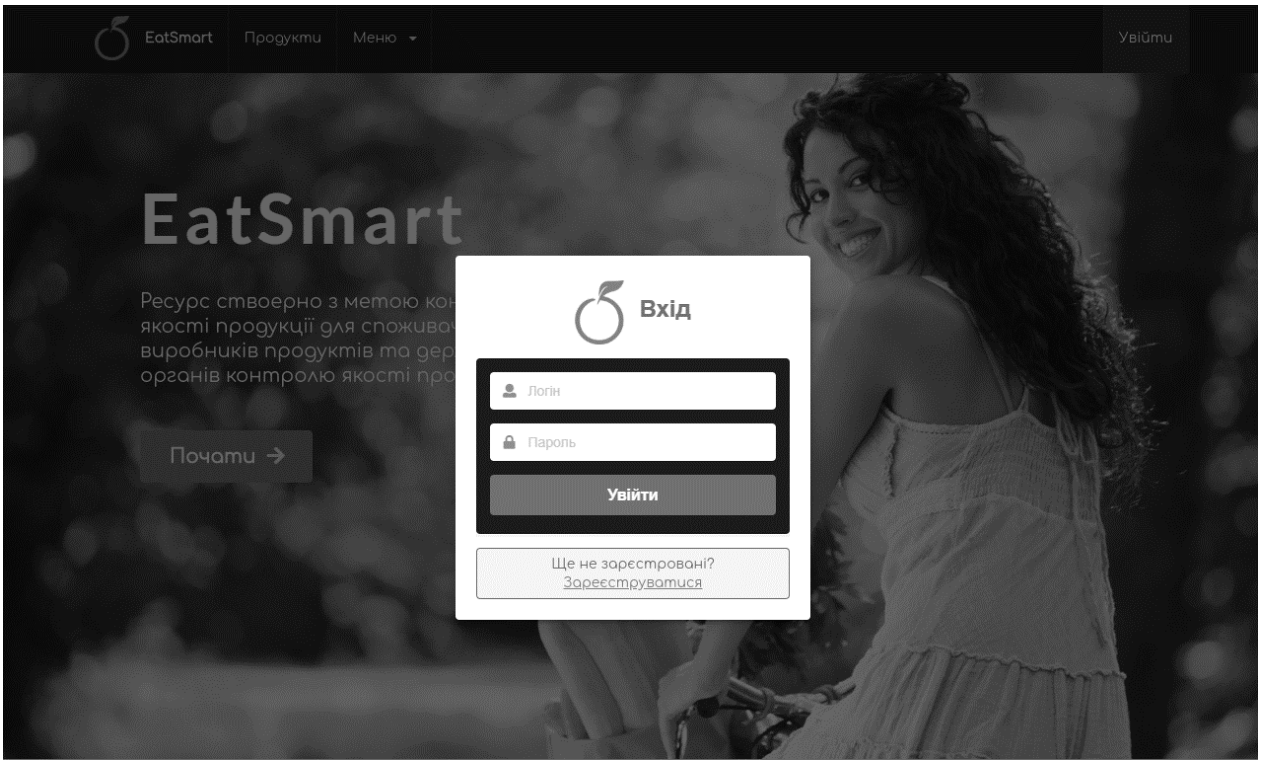


Рис. 8 Сторінка веб-сайту з формою входу

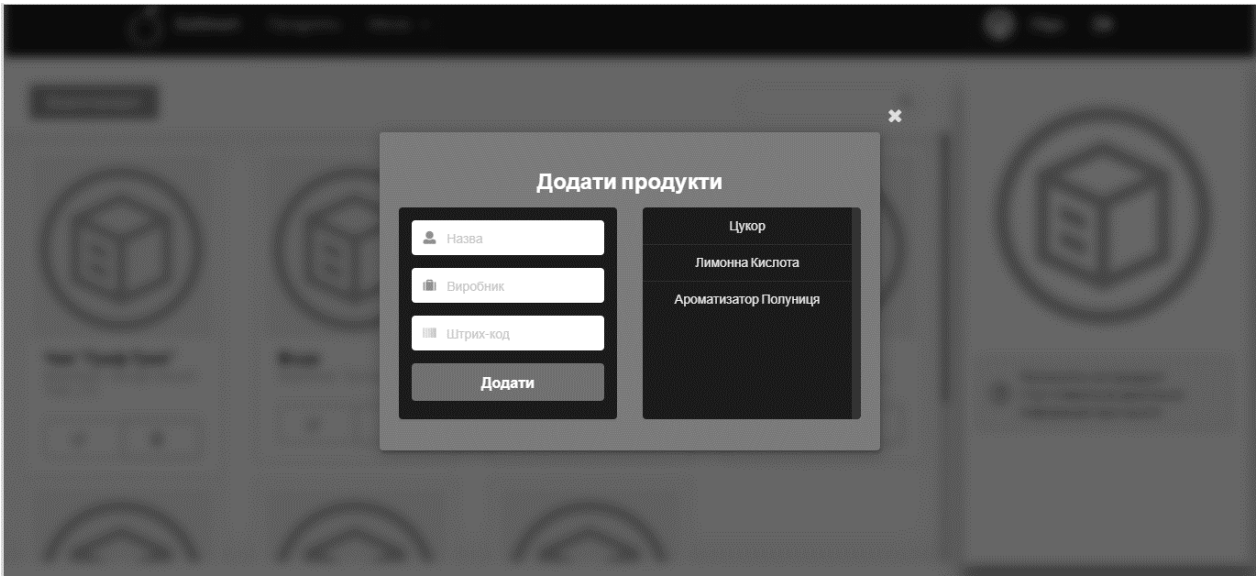


Рис. 9. Сторінка веб-сайту з формою додавання продуктів

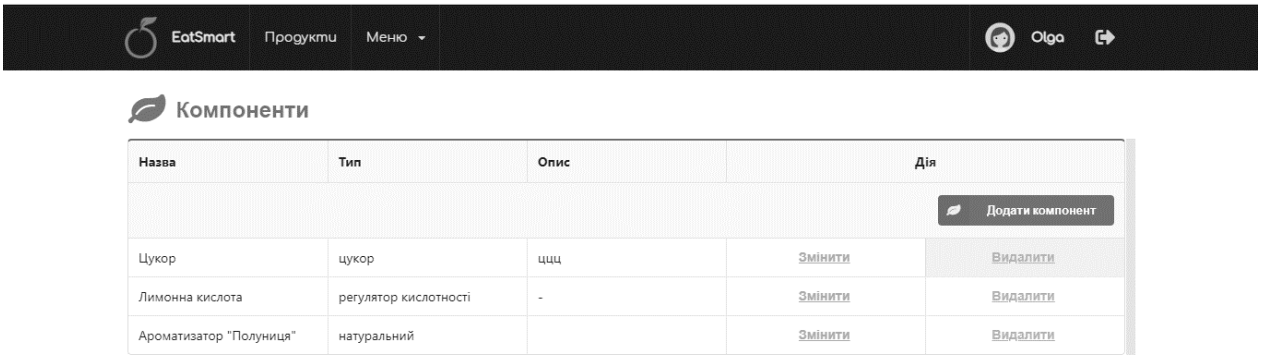


Рис. 10. Сторінка «Компоненти» веб-сайту

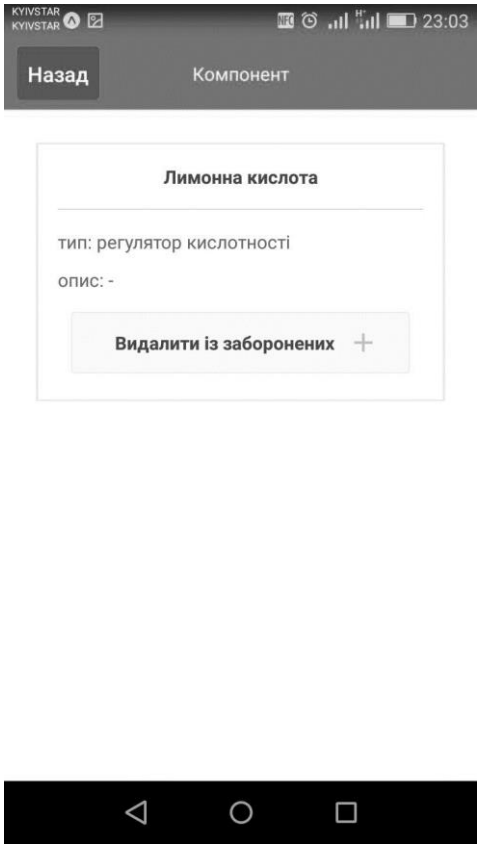


Рис. 11. Екран «Компонент» мобільного застосунку

Рис. 12. Екран «Відсканувати» мобільного застосунку

3. ОПИС РОЗРОБЛЕНИХ ПІДПРОГРАМ ТА АЛГОРИТМІВ

3.1. Серверна частина

3.1.1. *Архітектура серверної частини*

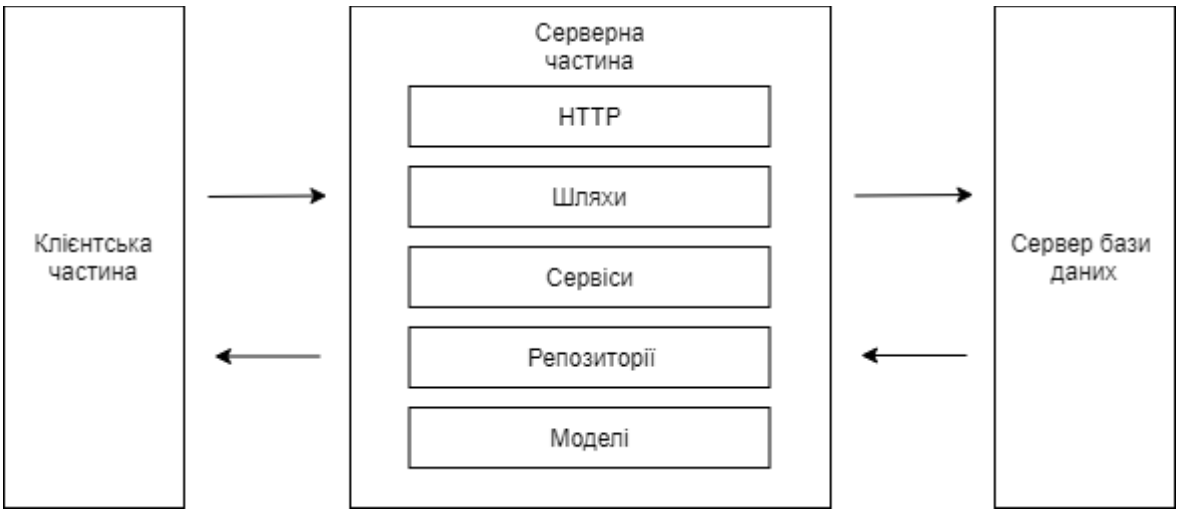


Рис. 13. Архітектура сервера

Розроблена архітектура сервера зображена на рис. 13. Серверна частина містить такі шари:

- HTTP: обробляє запит та створює структуру даних з отриманої із запиту інформації. Запит складається із: рядка запиту, який визначає тип запиту (GET, POST, HEAD тощо), адресу ресурсу, версію протоколу; заголовків запиту, які характеризують тіло повідомлення, параметри передачі та інші відомості; порожнього рядка; тіла повідомлення [22]. Структура запиту зображена на рис. 14.
- Шляхи: визначає спосіб обробки запиту, спираючись на шлях, за яким цей запит надійшов.
- Сервіси: містить всю бізнес-логіку застосунку.
- Репозиторії: містить методи для роботи з об’єктами бази даних.
- Моделі: описує структуру об’єктів бази даних та містить засоби валідації.

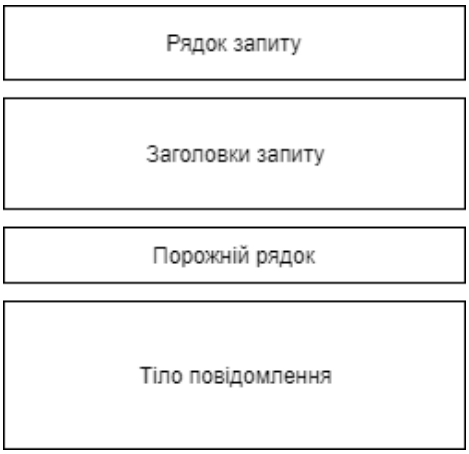


Рис. 14. Структура запиту

.

3.1.2. *Модулі серверної частини*

Серверна частина розроблена з допомогою JavaScript фреймворку Node.js, який використовує модульну систему, тобто вся вбудована функціональність розбита на окремі пакети або модулі. Модуль є блоком коду, який може використовуватися повторно в інших модулях. Розроблений серверний застосунок використовує внутрішні модулі, тобто розроблені мною та зовнішні модулі, які є у відкритому доступі в інтернеті. Серед зовнішніх модулів:

- express – керує маршрутизацією та обробкою запитів [23];
- express-session – модуль для створення сесій. Механізм сесій надає можливість тимчасово зберігати та передавати інформацію між скриптами в межах одного сайту;
- path – модуль для роботи з шляхами до файлів та директорій,
- body-parser – розбиває на логічні частин тіло вхідного запиту та поміщає цю структуру даних у змінну req.body;
- cors – надає можливість отримати дозвіл на використання даних, що знаходяться на інших доменах;
- connect-mongo – модуль MongoDB для створення сесій;
- mongoose – модуль для моделювання об’єктів MongoDB [24].

До внутрішніх модулів належать:

- dbconnect – керує підключенням до сервера бази даних;
- config – визначає конфігурацію підключення до сервера бази даних;
- component – модуль, який описує схему об’єкта MongoDB «Компонент»;
- diet – модуль, який описує схему об’єкта MongoDB «Дієта»;
- product – модуль, який описує схему об’єкта MongoDB «Продукт»;
- user – модуль, який описує схему об’єкта MongoDB «Користувач»;
- componentRepository – модуль, який містить засоби для роботи з об’єктом MongoDB «Компонент»;
- dietRepository – модуль, який містить функції для роботи з об’єктом MongoDB «Дієта»;
- productRepository – модуль, який містить функції для роботи з об’єктом MongoDB «Продукт»;
- userRepository – модуль, який містить функції для роботи з об’єктом MongoDB «Користувач»;
- generalRepository – модуль, який містить функції для роботи з об’єктами MongoDB;
- routes – модуль, який визначає спосіб обробки запиту, який прийшов за певним шляхом;
- componentRoutes – модуль, який визначає спосіб обробки запиту, який прийшов за певним шляхом, який стосується об’єктів MongoDB «Компонент»;
- dietRoutes – модуль, який визначає спосіб обробки запиту, який прийшов за певним шляхом, який стосується об’єктів MongoDB «Дієта»;

- productRoutes – модуль, який визначає спосіб обробки запиту, який прийшов за певним шляхом, який стосується об’єктів MongoDB «Продукт»;
- userRoutes – модуль, який визначає спосіб обробки запиту, який прийшов за певним шляхом, який стосується об’єктів MongoDB «Користувач».

3.1.3. *CRUD парадигма*

Розроблений серверний застосунок реалізовує CRUD парадигму, тобто виконує у відношенні кожної моделі чотири функції:

- create – створення,
- read – читання,
- update – оновлення,
- delete – видалення [25].

Таким чином моделі, розроблені у даному застосунку, є повними та готовими до використання. У табл. 8, 9, 10, 11 наведені реалізовані методи CRUD для моделі «Продукт»:

Таблиця 8

Метод «Створення»

Функція	Create
HTTP-метод	POST

Тіло запиту (json)	<pre>{
	"name": "Шоколад Roshen",
	"components": [],
	"rating": 6,
	"producer": "Roshen",
	"barCode": 4356789034122,
	"date": "01 Jan 1970 00:00:00 GMT"
	}</pre>

Шлях	api/product/

Продовження табл. 8

Програмна реалізація	

<pre>router.post('/',(req,res) => {
    let model = new ProductModel(req.body);
    model.save()
        .then(doc => {
            if(!doc || doc.length === 0){
                return res.status(500).send(doc)
            }
            res.status(201).send(doc)
   })
   .catch(err => {
            res.status(500).json(err)
   })
})</pre>	

Результат (json)	<pre>{
	    "components": [],
	    "barCode": 4356789034122,
	    "_id": "5cfa6497c5ee5900046c92b9",
	    "name": "Шоколад",
	    "rating": 6,
	    "date": "1970-01 01T00:00:00.000Z",
	    "__v": 0
	}</pre>

Таблиця 9

Метод «Читання»

Функція	Read
HTTP-метод	GET
Шлях	api/product/barCode?barCode=4356789034122
Програмна реалізація	

<pre>router.get('/barCode', (req, res) => {
    ProductModel.findOne({
        barCode: req.query.barCode
    })</pre>	

Продовження табл. 9

```
        .then(doc => {
            res.json(doc)
        })
        .catch(err => {
         res.status(500).json(err)})
})
```

Результат (json)	<pre>{
	"components": [],
	"barCode": 4356789034122,
	"_id": "5cfa6497c5ee5900046c92b9",
	"name": "Шоколад",
	"rating": 6,
	"date": "1970-01-01T00:00:00.000Z",
	"__v": 0
	}</pre>

Таблиця 10

Метод «Оновлення»

Функція	Update
HTTP-метод	PUT

Тіло запиту (json)	<pre>{
	"name": "Шоколад Roshen",
	"components": [],
	"rating": 9,
	"producer": "Roshen",
	"barCode": 4356789804579,
	"date": "01 Jan 1970 00:00:00 GMT"
	}</pre>

Шлях	api/product/product?name=Шоколад
Програмна реалізація	

```
router.put('/product', (req, res) => {
    ProductModel.findOneAndUpdate({
      name: req.query.name
```

Продовження табл. 10

```
    }, req.body, {
        new: true
    })
        .then(doc => {
            res.json(doc)
        })
        .catch(err => {
            res.status(500).json(err)
        })
} )
```

Результат (json)	{
	"components": [],
	"barCode": 4356789804579,
	"_id": "5cfa6497c5ee5900046c92b9",
	"name": "Шоколад Roshen",
	"rating": 9,
	"date": "1970-01-01T00:00:00.000Z",
	"__v": 0
	}

Таблиця 11

Метод «Видалення»

Функція	Delete
HTTP-метод	DELETE
Шлях	api/product/product?name= Шоколад Roshen
Програмна реалізація	

```
router.delete('/product', (req, res) => {
   ProductModel.findOneAndRemove({
        name: req.query.name
    })
        .then(doc => {
            res.json(doc)
        })
        .catch(err => {
            res.status(500).json(err)
```

Продовження табл. 11

}) })	

Результат (json)	{
	"components": [],
	"barCode": 4356789804579,
	"_id": "5cfa6497c5ee5900046c92b9",
	"name": "Шоколад Roshen",
	"rating": 9,
	"date": "1970-01-01T00:00:00.000Z",
	"__v": 0
	}

3.2. Веб-застосунок та мобільний застосунок

3.2.1. *Архітектура веб-застосунку та мобільного застосунку*

Веб-застосунок та мобільний застосунок розроблені з допомогою технологій React та Redux, які вплинули на підхід до архітектурного проектування. Саме через використання однакового набору технологій ця архітектура є спільною для веб та мобільного застосунків [26].

Основні компоненти архітектури:

- Action: методи, які створюють actions – структури даних, які містять тип та параметри, які було передано у функцію та передають створену структуру в Dispatcher.
- Dispatcher: отримує actions та відправляє її в Store з допомогою зареєстрованих callback-функцій.
- Store: отримує actions та оновлює себе, ініціює подію зміни View.
- View: очікує подію зміни, бере оновлені дані зі Store та передає їх у views, які мають зв’язок зі Store.

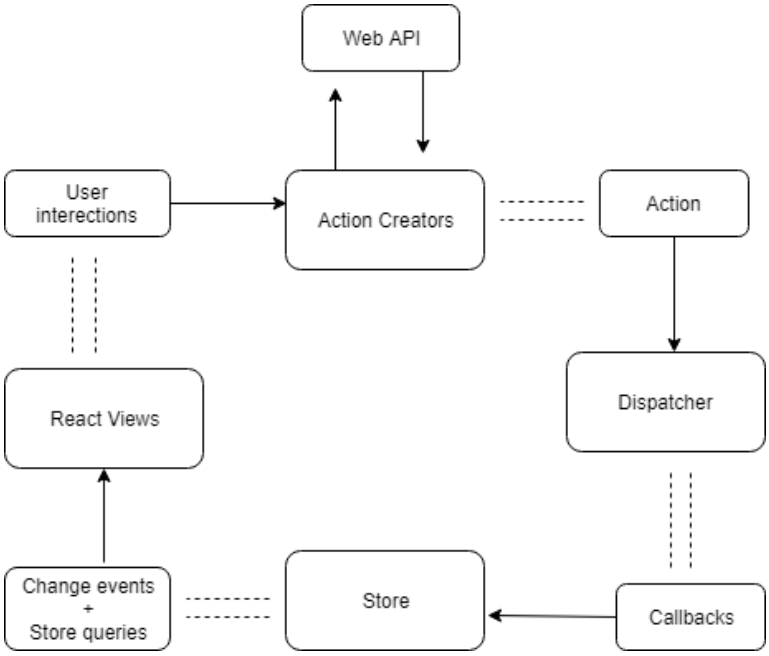


Рис. 15. Архітектура веб-застосунку

3.2.2. *Модулі застосунків*

Модулі веб-застосунку та мобільного застосунку:

- App – кореневий модуль, в якому відбувається навігацію по стрінкам.
  - Pages (для веб-застосунку) та Screens (для мобільного застосунку) – модуль, який містить частини інтерфейсу, які відображені на певній сторінці/екрані.
- Component – статична частина інтерфейсу, яка не має зв’язку зі Store.
- Container – динамічна частина інтерфейсу, яка змінюється, якщо оновлюється Store.
- Analysis – модуль, який реалізовує методи аналізу.
- Logic – модуль, який відповідає за оновлення Store.
  - Actions – модуль, який містить у собі структури даних, які описують, яким чином необхідно оновити Store.

    - Saga – модуль, який містить методи, які використовують зовнішні API та оновлюють Store.
    - Reducer – модуль, який містить функцію, яка приймає Actions та ініціює оновлення Store, відповідно до того, як описано в структурі Action.
    - DefaultState – модуль, який зберігає у собі структур даних Store, у стані за замовчуванням.
- Config – модуль, який містить у собі структуру даних, яка визначає параметри з'єднання із зовнішнім API.
- History – модуль, який містить функцію, яка створює історію переходів між сторінками.
- Store – модуль, який створює Store та підключає до нього модулі Saga та Reducer [27].

3.3. База даних

Проектування бази даних полягало у створені моделей даних. Моделі даних, які були створені:

- user;
- diet;
- product;
- component.

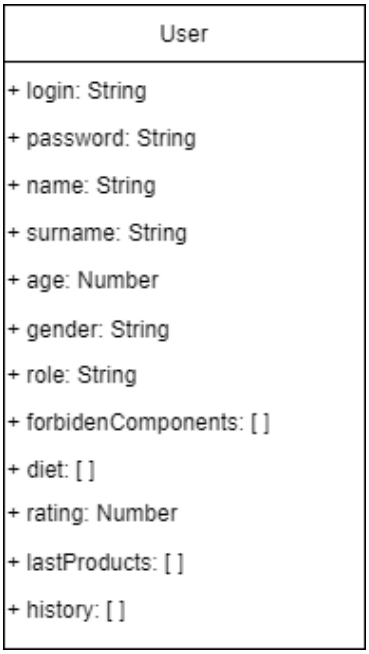


Рис. 16. Модель «Користувач»

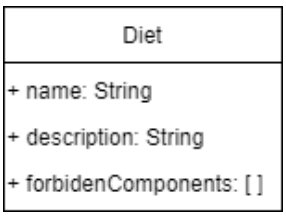


Рис. 17. Модель «Дієта»

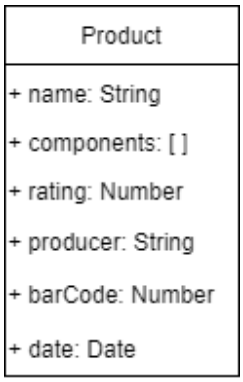


Рис. 18. Модель «Продукт»

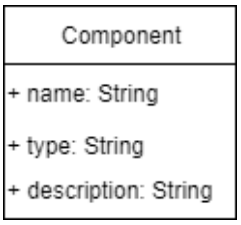


Рис.19. Модель «Компонент»

3.4. Алгоритми аналізу

Для аналізу раціону харчування були обрані такі метрики:

1. мода серед сукупності усіх спожитих продуктів – продукти, які споживаються найбільш часто;
2. входження продуктів, що містять заборонені компоненти, у множину спожитих продуктів;
3. частота вживання цукоровмісних продуктів – кількість цукоровмісних продуктів на одиницю часу, тобто день. Якщо прийняти за дійсне, що добова норма це два цукоровмісні продукти, то можна спрогнозувати наслідки перевищення даної норми.

3.4.1. *Алгоритм знаходження метрики №1*

На рис. 20 зображена схема алгоритму у нотації UML роботи функції, яка знаходить моду серед сукупності спожитих продуктів.

Лістинг 1. Програмна реалізація аналітичного методу для знаходження моди серед сукупності спожитих продуктів.

```
function mostOftenEatenProduct(productsList){
var maxCount = 0;
    for(var y = 0; y < array.length; y++){
        if(array[y].count > maxCount){
            maxCount = array[y].count;}
    }
    var resultProductsArray=[];
    for(var p = 0; p < array.length; p++){
        if(array[p].count == maxCount){
            resultProductsArray.unshift(array[p].product)
        }
    }
    return resultProductsArray;
}
```

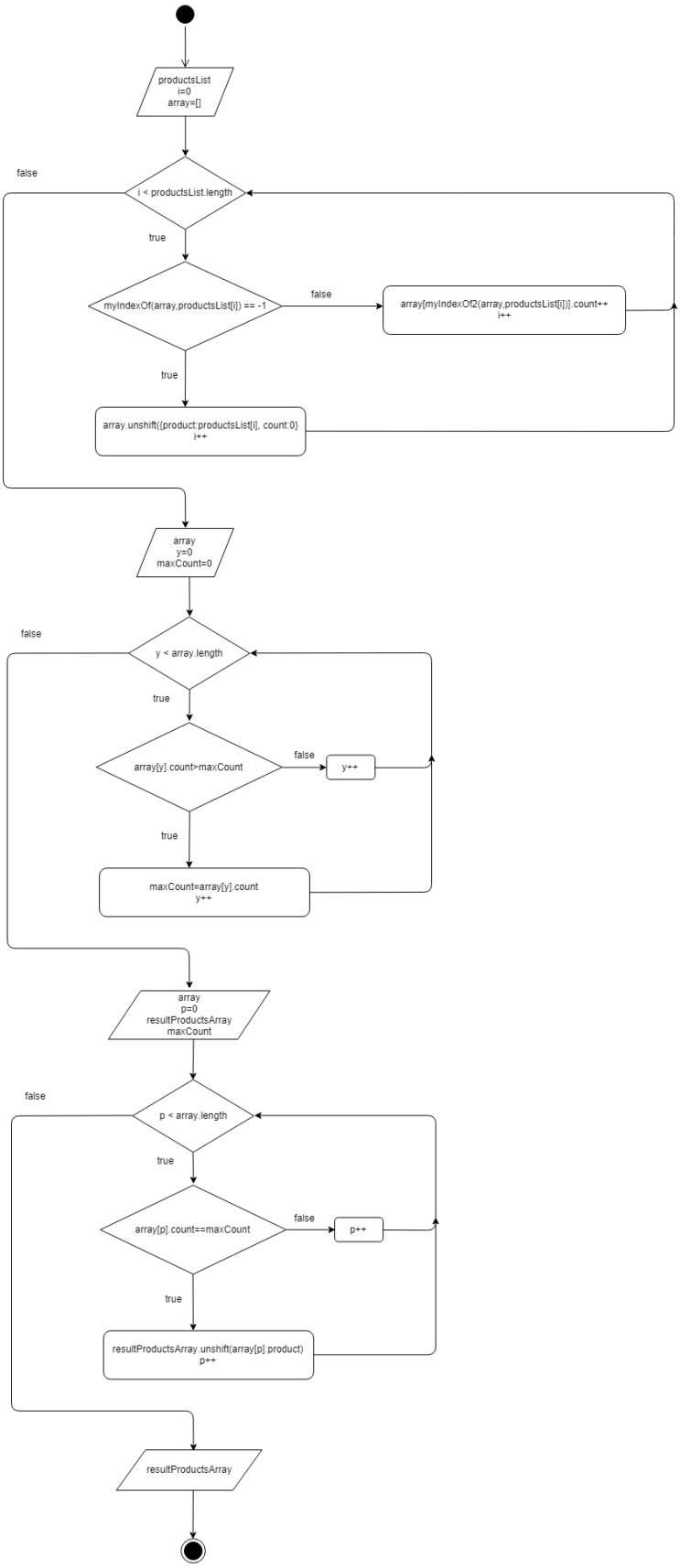


Рис. 20. Алгоритм знаходження моди серед сукупності спожитих продуктів

3.4.2. *Алгоритм знаходження метрики №2*

На рис. 21 зображена схема алгоритму у нотації UML роботи функції для знаходження входження продуктів, що містять заборонені компоненти, у множину спожитих продуктів.

Лістинг 2. Програмна реалізація аналітичного методу для знаходження входження продуктів, що містять заборонені компоненти, у множину спожитих продуктів.

```
function
eatenProductsWithForbidenComponents(productsList,FCList){

 var resultArr = [];
   for(var j = 0; j < productsList.length; j++){
      for(var i = 0; i < FCList.length; i++){
        if(IndexOf(productsList[j].components, FCList[i])!= -1){
           if(IndexOf(resultArr,productsList[j])== -1){
                  resultArr.unshift(productsList[j]);
            }
        }
     }
   }
 return resultArr;
}
```

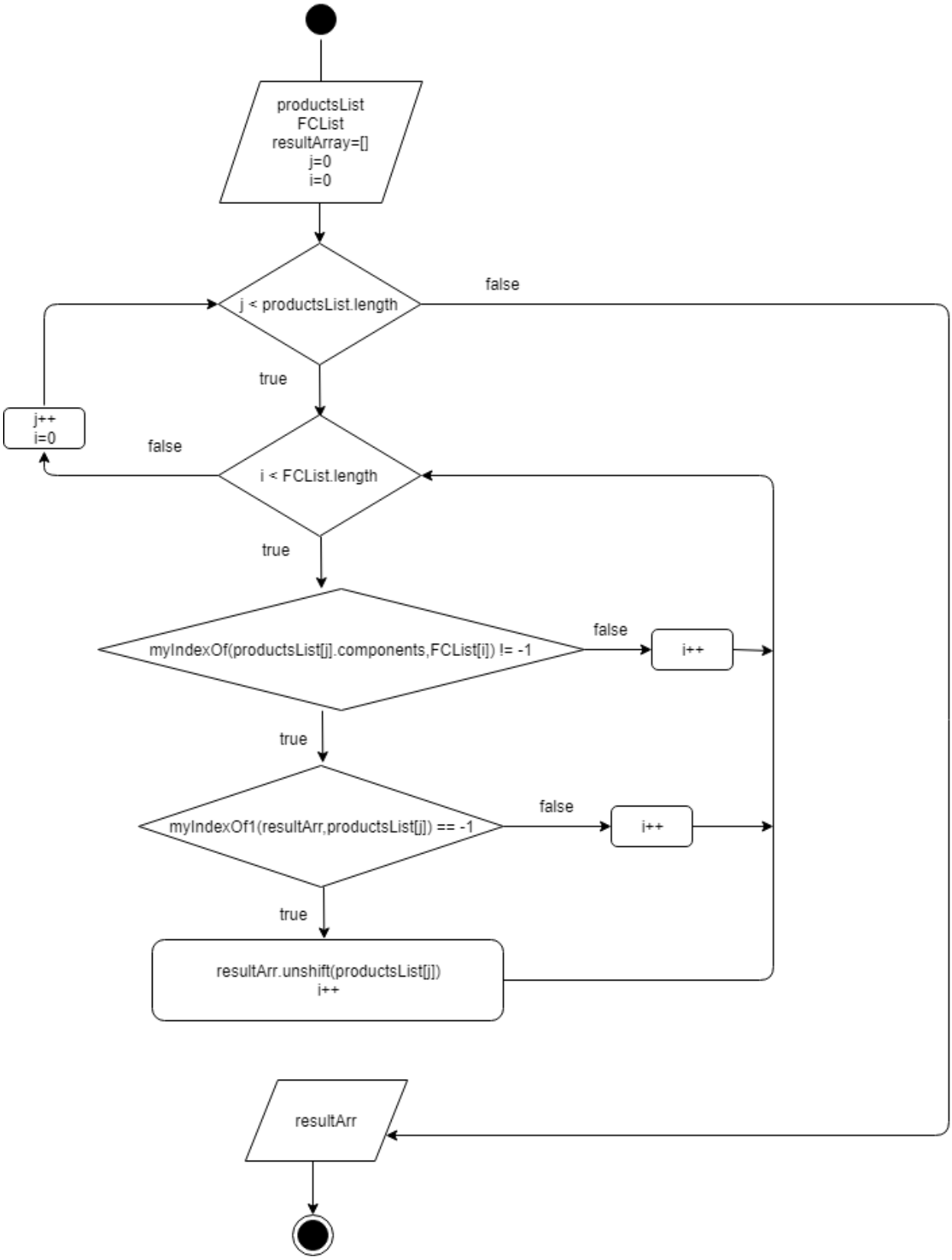


Рис. 21. Алгоритм знаходження моди серед сукупності спожитих продуктів

3.4.3. *Алгоритм знаходження метрики №3*

На рис. 22 зображена схема алгоритму у нотації UML роботи функції.

Лістинг 3. Програмна реалізація аналітичного методу для знаходження частоти вживання цукоровмісних продуктів.

```
function SugarRatio(productsList){
  if(productsList.length > 0){
    var firstTime = moment( productsList[productsList.length-
    1].date);
    var secondTime = moment(productsList[0].date);
    var Days = secondTime.diff(firstTime,'days') + 1;
    var productSugarCount = 0;
    for(var i = 0; i < productsList.length; i++){
        if(IndexOf(productsList[i].components,'Цукор')!= -1){
            productSugarCount++;
        }
    }
    var result = Math.ceil(productSugarCount / Days);
    return result;
    }
    else{
        return 0;
    }
}
```

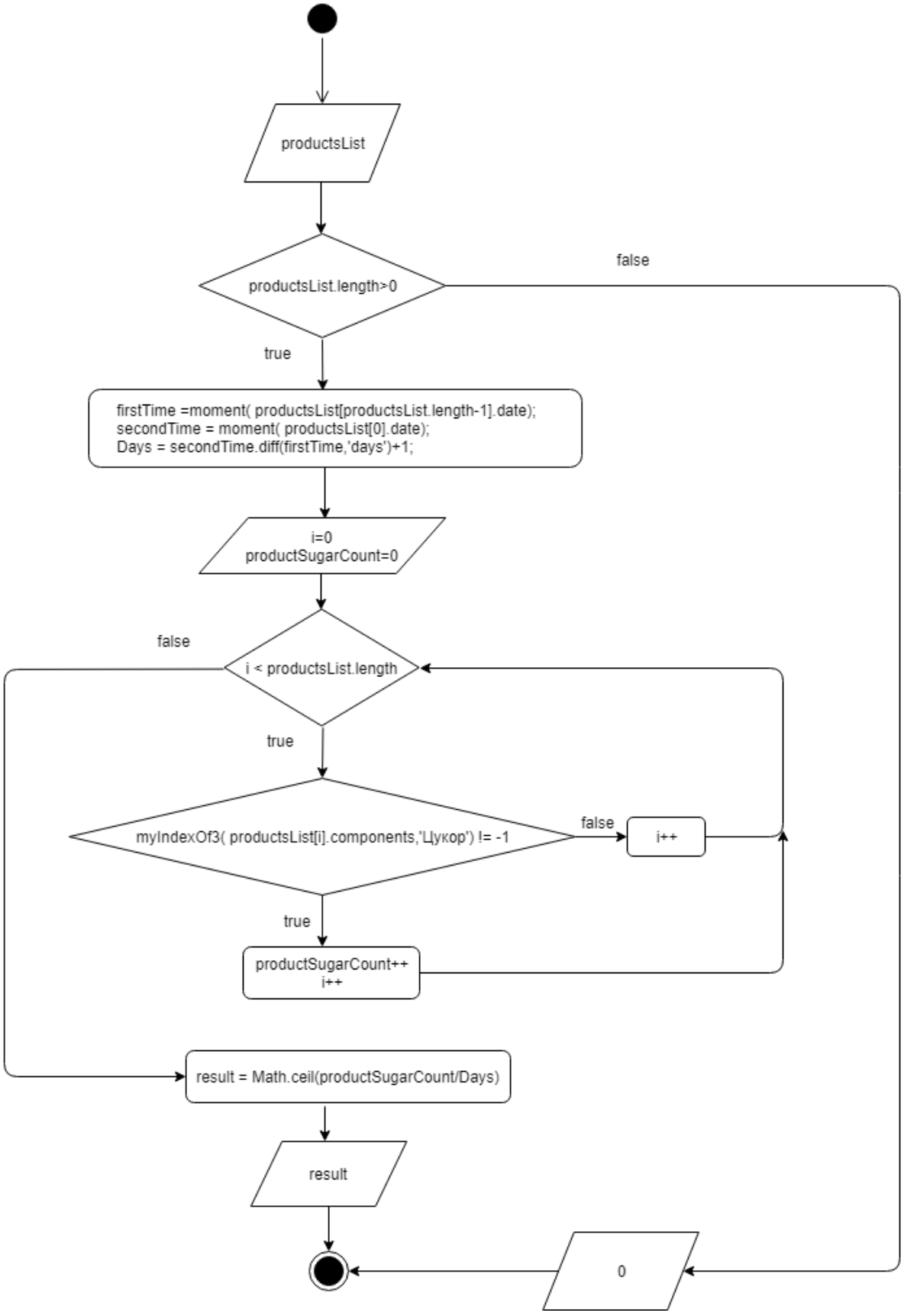


Рис. 22. Алгоритм знаходження частоти вживання цукоровмісних продуктів

4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Тестування системи

Функціональне тестування системи виконувалося з допомогою діаграми переходів станів, яка у зручній формі відображає всі стани, функції системи та реакції на певні дії. Діаграма переходів станів описує складні системи в простій і доступній формі [28].

Елементи діаграми переходів станів:

- Стан (на діаграмі має форму прямокутника із заокругленими кутами). Система знаходиться в певному стані до того, як буде виконано якусь дію.
- Перехід (зображується у формі стрілки). Якась подія провокує зміну стану системи. Перехід на діаграмі зображується у формі стрілки.
- Подія (на діаграмі позначається як коментар над стрілкою переходу). Сама подія відбувається не в системі, це якесь зовнішня дія. Події можуть бути пов'язані або ж відбуватися незалежно один від одного. Це моментальні дії. Вони можуть викликати зміну стану системи, або ж система збереже свій стан без змін.
- Дія (команда, яка слідує за подією). Результатом дії часто є якийсь вихідний результат (щось створюється внаслідок цієї дії).
- Відправна точка, з якої все починається, зображується відповідно як точка. Кінцева точка – як символ позначає потрапляння в яблучко [29].

Тестування системи допомогло виявити нелогічну поведінку, нестачу повідомлень для користувачів, які б дали розуміння, що відбувається у системі, а також несправності. Завдяки тестуванню вдалося виправити несправності та покращити інтерфейс, таким чином, щоб він був зручний та зрозумілий для користувача.

4.1.1. *Тест-кейс «Додавання продуктів»*

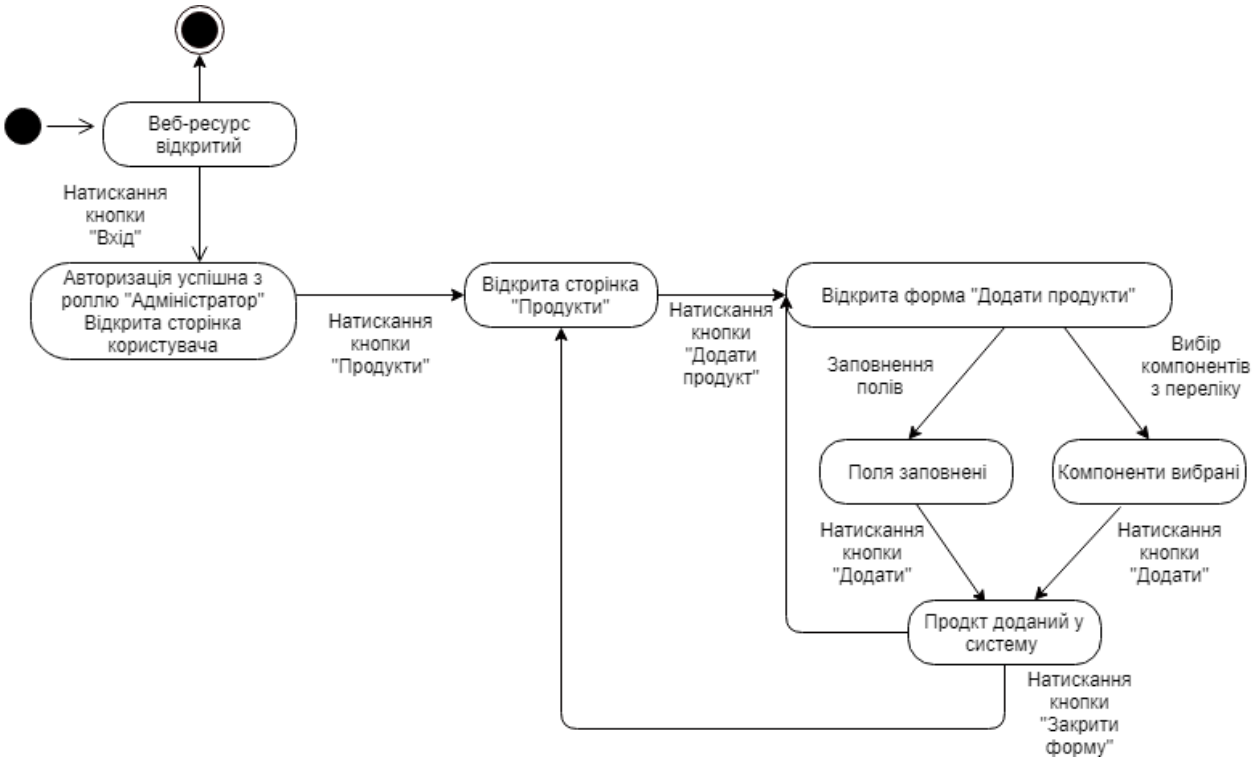


Рис. 23. Діаграма переходів станів для тест-кейсу «Додавання продуктів»

4.2.1. *Тест-кейс «Налаштування заборонених компонентів»*

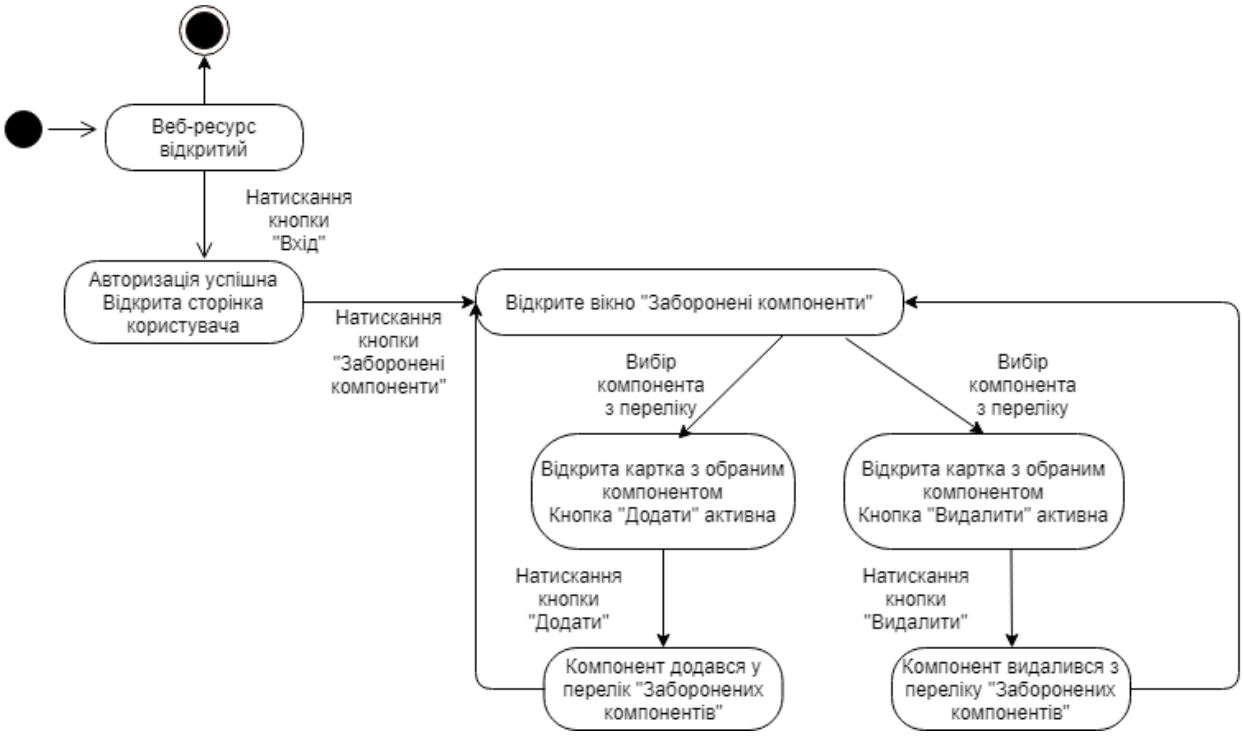


Рис. 24. Діаграма переходів станів для тест-кейсу «Налаштування заборонених компонентів»

4.2. Порівняння системи з існуючими аналогами

Реалізацією подібних технічних засобів займалася не одна команда розробників, тому маємо три аналоги для аналізу, щодо задоволення потреб користувачів. Метою даних додатків є швидка допомога споживачам у виборі якісних продуктів. Аналіз та порівняння існуючих рішень необхідно здійснити за трьома параметрами:

- функціональні можливості – характеристика корисності додатку.
- інтерфейс користувача – характеристика зручності додатку.
- платформа – характеристика доступності додатку.

4.2.1. *Аналог №1: «Е код – харчові добавки»*

Мобільний офлайн додаток, який надає довідник харчових добавок з кодом "Е". Каталог містить опис груп харчових добавок: барвників, консервантів, антиоксидантів, стабілізаторів, емульгаторів, підсилювачів смаку, а також їх шкідливий вплив на здоров'я. Інформацію про добавку можна отримати натиснувши на коді або назві в довіднику, тоді з’явиться повний опис добавки. Окрім того, є три види пошуку: швидкий пошук за номером, пошук за вмістом частини тексту, також можна відкрити опис з історії переглядів [30].

Працює на платформі Android 2.3 і вище. Мова інтерфейсу: російська.

Таблиця 15

Порівняльний аналіз «Е код – харчові добавки» та розробленого застосунку

	Е код – харчові добавки	Розроблений застосунок
Функціональні можливості	• надання інформації про харчові добавки з кодом Е; • функції фільтрації інформації.	• надання інформації про продукт; • надання інформації про компонент;

Продовження табл. 15

	Е код – харчові добавки	Розроблений застосунок
		оновлення інформації про продукт; • оновлення інформації про компонент; • аналіз впливу спожитих продуктів на здоров'я споживача; • налаштування особливостей раціону користувача; • зчитування штрих-коду продукту для отримання інформації; можливість пошуку продукту за назвою.
Інтерфейс користувача	• простий та інтуїтивно зрозумілий; • є підказки для користувача; • мова інтерфейсу російська.	• простий, очевидний, інтуїтивно зрозумілий; • тільки справді необхідні елементи інтерфейсу, без нагромадження; поділ інформації на сторінці на логічні блоки; • все найважливіше завжди у швидкому доступі; • є підказки для користувача;

Продовження табл. 15

	Е код – харчові добавки	Розроблений застосунок
		• однакова форма відображення однотипної інформації чи надання реалізації однотипних можливостей у всіх блоках програми; • мова інтерфейсу українська.
Платформа	Android 2.3 та вище	Android 4.0.1 та вище, веб-ресурс доступний для усіх користувачів мережі інтернет.

Висновок: Розроблений застосунок надає більше корисних для користувача функцій та доступний не тільки для користувачів мобільних телефонів з операційною системою Android, а й для користувачі мережі інтернет, що визначає його більшу доступність. Інтерфейс розробленого застосунку більш сучасний та приємніший для користувача. Мова інтерфейсу українська, тому цей застосунок є зручнішим для україномовної аудиторії.

4.2.2. *Аналог №2: «OutScanner»*

Сканер штрих-кодів продуктів харчування. Це мобільний онлайн додаток, який може надати повний склад продуктів і попередити про шкідливі добавки та інгредієнти. Для визначення складу продукту необхідно скористатися сканером штрих-коду та зчитати код з упаковки. Можливість швидкої оцінки безпеки продукту або порівняння кількох продуктів, «OutScanner» реалізовує за допомогою розрахунку

комплексного індексу небезпеки продукту: чим менше індекс, тим безпечніше продукт.

Найбільш небезпечні добавки та інгредієнти, які необхідно виключити з ужитку, можна винести в спеціальні stop-list. Надана можливість вибору та активації потрібних stop-list, щоб «OutScanner» попередив, якщо сканується продукт, який містить інгредієнти з обраних stop-list. Також можна створити свій власний stop-list.

Додаток містить описи всіх основних інгредієнтів та харчових добавок. Також «OutScanner» зберігає історію сканованих продуктів. Якщо відсканованого продукту не було знайдено, про це додаток одразу повідомить розробника і невдовзі інформація про даний продукт з'явиться у додатку [31].

Працює на платформі iOS 10.3 і вище. Мова інтерфейсу: російська.

Таблиця 16

Порівняльний аналіз «OutScanner» та розробленого застосунку

	OutScanner	Розроблений застосунок
Функціональні можливості	• надання складу продукту; • навігація знаходиться у верхній панелі додатку та завжди доступна, що дозволяє користувачу не тримати у пам'яті свої попередні дії; • пошук у складі заборонених для користувача інгредієнтів та надання відповідних повідомлень; • надання інформації про шкідливі складники;	• надання інформації про продукт; • оновлення інформації про продукт; • надання інформації про компонент; • оновлення інформації про компонент; • можливість пошуку продукту за назвою; • аналіз впливу спожитих продуктів на здоров'я споживача;

Продовження табл. 16

	OutScanner	Розроблений застосунок
	• збереження зчитаних складів; • порівняння складів кількох продуктів; • можливість зчитування штрих-коду з допомогою камери мобільного телефону для отримання інформації, щодо складу продукту; • можливість створення користувацького стоп-листа компонентів.	• налаштування особливостей раціону користувача; • зчитування штрих-коду продукту для отримання інформації.
Інтерфейс користувача	• простий та інтуїтивно зрозумілий; • кольори, які використані для позначення повідомлень про шкоду та користь, однозначно зрозумілі; • мова інтерфейсу російська.	• простий, очевидний, інтуїтивно зрозумілий; тільки справді необхідні елементи інтерфейсу, все найважливіше завжди у швидкому доступі; • без нагромадження; поділ інформації на сторінці на логічні блоки; • однакова форма відображення однотипної інформації чи надання реалізації однотипних можливостей у всіх блоках програми; • є підказки для користувача; • мова інтерфейсу

Продовження табл. 16

		українська.
Платформа	iOS 10.3 та вище	Android 4.0.1 та вище, веб-ресурс доступний для усіх користувачів мережі інтернет.

Висновок: Розроблений застосунок надає однакові можливості для користувача, що й розглянутий аналог, але є доступний для користувачів мобільних телефонів з операційною системою Android, а також для користувачі мережі інтернет, що визначає його більшу доступність. Інтерфейси двох застосунків досить зручні та приємні для користувача. Мова інтерфейсу українська, тому цей застосунок є зручнішим для україномовної аудиторії.

4.2.3. *Аналог №3 «PINTO»*

Веб-ресурс та мобільний додаток, які допомагають дотримуватися індивідуального здорового способу харчування. Pinto дозволяє відстежувати та візуалізувати харчування, отримувати індивідуальні підказки і оцінку якості споживаних продуктів. Також додаток може порадити нові продукти, які можуть сподобатися користувачу та стануть більш якісною альтернативою.

Ця система містить вичерпну інформацію про різноманітні дієти: кето, веганську, безглютенову, діабетичну тощо, а також засоби, які допоможуть користувачу з її дотриманням. Користувач може визначити індивідуальні особливості раціону, які також будуть враховані системою, щоб надати персоналізовані поради.

База продуктів містить більше 100 000 одиниць, їх склад та оцінку якості. Цю інформацію можна отримати або на сайті у відповідному розділі, або з допомогою мобільного додатку, відсканувавши штрих-код продукту.

«PINTO» надає можливість відстежувати харчування: зберігати в архів фотографії порцій та опис всіх продуктів, які користувач споживає кожного прийому їжі. Також враховується вміст білків, жирів та вуглеводів, що дозволяє системі вираховувати відхилення від норми та надавати поради для коригування способу харчування.

Веб-ресурс доступний в мережі інтернет. Мобільний додаток працює на платформі iOS 10.3 і вище. Мова інтерфейсу: англійська [32].

Таблиця 17

Порівняльний аналіз «PINTO» та розробленого застосунку

	PINTO	Розроблений застосунок
Функціональні можливості	• надання складу продукту; надання інформації про шкідливі складники; • пошук у складі заборонених для користувача інгредієнтів та надання відповідних повідомлень; • збереження зчитаних складів; • збереження спожитих продуктів з вказанням кількості жирів, білків та вуглеводів; • надання порад, щодо потенційно цікавих для користувача продуктів. • надання інформації про різноманітні дієти;	• надання інформації про продукт; • надання інформації про компонент; • оновлення інформації про продукт; • оновлення інформації про компонент; • аналіз впливу спожитих продуктів на здоров’я споживача; • налаштування особливостей раціону користувача; зчитування штрих-коду продукту для отримання інформації;

Продовження табл. 17

	PINTO	Розроблений застосунок
	• коригування раціону користувача в залежності від обраної дієти; • можливість зчитування штрих-коду з допомогою камери мобільного телефону для отримання інформації, щодо складу продукту; • можливість створення користувацького стоп-листа компонентів; • можливість фотографувати та зберігати продукти, які вживає користувач для аналізу способу харчування.	• можливість пошуку продукту за назвою.
Інтерфейс користувача	• велике нагромадження інформаційних елементів, що ускладнює розуміння, але інтуїтивність збережена; • є підказки для користувача; • присутня можливість персоналізації інтерфейсу; • навігація завжди доступна, що дозволяє користувачу не тримати у пам’яті свої попередні дії, втім вона інколи змінює розташування та стиль, тому вимагає додаткового часу для пристосування;	• простий, очевидний, інтуїтивно зрозумілий; • тільки справді необхідні елементи інтерфейсу, без нагромадження; • поділ інформації на сторінці на логічні блоки; • все найважливіше завжди у швидкому доступі; • однакова форма відображення однотипної інформації чи

Продовження табл. 17

	• фон під модальним вікном затемнений – користувачу легко зрозуміти, що він має закрити вікно для продовження роботи з додатком; • текстова інформація у більшості випадків ілюструється круговими діаграмами, смугами прогресу та іншими засобами, що також полегшує сприйняття; • Мова інтерфейсу англійська.	надання реалізації однотипних можливостей у всіх блоках програми; • є підказки для користувача; • мова інтерфейсу українська.
Платформа	iOS 10.3 та вище, веб-ресурс доступний для усіх користувачів мережі інтернет.	Android 4.0.1 та вище, веб-ресурс доступний для усіх користувачів мережі інтернет.

Висновок: Розроблений застосунок надає менше можливостей для користувача, але є доступний для користувачів мобільних телефонів з операційною системою Android, а також для користувачі мережі інтернет, що визначає однакову доступність обидвох застосунків. Інтерфейс аналогу має лише один недолік – велике нагромадження елементів, за іншими параметрами він є кращим за інтерфейс розробленого застосунку. Мова інтерфейсу українська, тому цей застосунок є зручнішим для україномовної аудиторії.

4.3. Рекомендації щодо подальшого вдосконалення

Розглянуті аналоги та власне бачення дозволили сформувати концепцію подальшого вдосконалення розробленого програмного засобу, яке полягає у наступних кроках допрацювання:

1. Додати у веб-ресурс наступну функціональність:
    - Переклад інформації різним мовами.
    - Поглиблений аналіз спожитих продуктів.
    - Покращене прогнозування наслідків спожитих продуктів.
    - Можливість додати друга.
    - Можливість читати/додавати/оновлювати статті та новини.
    - Можливість проходити тести.
    - Переглядати рейтинг за кількістю та успішністю пройдених тестів серед користувачів.
    - Збереження історії спожитих продуктів, їх аналізу та прогнозу у форматі pdf.
    - Можливість додавати та оновлювати дієти.
    - Змінити алгоритм додавання продуктів таким чином, щоб одразу можна було вносити нові компоненти у систему.
    - Пошук компонента за назвою чи типом.
    - Можливість додавати рекламу.
    - Надавати статистику сканованих продуктів.
    - Можливість персоналізації.
    - Можливість завантаження зображень продуктів.
2. Додати у мобільний застосунок наступну функціональність:
    - Реєстрація.
    - Налаштування дієти.
    - Поглиблений аналіз спожитих продуктів.
    - Покращене прогнозування наслідків спожитих продуктів.
    - Можливість додати друга.
    - Можливість читати статті та новини.
    - Можливість проходити тести.
    - Переглядати рейтинг за кількістю та успішністю пройдених тестів серед користувачів.

- Збереження історії спожитих продуктів, їх аналізу та прогнозу у форматі pdf.
- Переклад інформації різними мовами.
- Можливість збільшувати розмір тексту.
- Можливість персоналізації.

3. Покращення нефункціональних показників:
   - Допрацювання програмної реалізації мобільного застосунку таким чином, щоб він справно працював на платформі iOS.
   - Більше інтерактивних елементів інтерфейсу та графічного ілюстрування текстової інформації.
   - Реалізація можливості користуватися певними функціями мобільного застосунку в режимі «без мережі».
   - Реалізація механізму захисту особистих даних.

ВИСНОВКИ

Метою даної дипломної роботи було реалізація програмного засобу – інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів.

Аналіз засобів розробки веб-застосунків та мобільних застосунків, попередньо виконаний в дипломному проекті, вказав на доцільність створення інформаційної системи з використанням технологій Node.js, React.js, MongoDB.

Розроблена інформаційна система:

1) надає можливість перегляду інформації про продукт;
2) надає можливість перегляду інформації про компонент;
3) надає можливість оновлення інформації про продукт;
4) надає можливість оновлення інформації про компонент;
5) надає можливість налаштування особливостей раціону користувача;
6) надає можливість аналізу впливу спожитих продуктів на здоров’я споживача.
7) має зрозумілий та зручний інтерфейс.

Розробка виконана у повному обсязі, всі вимоги враховані, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

Використання розробленої інформаційної системи дозволить:

1) допомогти пересічному споживачу аналізувати якість продуктів харчування;
2) допомогти органам якості контролю продуктів харчування контролювати якість продуктів харчування.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Сучасний підручник JavaScript: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://learn.javascript.ru/. – (10.01.2019).
2. Довідник з HTML: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://htmlbook.ru/html. – (30.01.2019).
3. Основи CSS: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://htmlacademy.ru/courses/basic-css. – (12.02.2019).
4. Вікі-сторінка (Node.js): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Node.js. – (12.02.2019).
5. Вікі-сторінка (PHP): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PHP. – (13.02.2019).
6. Основи React.js: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://learn.javascript.ru/screencast/react. – (14.02.2019).
7. Вікі-сторінка (React): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/React. – (20.02.2019).
8. Новий рівень React - Redux: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://getinstance.info/articles/react/learning-react-redux/. – (23.02.2019).
9. Вікі-сторінка (AngularJS): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS. – (24.02.2019).
10. Основи AngularJS: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://learn.javascript.ru/screencast/angular. – (27.02.2019).
11. Build native mobile apps using JavaScript and React: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://facebook.github.io/react-native/. – (01.03.2019).
12. Пишемо перший додаток на React Native: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://medium.com/devschacht/create-devschacht-app-part-1-db9570cf3f9b. – (03.03.2019).

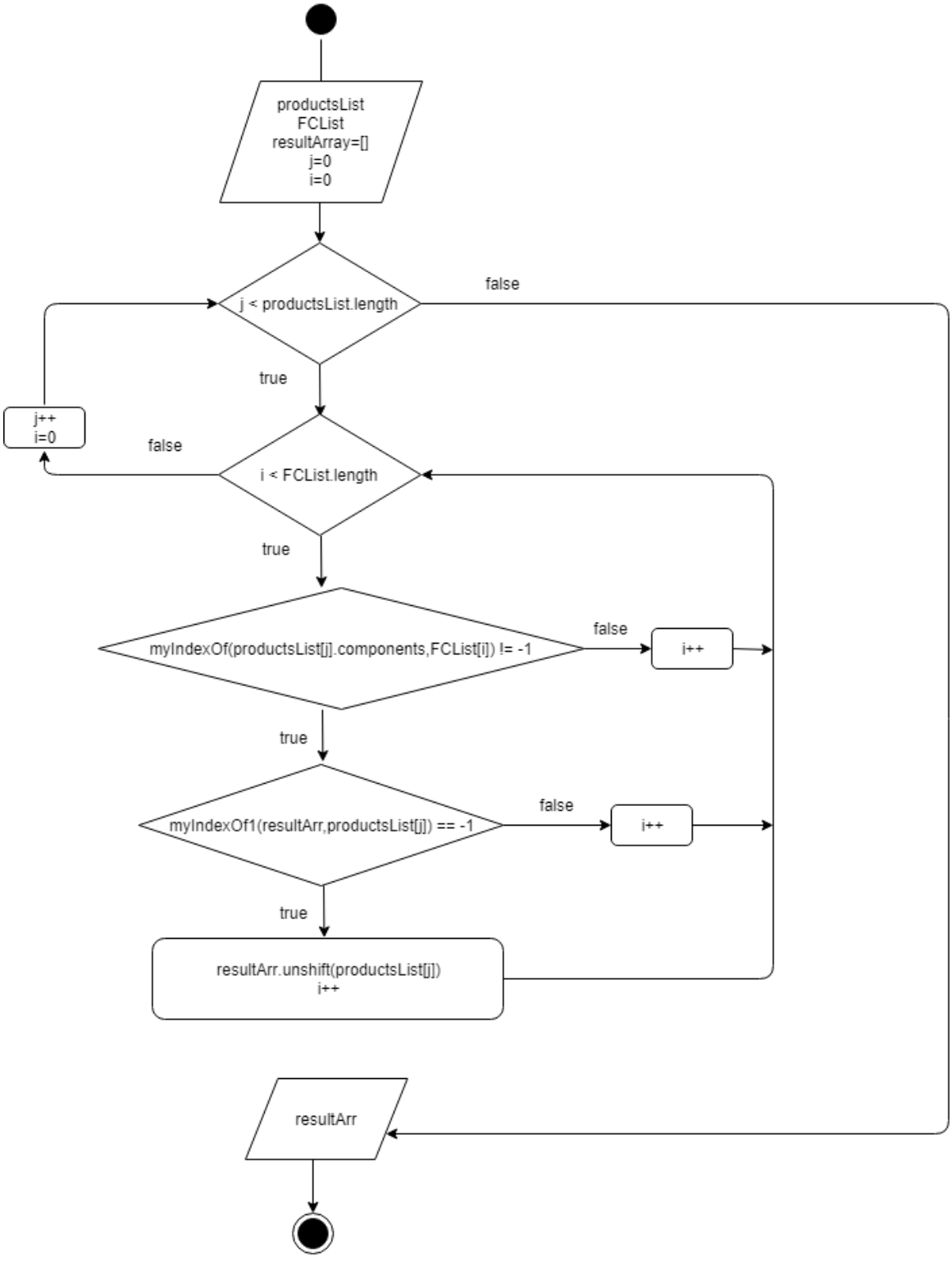
13. React Native — одного JS мало: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/post/323214/. – (07.03.2019).
14. React Native vs. Real Native Apps – Which is Better?: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://hackernoon.com/react-native-vs-real-native-apps-which-is-better-a8383d6f7ca5. – (10.03.2019).
15. Вікі-сторінка (База даних): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85. – (13.03.2019).
16. Вікі-сторінка (MongoDB): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB. – (15.03.2019).
17. Введение в MongoDB: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://mkdev.me/posts/vvedenie-v-mongodb. – (16.03.2019).
18. Вікі-сторінка (Клієнт-серверна архітектура): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0. – (20.03.2019).
19. О модели взаимодействия клиент-сервер простыми словами: [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostymi-slovami-arxitektura-klient-server-s-primerami.html. – (21.03.2019).

20. Вікі-сторінка (Прикладний програмний інтерфейс): [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81. – (30.03.2019).
21. 10 заповідей UI дизайну: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://internetdevels.ua/blog/the-10-commandments-of-user-interface-design. – (04.04.2019).
22. Простым языком об HTTP: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/215117/. – (08.04.2019).
23. Веб-фреймворк для приложений Node.js: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://expressjs.com/ru/. – (09.04.2019).
24. Node.js модули, о которых вы должны знать: express.js: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://nodeguide.ru/doc/modules-you-should-know/express/. – (10.04.2019).
25. Вікі-сторінка (CRUD): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CRUD. – (14.04.2019).
26. Что такое CRUD-приложение?: [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://toster.ru/q/538933. – (17.04.2019).
27. Архитектура модульных React + Redux приложений: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/post/326484/. – (19.04.2019).
28. Вікі-сторінка (Диаграмма состояний): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(UML). – (20.04.2019).

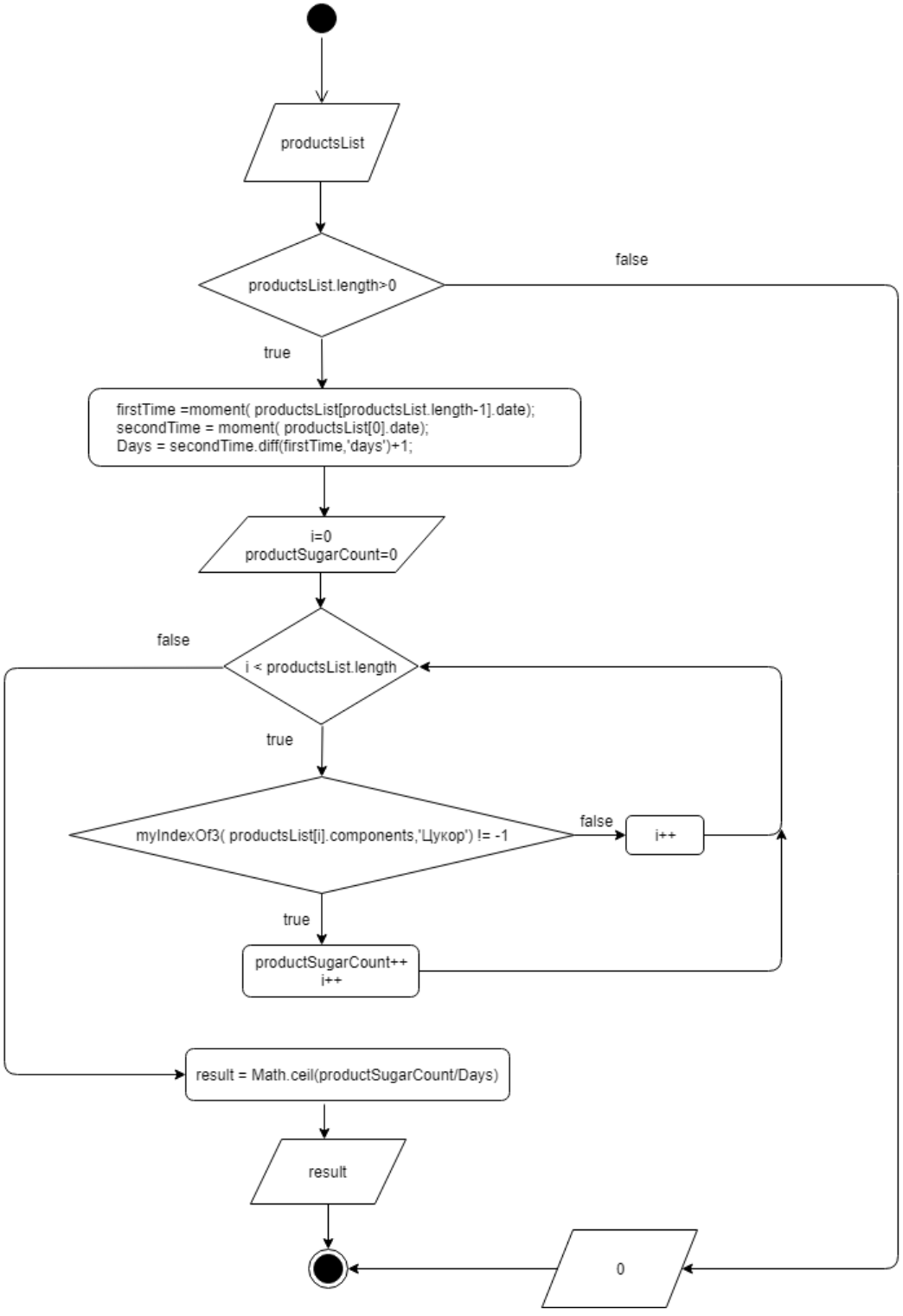
29. Диаграммы переходов состояний: [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://studfiles.net/preview/3828360/page:8/. – (06.05.2019).
30. Е код - пищевые добавки: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=svs.eee&hl=ru. – (14.05.2019).
31. Outscanner: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://itunes.apple.com/us/app/outscanner/id1432672583?mt=8. – (17.05.2019).
32. PINTO: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://pinto.co/. – (19.05.2019).

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

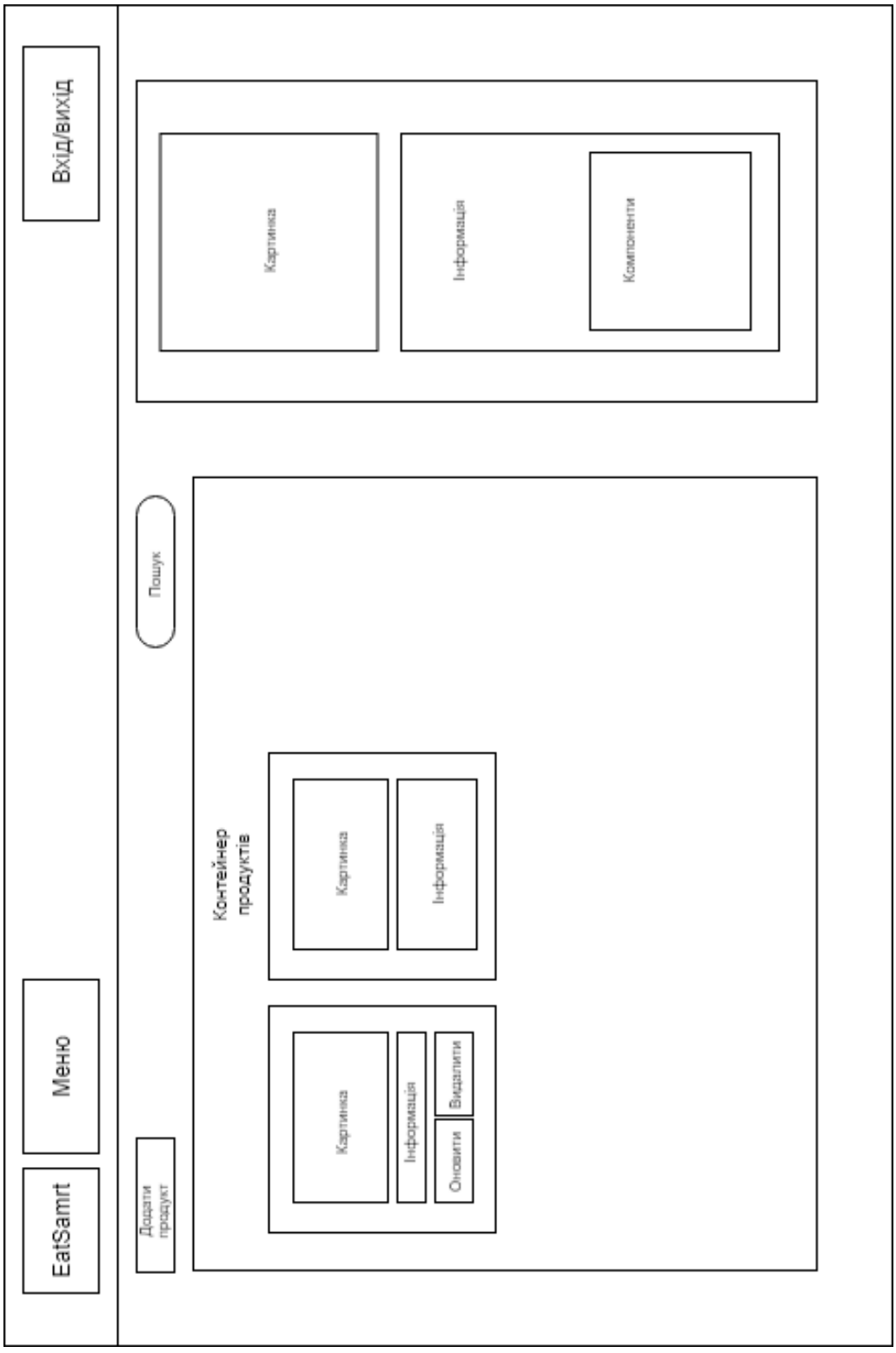


ДП. 045490-06-99
Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів
Пошук серед спожитих продуктів таких, що містять заборонені компоненти. Схема алгоритму

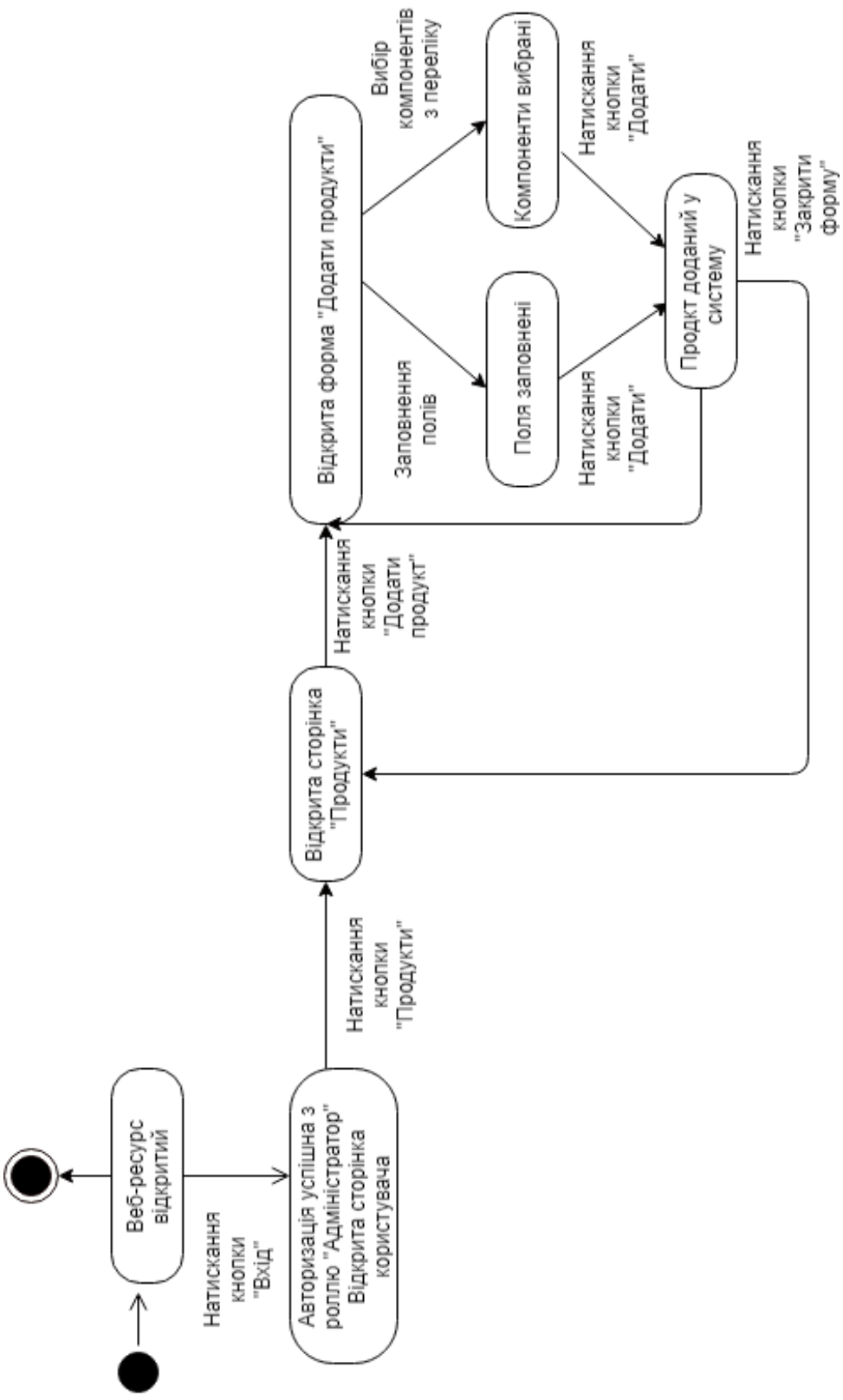


ДП. 045490-07-99
Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів
Обрахунок кількості спожитих цукоровмісних продуктів.
Схема алгоритму

Прототип інтерфейсу сторінки «Продукти» веб-ресурсу



Тест-кейс «Додавання продукту»



Додаток 2
Лістинг програми

Серверний застосунок. Файл index.js

```
const express = require("express");
const session = require("express-session");
const MongoStore = require("connect-mongo")(session);
const mongooseConnection = require("./db/dbconnect").connection;
const path = require("path");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();
app.use(
  session({
    secret: "sessionsecretsessionsecret",
    resave: true,
    saveUninitialized: true,
    store: new MongoStore({
      mongooseConnection: mongooseConnection
    })
  })
);
const staticPath = path.normalize(__dirname + "/public");
app.use(express.static(staticPath));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use((req,res,next)=>{
  console.log(new Date().toString() +' => '+ req.originalUrl);
  next();
})
app.use(bodyParser.json());
app.use(cors());
```

```
const routes = require("./routes/api/routes")(app);

app.use((req,res,next)=>{

    res.status(404).send('We think you are lost!');

})

app.use((err, req,res,next)=>{

  console.error(err.stack);

  res.sendFile(path.join(__dirname,  '/public/500.html'));

})

let port = process.env.PORT;

if (port == null || port == "") {

  port = 80;

}

const server = app.listen(port, ()=>console.info('Server has started on ' +
port));
```

Серверний застосунок. Файл product.js

```
let mongoose = require('mongoose');

let ProductSchema = new mongoose.Schema({

    name: String,

    components: [],

    rating: Number,

    id_producer: String,

    barCode: {type: Number, default: 0},

    date:{type: Date, default: Date.now},

})

module.exports = mongoose.model("Product", ProductSchema);
```

Серверний застосунок. Файл generalRepository.js

```
function Repository() {}

Repository.prototype.getAll = getAll;

Repository.prototype.getById = getById;

function getAll(callback) {

  var model = this.model;

  var query = model.find();

  query.exec(callback);

}

function getById(id, callback) {

  var model = this.model;

  var query = model.findOne({

    _id: id

  });

  query.exec(callback);

}

module.exports = Repository;
```

Веб-клієнт. Файл header_user.js

```
import React, { Component, Fragment } from "react";
import { connect } from "react-redux";
import { mapStateToProps, mapDispatchToProps } from "./container";
import Header from '../../components/header'
import "./index.scss";
import history from "../../history";
export class UserHeader extends Component{
    authorization = (login, password)=>{
        this.props.authorizeUser(login,password);
    }
     componentDidUpdate(prevProps){
        if(this.props.user.login!='999'){
            if(prevProps.user.name!=this.props.user.name){
                this.redirect();
            }
        }
     }
     redirect =()=>{
        history.push("/user");
    }
      registration = (login,password,name,surname,gender,age)=>{
            this.props.registrationUser(login,password,name,surname,gender,
            age,"user",0);
    }
    render(){
        const { error, loading, user } = this.props;
```

```
        return( <Fragment>
                    <Header registration={this.registration} user={user}
                    error={error} loading={loading}
                    authorization={this.authorization} />
                </Fragment>
        )
    }
}
export default connect(
    mapStateToProps,
    mapDispatchToProps
)(UserHeader);
```

Веб-клієнт. Файл logic_user.js

```
import { put, all, takeLatest } from "redux-saga/effects";
import { call } from "redux-saga/effects";
import * as actionTypes from "./actionTypes";
import api from "../../services/api";
function* authorizeUser(action) {
  try {
    const response = yield call(api.sendRequest,
      "api/user?login="+action.payload.login+"&password="+
      action.payload.password, "get");
    yield put({
      type: actionTypes.LOAD_USER_SUCCESS,
      payload: {
        user: response.data
      }
    });
  } catch (err) {
    yield put({
      type: actionTypes.LOAD_USER_ERROR,
      payload: {
        error: err.response.data
      }
    });
  }
}
function* registrationUser(action) {
  console.log(action.payload);
```

```
  try {
    const response = yield call(api.sendRequest, "api/user/",
      "post",action.payload);
    yield put({
      type: actionTypes.ADD_USER_SUCCESS,
      payload: {
        user: response.data
      }
    });
  } catch (err) {
    yield put({
      type: actionTypes.ADD_USER_ERROR,
      payload: {
        error: err.response.data
      }
    });
  }
}
function* updateUser(action) {
  try {
    console.log(action.payload.arr);
    const response = yield call(api.sendRequest,
      "api/user/products?login="+action.payload.arr.login+"&password="+action.
      payload.arr.password, "put", action.payload.arr);
    yield put({
      type: actionTypes.UPDATE_USER_SUCCESS,
      payload: {
        user: response.data
```

```
      }
    });
  } catch (err) {
    yield put({
      type: actionTypes.UPDATE_USER_ERROR,
      payload: {
        error: err.response.data
      }
    });
  }
}
export default function* userSaga() {
  yield all([takeLatest(actionTypes.LOAD_USER,
authorizeUser),takeLatest(actionTypes.UPDATE_USER,
updateUser),takeLatest(actionTypes.ADD_USER, registrationUser)]);
}
```

Мобільний клієнт. Файл settings_screen.js

```
import React, {Component} from 'react';
import {View, Text, StyleSheet, Image} from 'react-native';
import {  Container,Content} from 'native-base';
import { mapStateToProps, mapDispatchToProps } from "./container";
import {connect} from 'react-redux';
import { Card, ListItem,Button,Header,CheckBox } from 'react-native-elements';
import img from './product.png'
import Icon from 'react-native-vector-icons/Ionicons'
class SettingsFCScreen extends Component{
    componentWillMount(){
        this.props.loadComponents();
    }
    static  navigationOptions = {
        drawerIcon:(
          <Icon name='ios-leaf' size={26} color='#3CB371'/>
          )
    }
    renderCards=()=>{
        return this.props.components.map((component, i) => {
        return(
            <Card key={i}>
                   <ListItem
                    title={component.name}
                    leftIcon={<Icon name='ios-leaf' size={26}
                    color='#3CB371'/>}
                     />
```

```
            <Text style={{marginBottom: 10, color:'#696969'}}>
                {"тип: " + component.type}
            </Text>
            <Button
                buttonStyle={{borderRadius: 0, marginLeft: 0, marginRight:
                    0, marginBottom: 0, backgroundColor:'#A0CB1B'}}
                title='Більше інформації'
                onPress={()=> this.props.navigation.navigate('Component',
                    { component: component })}/>
        </Card>
    )
})
}
render(){
    return(
        <Container style={styles.home} >
            <Header
                backgroundColor='#A0CB1B'
                leftComponent={{ icon: 'menu', color: '#fff' }}
                centerComponent={{ text: 'Компоненти', style: { color:
                    '#fff' } }}
                rightComponent={{ icon: 'home', color: '#fff' }}
                />
            <Content style={styles.container}>
            {this.renderCards()}
            </Content>
        </Container>
    );
```

```
    }
}
export default connect(
    mapStateToProps,
    mapDispatchToProps
  )(SettingsFCScreen);


const styles = StyleSheet.create({
    home:{
        flex: 1,
        marginBottom:20
      },
        container: {
          flex: 1,
          paddingBottom: 10
        },
  });
```

Додаток З
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ’Я СПОЖИВАЧІВ

Виконала: Яшан Ольга Сергіївна

Науковий керівник: старший викладач кафедри ПЗКС, к.т.н., Хіцко Яна Володимирівна

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: покращити спосіб контролю якості продукції та пришвидшити аналіз впливу продуктів харчування для пересічних споживачів

Завдання:

1. Проаналізувати сучасні технології для розробки веб та мобільних застосунків, а також технології для керування базами даних
2. Розробити інформаційну систему для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів, яка складається з веб-застосунку, мобільного застосунку та бази даних
3. Протестувати розроблені застосунки

АКТУАЛЬНІСТЬ

Існуючі проблеми:

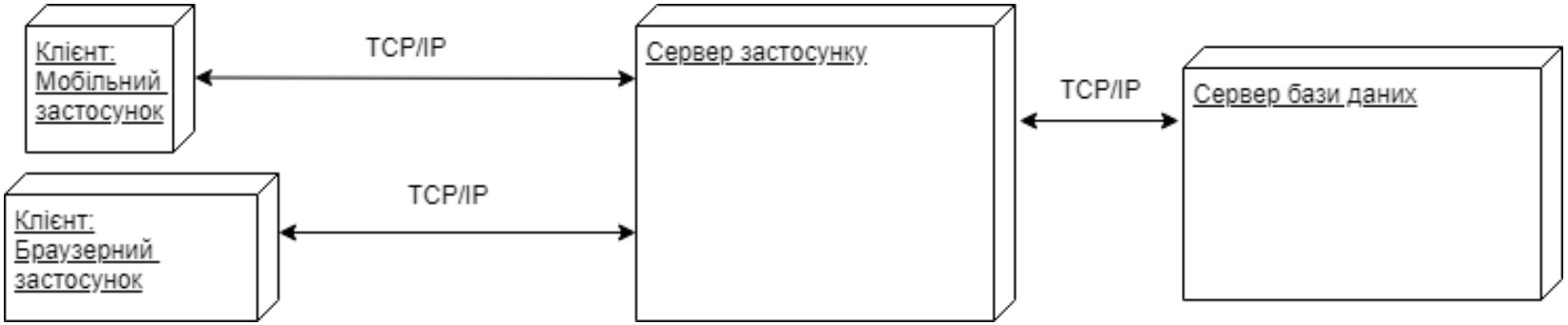
1. Неможливість оцінити вплив складників продукту на організм
2. Неочевидність назв складників для пересічного споживача
3. Відсутність освітніх додатків у сфері здорового харчування
4. Відсутність засобів для збирання та аналізу даних про продукти, як наслідок, неможливість контролю якості

Розроблена інформаційна система для аналізу впливу продуктів харчування на здоров'я споживачів

Веб-ресурс

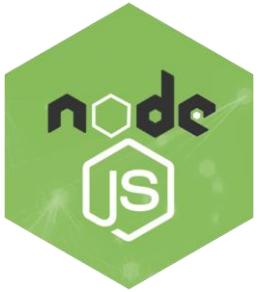
Мобільний застосунок

Архітектура системи
Трирівнева клієнт-серверна архітектура



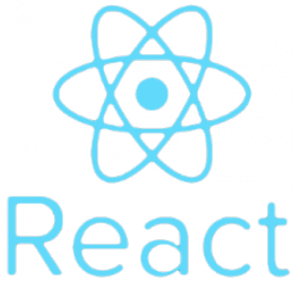
- надає можливість конфігурування, завдяки ізольованості рівнів один від одного
- низькі вимоги до швидкості мережі між клієнтами та сервером застосунків
- низькі вимоги до продуктивності і технічних характеристик клієнтів - клієнтом може бути навіть мобільний телефон

Засоби розроблення



Node.js – це крос-платформне середовище виконання JavaScript, яке виконує код поза браузером

Node.js дозволив написання застосунків-клієнтів та сервера застосунку з допомогою однієї мови програмування – JavaScript



React.js – це бібліотека, написана мовою JavaScript, для створення користувацьких інтерфейсів

React.js допоміг реалізувати користувацькі інтерфейси веб-застосунку та мобільного застосунку, а також вирішити проблему часткового оновлення вмісту веб-сторінки

MongoDB – це документо-орієнтована система керування базами даних з відкритим вихідним кодом

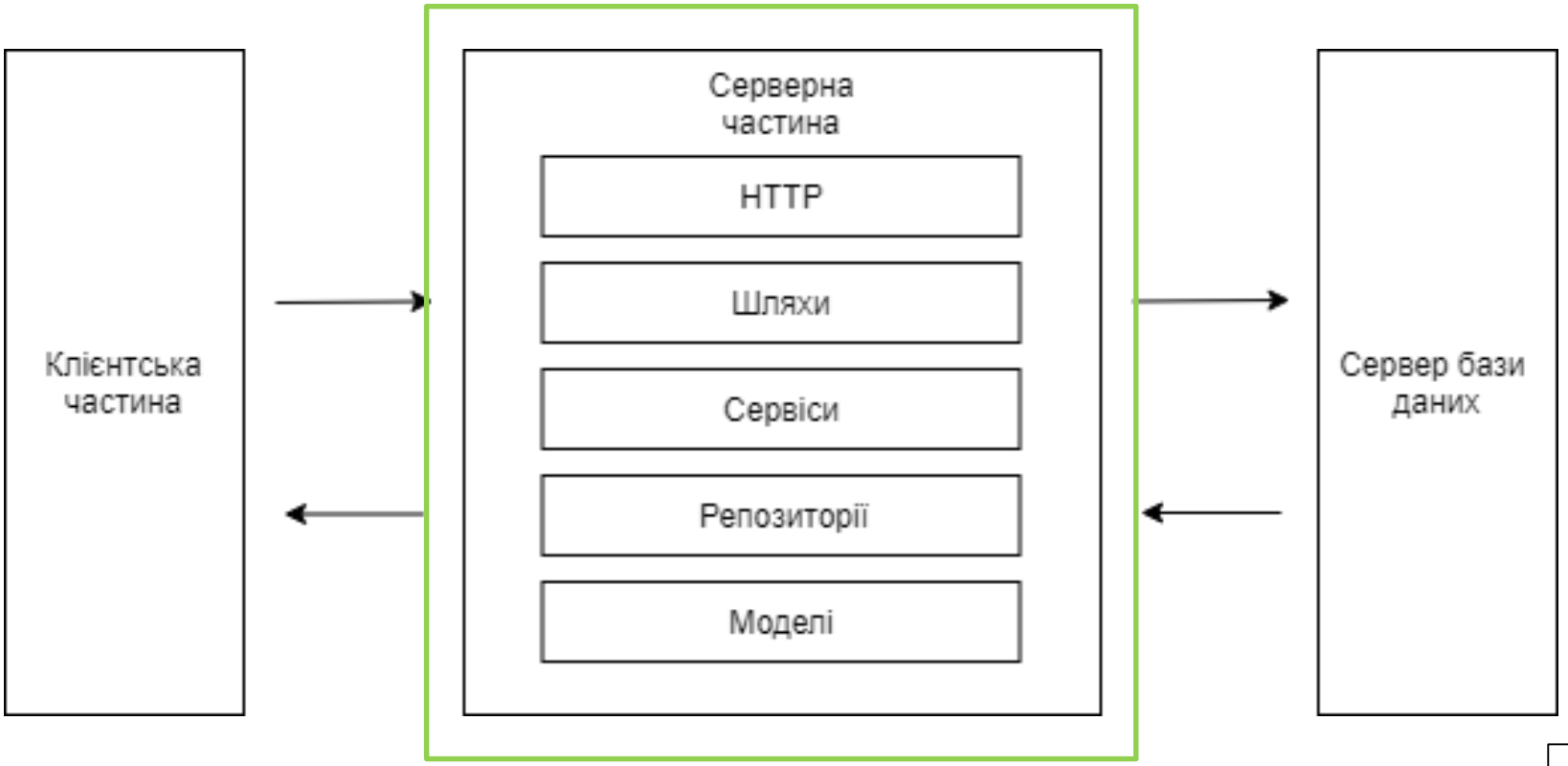
MongoDB:

- належить до класифікації NoSQL
- слугує для зберігання ієрархічних структур даних
- використовує JSON-подібні документи та схему баз даних

Сервер застосунку

Архітектура

Серверна частина має поділ на модулі, тому що засіб розроблення Node.js вимагає таку структуру



Сервер застосунку

Архітектура: модулі

1. HTTP: обробляє запит та створює структуру даних з отриманої із запиту інформації
2. Шляхи: визначає спосіб обробки запиту, спираючись на шлях, за яким цей запит надійшов
3. Сервіси: містить всю бізнес-логіку застосунку
4. Репозиторії: містить методи для роботи з об’єктами бази даних
5. Моделі: описує структуру об’єктів бази даних та містить засоби валідації

Сервер застосунку

Функціональні можливості

- повернення записів з БД
- створення запису у БД
- оновлення записів у БД
- видалення записів з БД

про: користувача, продукт, компонент, дієту

База даних
Моделі

Product
+ name: String
+ components: [ ]
+ rating: Number
+ producer: String
+ barCode: Number
+ date: Date

Diet
+ name: String
+ description: String
+ forbidenComponents: [ ]

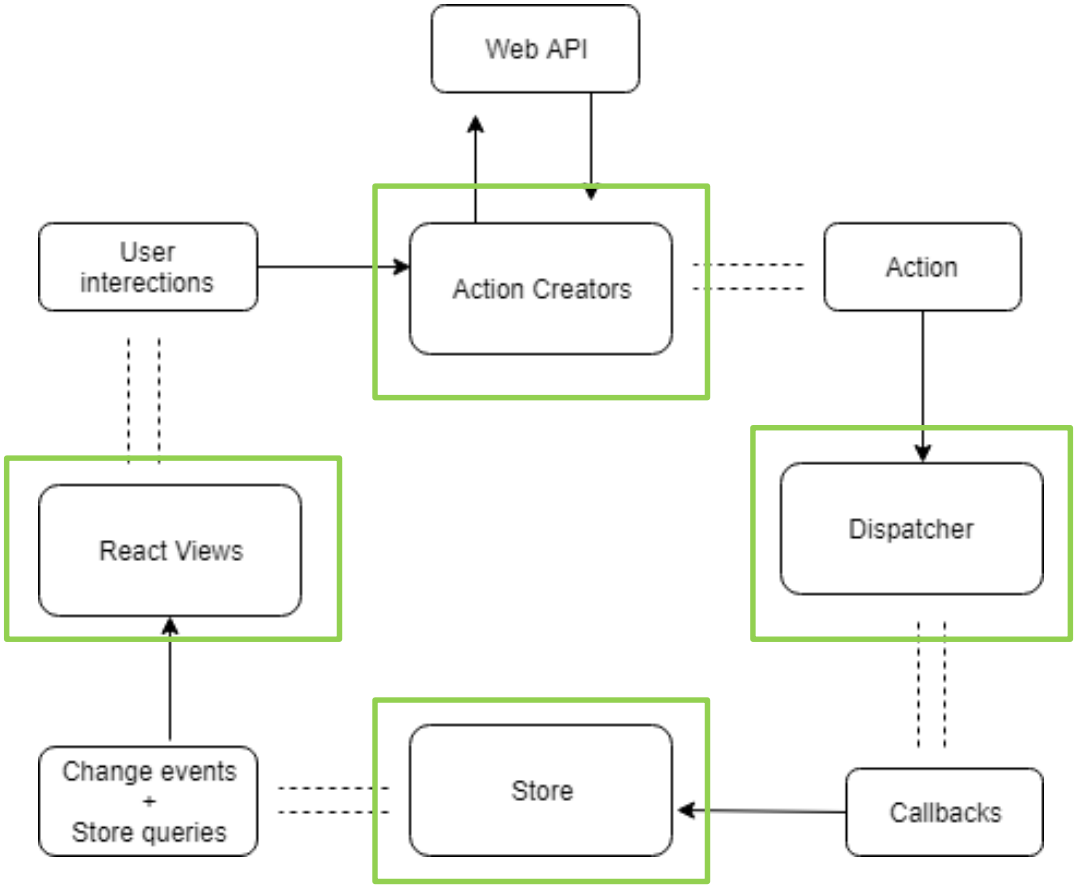
Component
+ name: String
+ type: String
+ description: String

User
+ login: String
+ password: String
+ name: String
+ surname: String
+ age: Number
+ gender: String
+ role: String
+ forbidenComponents: [ ]
+ diet: [ ]
+ rating: Number
+ lastProducts: [ ]
+ history: [ ]

Застосунок-клієнт
Архітектура

Веб-застосунок та мобільний застосунок розроблені з допомогою технологій React та Redux, які вплинули на підхід до архітектурного проектування

Саме через використання однакового набору технологій ця архітектура є спільною для веб та мобільного застосунків

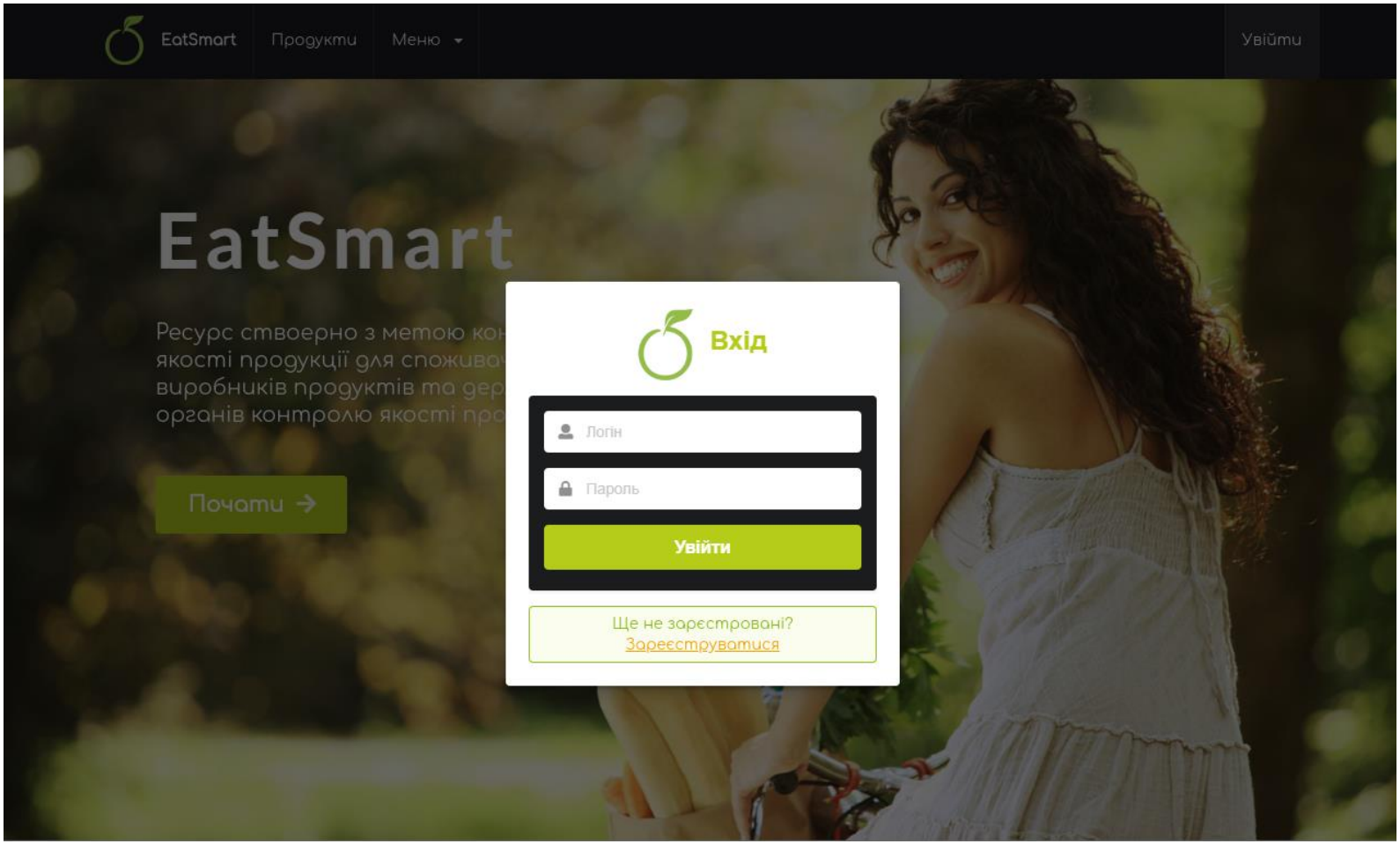


Веб-застосунок

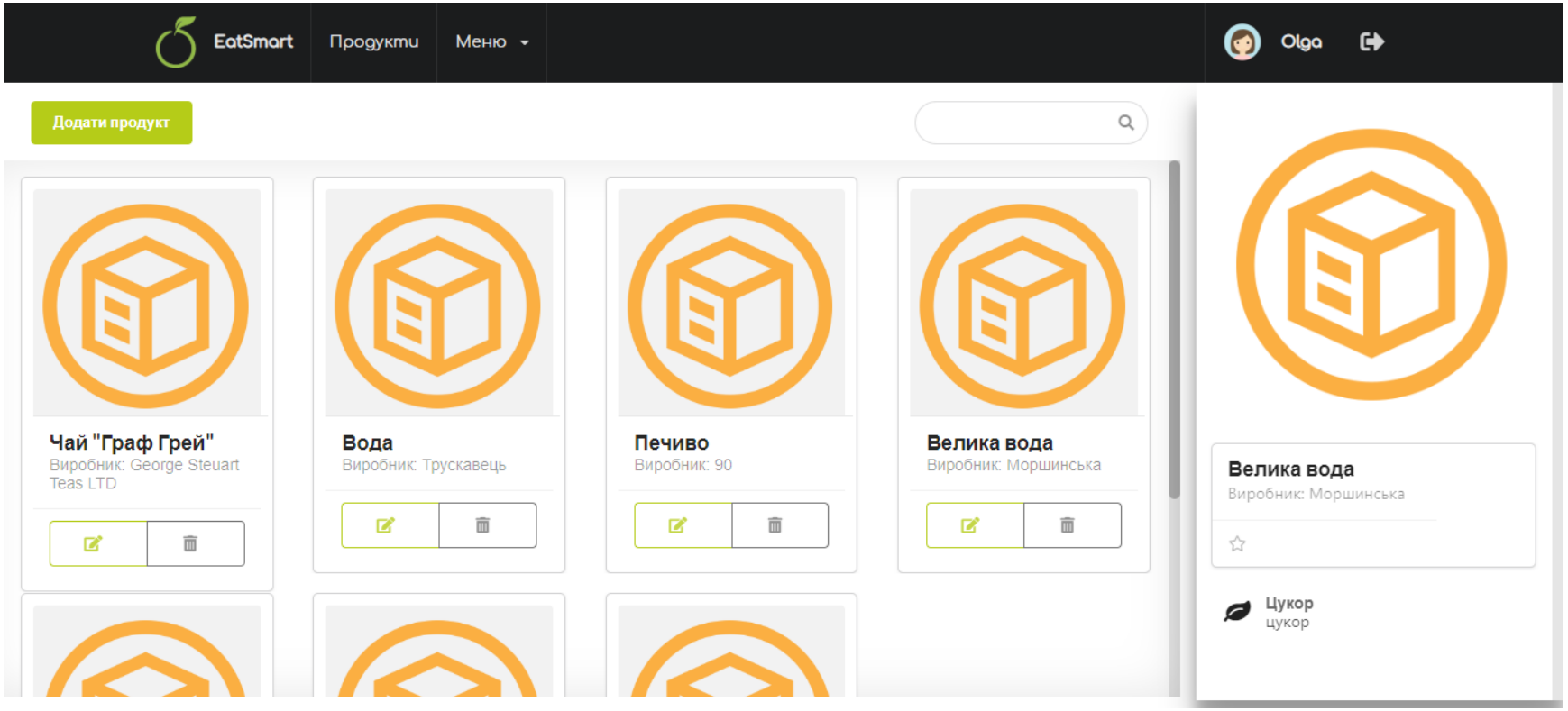
Функціональні можливості

- перегляд інформації про продукт
- перегляд інформації про компонент
- оновлення інформації про продукт
- оновлення інформації про компонент
- налаштування особливостей раціону користувача
- вибір дієти користувача
- перегляд історії з’їдених продуктів
- аналіз впливу спожитих продуктів на здоров’я споживача

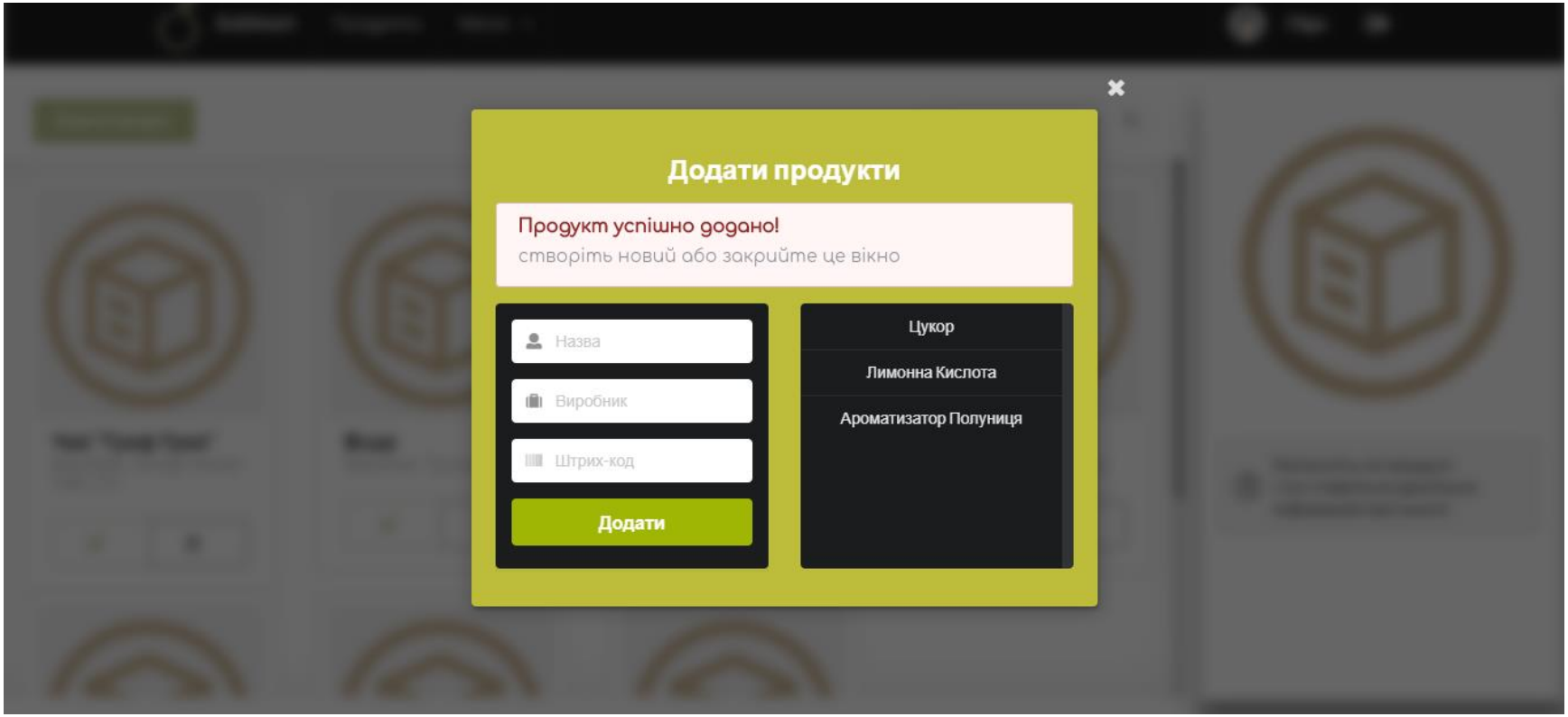
Веб-застосунок
Інтерфейс



Веб-застосунок
Інтерфейс



Веб-застосунок
Інтерфейс



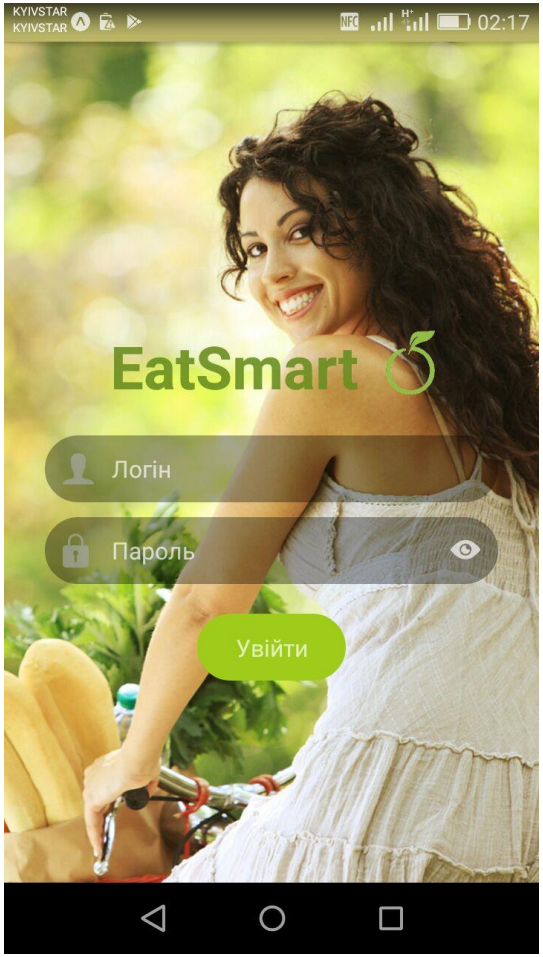
Мобільний застосунок

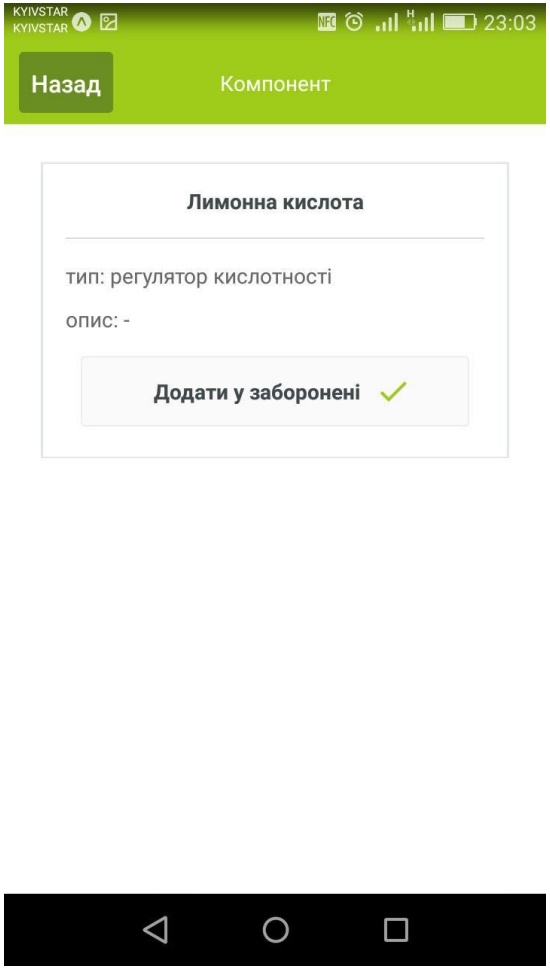
Функціональні можливості

- перегляд інформації про продукт
- перегляд інформації про компонент
- налаштування особливостей раціону користувача
- зчитування штрих-коду продукту, для отримання інформації про продукт
- збереження історії з’їдених продуктів

Мобільний застосунок

Інтерфейс



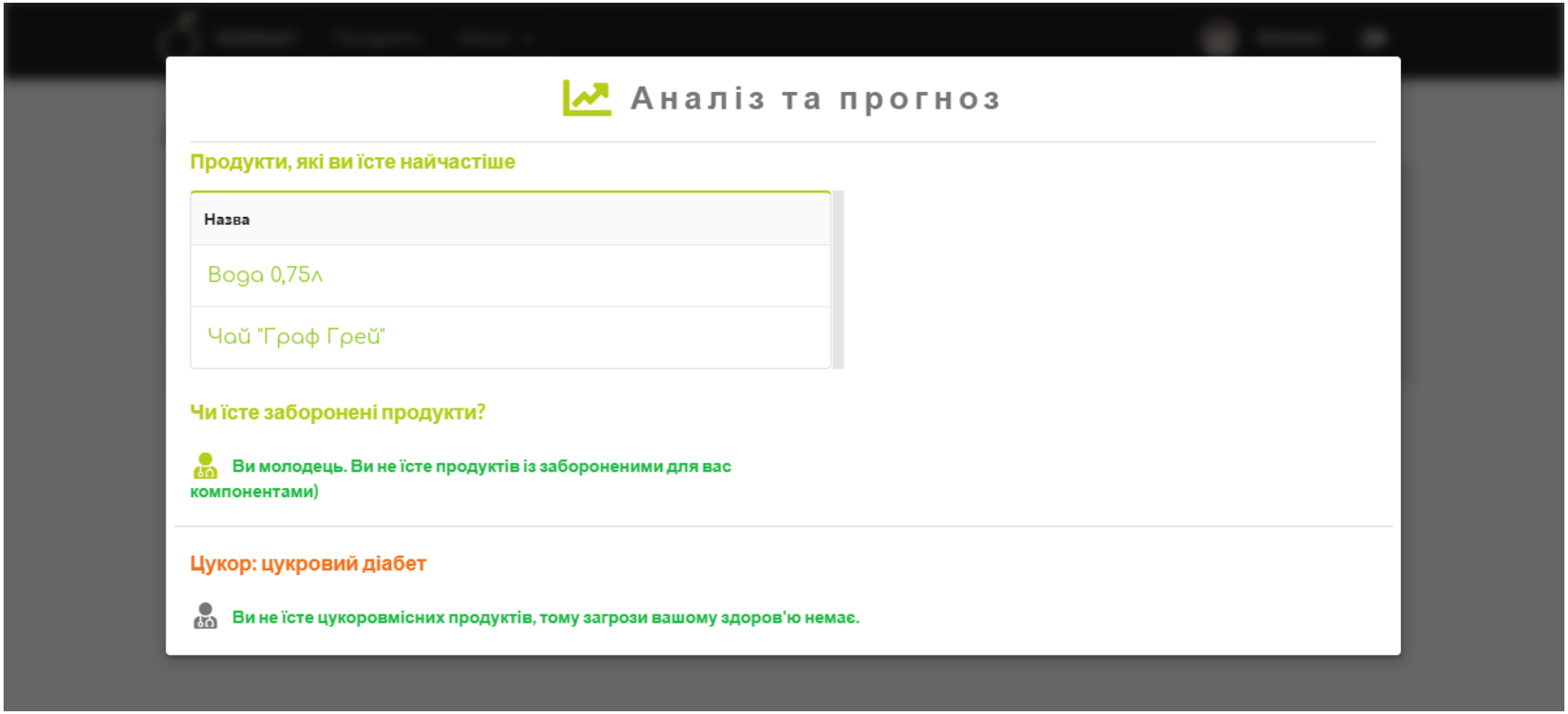


Розроблені алгоритми
Аналіз

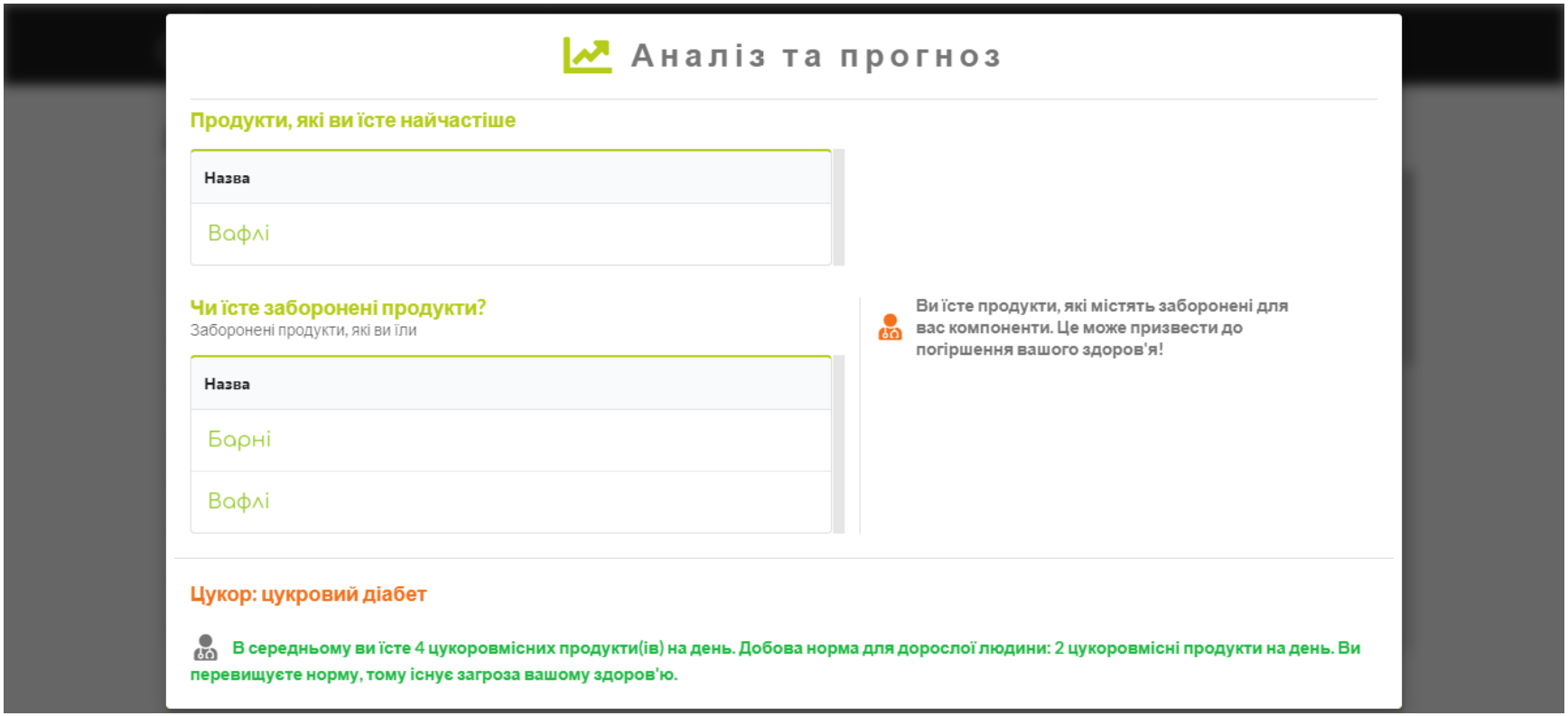
Для аналізу раціону харчування були обрані такі метрики:

- мода серед сукупності усіх спожитих продуктів - продукти, які споживаються найбільш часто
- входження продуктів, що містять заборонені компоненти, у множину спожитих продуктів
- частота вживання цукоровмісних продуктів – кількість цукоровмісних продуктів на одиницю часу, тобто день. Якщо прийняти за дійсне, що добова норма це два цукоровмісні продукти, то можна спрогнозувати наслідки перевищення даної норми

Розроблені алгоритми
Результати



Розроблені алгоритми
Результати



Тестування

Критерії оцінювання якості:
Функціональні можливості – характеристика корисності системи
Інтерфейс користувача – характеристика зручності системи
Платформа – характеристика доступності системи

- **Функціональне тестування** системи виконувалося з допомогою діаграми переходів станів, яка у зручній формі відображає всі стани, функції системи та реакції на певні дії
- **Тестування інтерфейсу користувача** виконувалося ручним способом
- **Тестування доступності** полягало у використанні розробленої системи у різних браузерах та на різних операційних системах мобільних пристроїв

Результати тестування

У процесі тестування було перевірено наступне:

- працездатність усіх функціональних можливостей, згідно функціональних вимог
- функціональна працездатність елементів сторінок веб-застосунку
- функціональна працездатність елементів екранів мобільного застосунку
- зручність роботи з веб-застосунком
- зручність роботи з мобільним застосунком
- правильність роботи алгоритмів аналізу
- робота застосунків у відповідних середовищах: браузерах Google Chrome, Mozilla Firefox, Safari та на мобільних пристроях з ОС Android

Тестування довело те, що розроблений програмний продукт відповідає усім критеріям якості

Порівняння з аналогами

Е код - харчові добавки

Функціональні можливості	Розроблений застосунок надає більше корисних для користувача функцій ніж аналог
Інтерфейс користувача	Інтерфейс розробленого застосунку більш сучасний та приємніший для користувача. Мова інтерфейсу розробленого застосунку українська, тому він є зручнішим для україномовної аудиторії, аніж російськомовний інтерфейс аналогу
Платформа	Розроблений застосунок доступний не тільки для користувачів мобільних телефонів з операційною системою Android, а й для користувачі мережі інтернет, на відміну від аналогу

Порівняння з аналогами

OutScanner

Функціональні можливості	Розроблений застосунок надає однакові можливості для користувача, що й розглянутий аналог
Інтерфейс користувача	Інтерфейси двох застосунків досить зручні та приємні для користувача. Мова інтерфейсу розробленого застосунку українська, тому він є зручнішим для україномовної аудиторії, аніж англомовний інтерфейс аналогу
Платформа	Розроблений застосунок доступний для користувачів мобільних телефонів з операційною системою Android, а також для користувачі мережі інтернет, на відміну від аналогу, який доступний лише для користувачів мобільних телефонів з ОС iOS

ВИСНОВКИ

Розроблено інформаційну систему для аналізу впливу продуктів харчування на здоров'я споживачів, яка може:

- допомогти пересічному споживачу аналізувати якість продуктів харчування
- допомогти органам якості контролю продуктів харчування контролювати якість продуктів

В ході роботи над дипломним проектом було:

1. Проаналізовано аналоги та визначено вимоги до розроблюваного програмного продукту
2. Визначено та застосовано архітектури сервера та клієнтів
3. Розроблено програмне забезпечення застосунків серверу та клієнтів
4. Протестовано розроблену інформаційну систему для аналізу та прогнозування продуктів харчування на здоров’я споживачів

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ'Я СПОЖИВАЧІВ

Програма та методика тестування

ДП.045490-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.С. Яшан

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Інформаційна система для аналізу та прогнозування впливу продуктів харчування на здоров'я споживачів, яка складається з веб-ресурсу та мобільного додатку.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок веб-ресурсу;
2) функціональна працездатність елементів екранів мобільного додатку;
3) зручність роботи з веб-сайтом;
4) зручність роботи з мобільним додатком;
5) правильність роботи алгоритмів аналізу.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом White Box Testing. Проводяться наступні види тестування:

1) функціональне тестування, зокрема на рівні модульного тестування;
2) нефункціональне тестування:
    - UI Testing (Тестування користувацького інтерфейсу);
    - Usability Testing.
3) тестування сумісності.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Виконується ручне та автоматизоване тестування.

Працездатність системи перевіряється шляхом:

1) динамічного ручного тестування на відповідність функціональним вимогам;
2) статичного тестування коду;

3) тестування веб-ресурсу в різних веб-браузерах;
4) тестування мобільного додатку на різних мобільних телефонах;
5) ручного тестування зручності використання інтерфейсу;
6) автоматизоване тестування з допомогою бібліотек Jest та Enzyme.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВПЛИВУ ПРОДУКТІВ ХАРЧУВАННЯ НА ЗДОРОВ'Я СПОЖИВАЧІВ

Керівництво користувача

ДП.045490-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Я.В. Хіцко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.С. Яшан

2019

ЗМІСТ

1. Опис структури веб-сайту

Користувацький інтерфейс веб-сайту інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів логічно поділений на сторінки. Поділ зумовлений метою, кожної сторінки. Певні функції, реалізовані в окремих елементах відображення – модальних вікнах. Кожна сторінка чи модальне вікно містить елементи керування. Для реалізації інтерфейсу була використана українська мова.

Перелік сторінок веб-сайту:

- Головна сторінка – містить опис системи та елементами керування, які надають можливість увійти в систему, зареєструватися у ній, перейти на сторінки «Продукти» і «Компоненти». Якщо користувач, вже здійснив вхід в систему, то йому також доступні елементи керування, які надають можливість перейти на сторінку користувача та сторінку «Історія».
- Продукти – на цій сторінці міститься інформація про продукти. Якщо користувач вже здійснив вхід у систему, то він може натиснути на продукт і він з’явиться на сторінці користувача в розділі «Останні переглянуті продукти». Якщо роль користувача у системі – «адміністратор», тоді йому також доступні елементи керування, які надають можливість додавання/видалення/оновлення продукту.
- Компоненти – на цій сторінці міститься інформація про компоненти. Якщо роль користувача у системі – «адміністратор», тоді йому також доступні елементи керування, які надають можливість додавання/видалення/оновлення компоненту.

- Сторінка користувача – на цій сторінці міститься інформація про користувача. Сторінка має логічний поділ на блоки: «Інформація про користувача», який містить ім’я, вік та стать користувача, а також заборонені для користувача компоненти і дієту; «Останні переглянуті продукти» містить продукти, які користувач переглянув нещодавно. На цій сторінці користувачу доступні елементи курування, які дозволять йому змінити перелік заборонених компонентів, змінити дієту та переглянути інформацію про кожен продукт, який міститься в блоці «Останні переглянуті продукти». Блок «Останні переглянуті продукти» також може бути корисним для користувача, адже надає підказки, які продукти містять заборонені компоненти для користувача, а які ні.
- Історія – сторінка, яка містить перелік продуктів, які були відсканованої користувачем з допомогою мобільного застосунку та вказано, що користувач збирається їх з’їсти. У таблиці поряд з кожним продуктом вказана дата та час, коли продукт був доданий в історію. На цій сторінці користувачу доступні елементи керування, які дозволять йому проаналізувати та спрогнозувати вплив на здоров’я продуктів, які він з’їв, а також переглянути інформацію про кожен продукт.

Між сторінками «Головна сторінка», «Продукти», «Компоненти», «Сторінка користувача» та «Історія» можна здійснити перехід за допомогою меню, яке розташоване у верхній частині екрану на кожній із сторінок. Елементи меню «Сторінка користувача» та «Історія» з’являться у меню після входу у систему. Сторінки, модальні вікна та переходи між ними зображені на рис. 1.

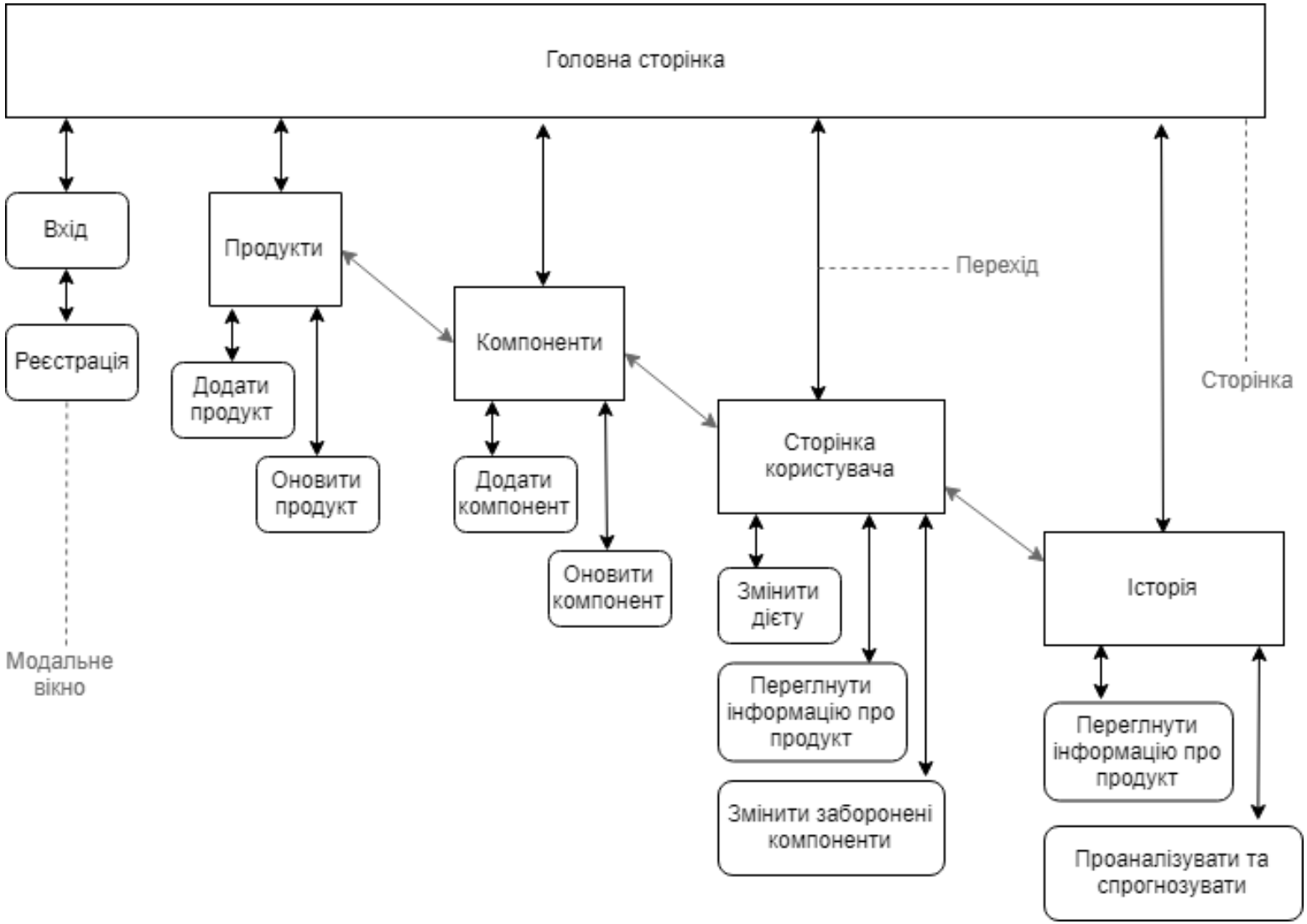


Рис. 1. Структура веб-сайту

2. Опис структури мобільного додатку

Користувацький інтерфейс мобільного застосунку інформаційної системи для аналізу та прогнозування впливу продуктів харчування на здоров’я споживачів логічно поділений на екрани. Поділ зумовлений метою, кожного екрану. Кожен екран містить елементи керування. Для реалізації інтерфейсу була використана українська мова.

Перелік екранів мобільного застосунку:

- Вхід – екран для авторизації користувача у системі. Містить наступні елементи керування: поле «Логін», поле «Пароль», кнопку «Відобразити пароль/сховати пароль» та кнопку «Увійти». Якщо дані введені правильно, з’явиться екран «Останні переглянуті продукти», інакше – з’явиться попереджальне повідомлення.
- Останні переглянуті продукти – екран, який містить перелік продуктів та елементи керування, які дозволять перейти на екран «Компоненти продукту». Також на цьому екрані міститься елемент керування, який надає можливість вийти із системи.
- Продукти – екран, який містить перелік продуктів та елементи керування, які дозволять перейти на екран «Компоненти продукту».
- Компоненти продукту – екран, який містить склад продукту та елемент керування кнопку «Назад», яка дозволить повернутися на екран «Останні переглянуті продукти».
- Компоненти – екран, який містить компоненти та елементи керування, які дозволять перейти на екран «Компонент».
- Компонент – екран, який містить інформацію про компонент та елемент керування, який дозволить додати/видалити компонент із

заборонених компонентів користувачу. Також тут міститься елемент керування кнопка «Назад», яка дозволить повернутися на екран «Компоненти».

- Відсканувати – екран, який дозволяє відсканувати штрих-код продукту, для цього необхідно піднести камеру телефону та сфокусувати її на штрих-коді продукту. Якщо штрих-код буде знайдено у системі, з’явиться екран «Продукт», інакше – з’явиться екран з відповідним повідомленням. Якщо продукт, штрих-код якого було відскановано, містить заборонені для користувача компоненти з’явиться відповідне повідомлення – для продовження роботи необхідно натиснути на частину екрану, де повідомлення нема. На екрані «Продукт» є елемент керування кнопка «Назад», яка дозволить повернутися на екран «Відсканувати». На екрані «Відсканувати» також є елемент керування кнопка «Повернутися», яка дозволить повернутися на екран «Останні переглянуті продукти».

Між екранами «Останні переглянуті продукти», «Продукти», «Компоненти» та «Відсканувати» можна здійснити перехід за допомогою меню, яке з’являється, якщо протягнути з лівого боку екрану до правого пальцем на кожному з цих екранів та натиснути відповідний елемент керування. Прибрати меню можна зворотнім рухом або натиснути на частину екрану де нема меню. Екрани та переходи між ними зображені на рис. 2.

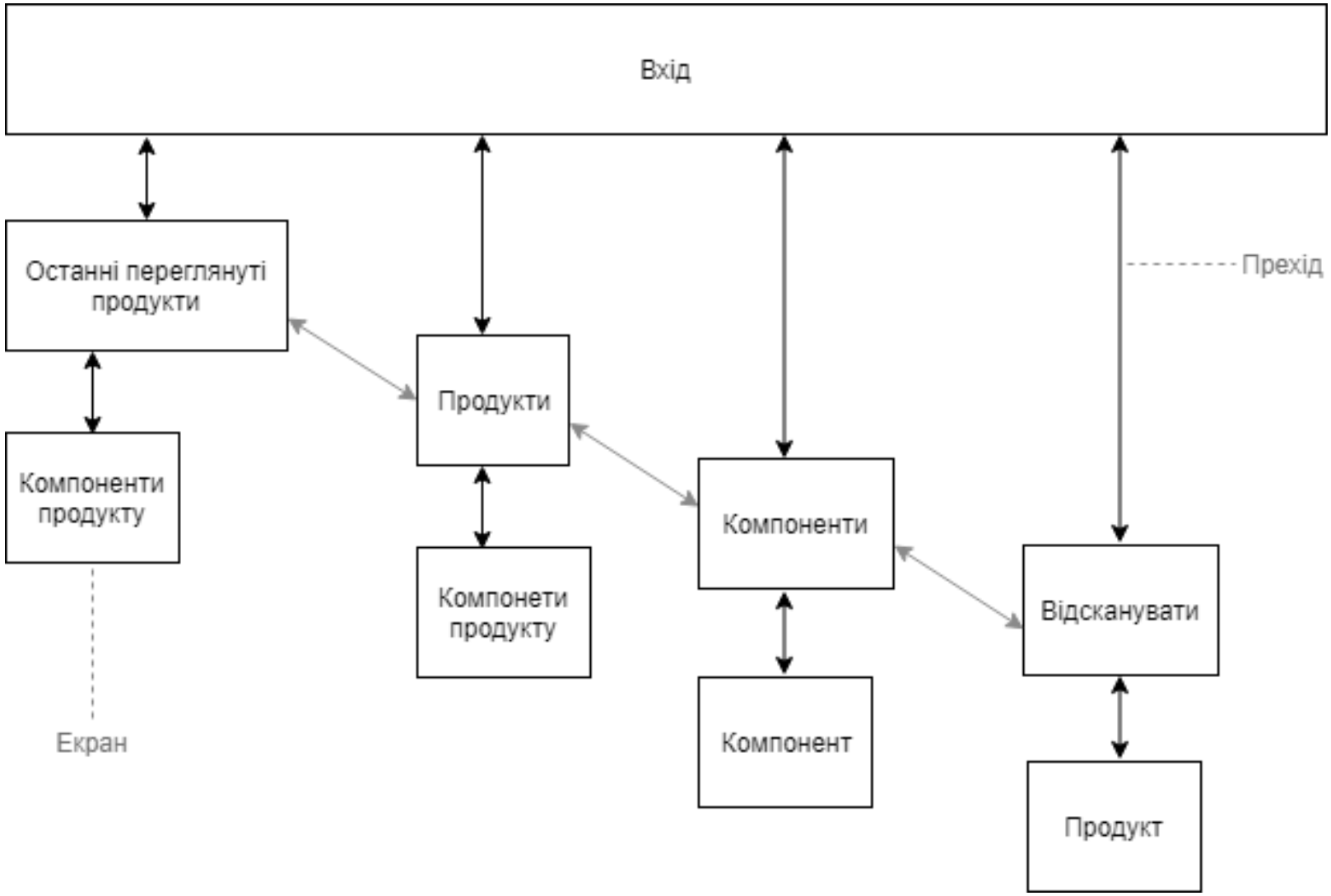


Рис. 2. Структура мобільного застосунку

3. Процедура авторизації користувача

3.1. Процедура авторизації користувача на веб-сайті

Виконати дану операцію можливо після натискання елементу керування кнопки «Увійти» на будь-якій сторінці у правому верхньому куті. Після натискання з’явиться модальне вікно з формою зображене на рис. 3.

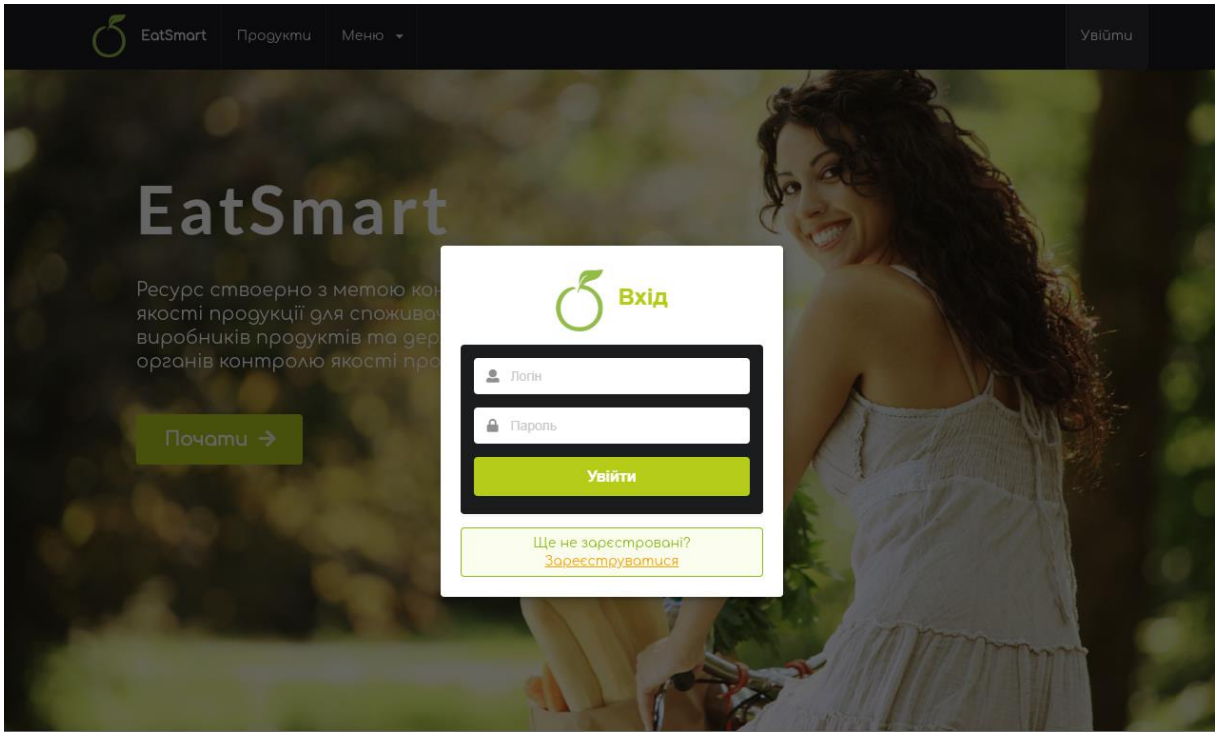


Рис. 3. Форма авторизації

Форма має два поля: логін та пароль. Необхідно ввести значення у ці поля та натиснути елемент керування кнопку «Увійти». За умови правильно введених даних, відкриється сторінка користувача, інакше – з’явиться відповідне повідомлення, як зображено на рис. 4.

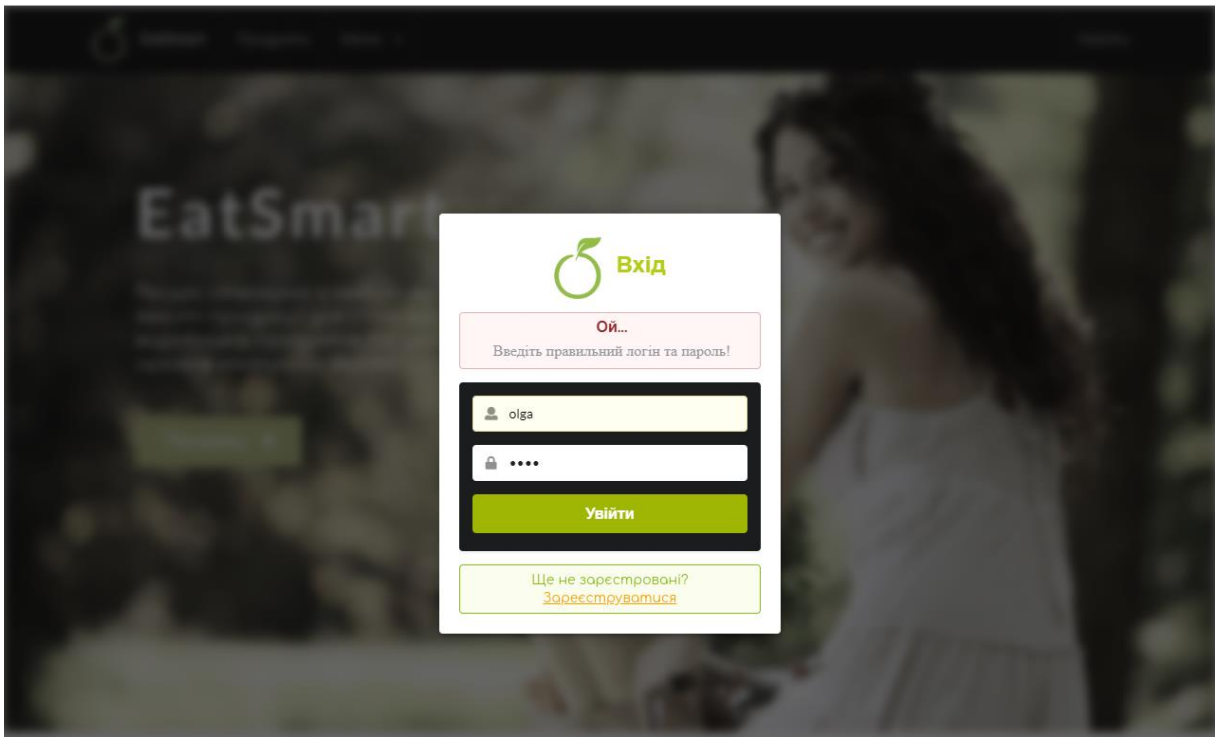


Рис. 4. Форма авторизації з повідомленням про помилкові дані

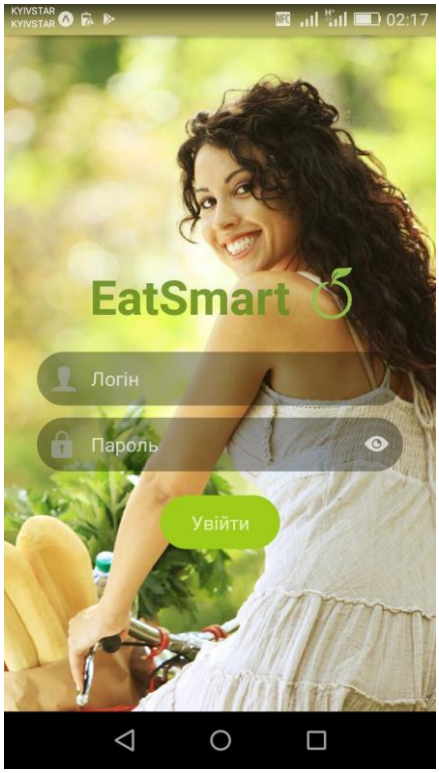


Рис. 5. Форма авторизації

Рис. 6. Форма авторизації з повідомленням

3.2. Процедура авторизації користувача у мобільному застосунку

Виконати дану операцію можливо після відкриття додатку або після виходу із системи на екрані вхід. На екрані буде форма зображена на рис. 5.

Форма має два поля: логін та пароль. Необхідно ввести значення у ці поля та натиснути елемент керування кнопку «Увійти». За умови правильно введених даних, відкриється екран «Останні переглянуті продукти», інакше – з’явиться відповідне повідомлення, як зображено на рис. 6.

4. Процедура додавання продукту

Виконати дану операцію можливо тільки на веб-сайті на сторінці «Продукти» після натискання елементу керування кнопки «Додати продукт», яка відобразиться за умови, що ваша роль у системі – «Адміністратор». Після натискання з’явиться модальне вікно з формою зображене на рис. 7.

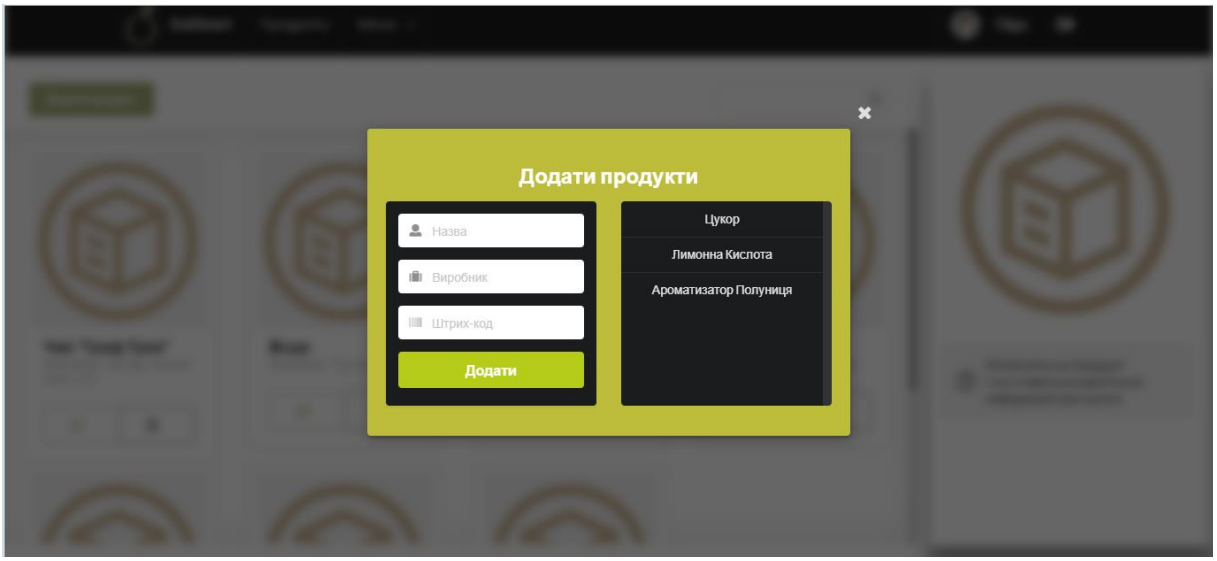


Рис. 7. Форма додавання продукту

Форма має три поля: назва, тип та штрих-код. З правого боку є список компонентів. Додати/видалити компонент можна натиснувши на нього. Необхідно ввести значення у ці поля, обрати компоненти та натиснути елемент керування кнопку «Додати». За умови успішного створення продукту, з’явиться повідомлення, як на рис. 8. Далі можна продовжити додавати продукти або закрити модальне вікно, натиснувши на хрестик у верхньому правому куті.

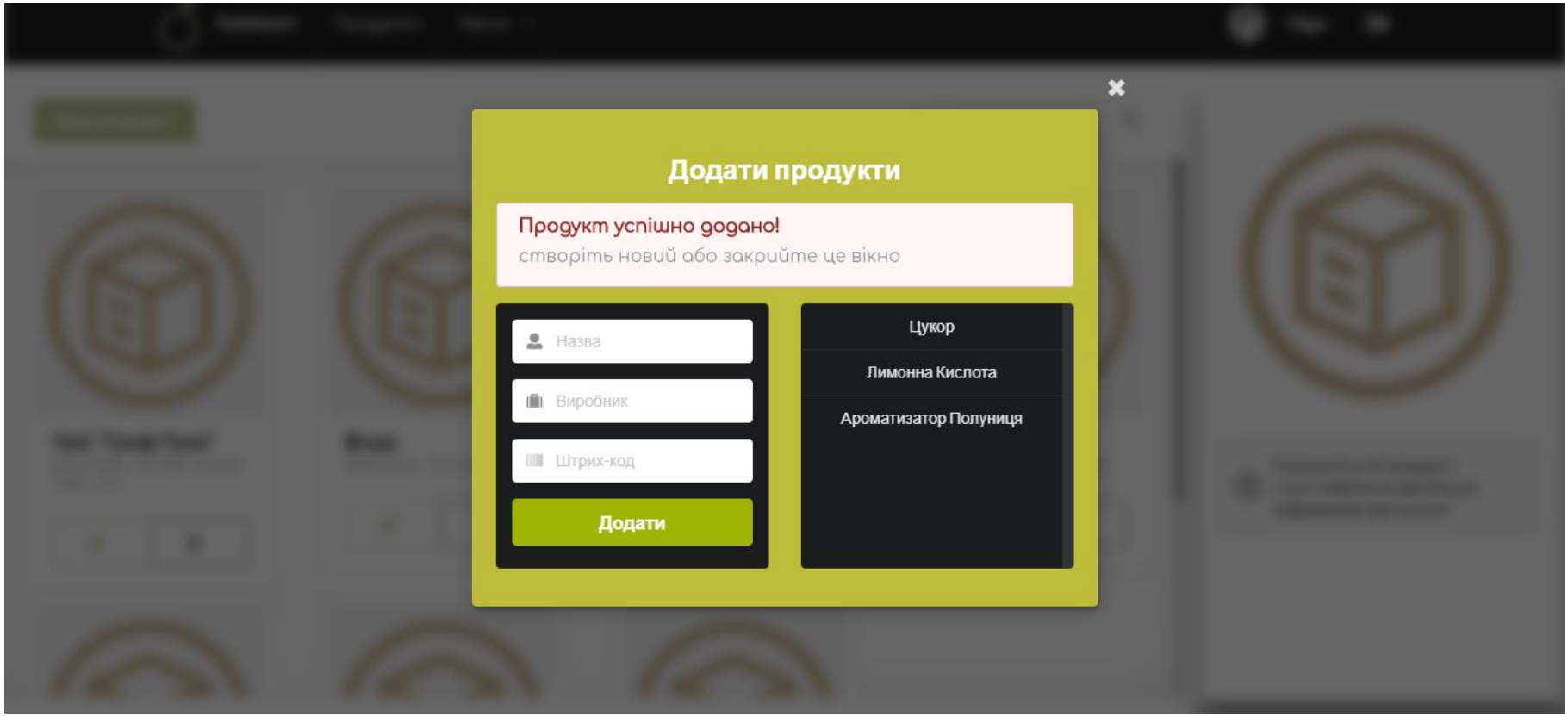


Рис. 8. Форма додавання продукту з повідомленням

5. Процедура видалення компоненту

Виконати дану операцію можливо тільки на веб-сайті на сторінці «Продукти» після натискання елементу керування кнопки «Видалити», яка відобразиться за умови, що ваша роль у системі – «Адміністратор», як зображено на рис. 9. Після натискання кнопки компонент зникне.

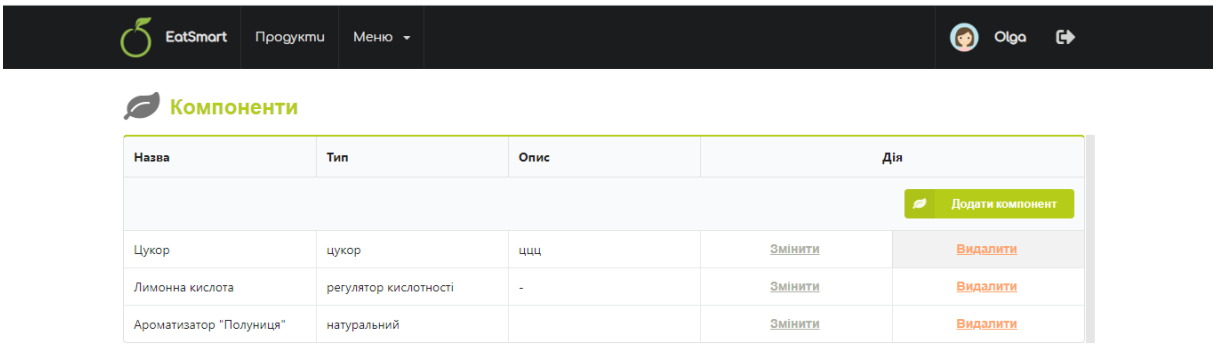


Рис. 9. Сторінка компоненти

6. Процедура зміни переліку заборонених компонентів

6.1. Процедура зміни переліку заборонених компонентів користувача на веб-сайті

Виконати дану операцію можливо на веб-сайті на сторінці користувача після натискання елементу керування кнопки «Заборонені компоненти».

Після натискання на кнопку «Заборонені компоненти» з’явиться модальне вікно, яке показано на рис. 10.

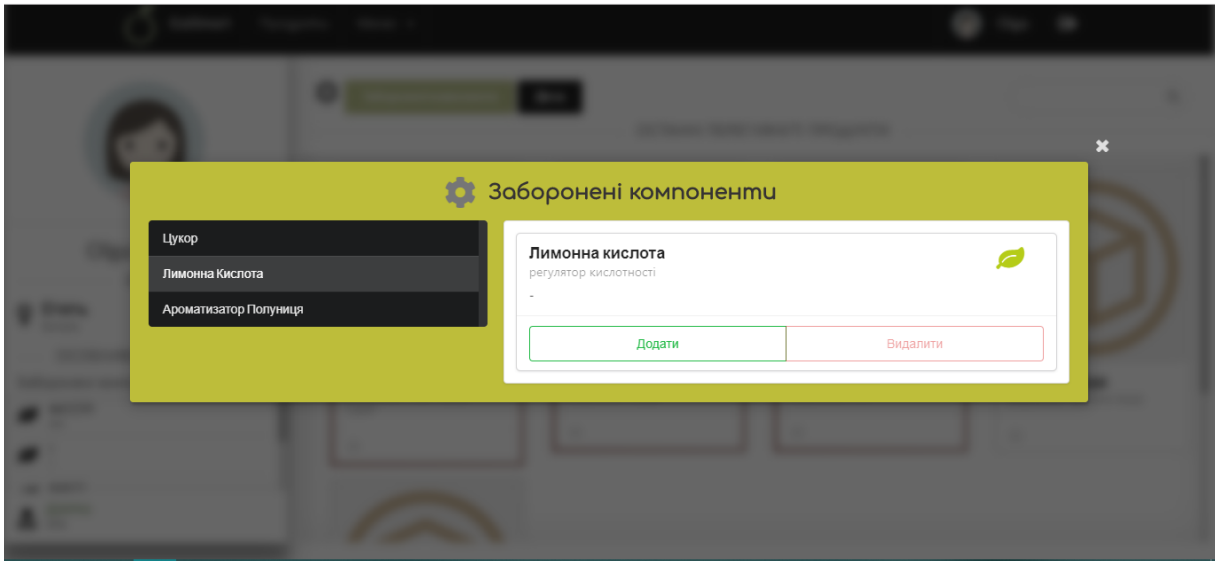


Рис.10. Форма додавання/видалення компоненту із переліку заборонених компонентів

У лівій частині перелік компонентів, праворуч картка з інформацією про обраний компонент. Щоб отримати інформацію про компонент у правій частині вікна, потрібно натиснути на нього у лівій частині вікна. Якщо цей компонент вже міститься у заборонених компонентах, тоді кнопка «Видалити» буде активною, а «Додати» – неактивною, інакше, якщо компонент не є забороненим, кнопка «Додати» буде активною, а «Видалити» – неактивно. Після завершення налаштувань натиснути хрестик для того, щоб закрити вікно.

6.2. Процедура зміни переліку заборонених компонентів користувача у мобільному застосунку

Цю процедуру також можливо виконати у мобільному додатку. Потрібно перейти на екран «Компоненти» натиснути елемент керування кнопку «Більше інформації». Для налаштування на екрані компонента буде елемент керування кнопка «Додати у заборонені/видалити із заборонених»,

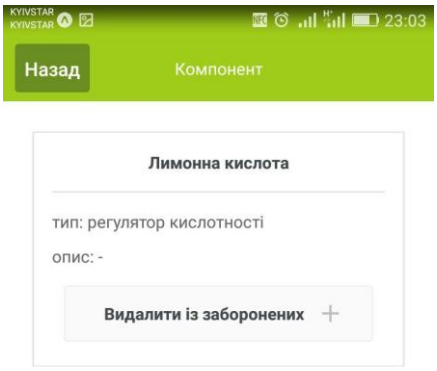


Рис. 12. Екран «Компонент» з кнопкою «Видалити із заборонених»

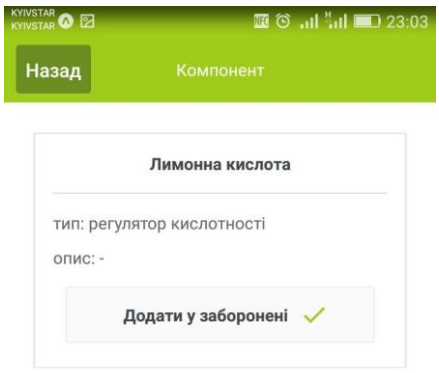


Рис. 13. Екран «Компонент» з кнопкою «Додати в заборонені»

залежно від того, чи є цей компонент у переліку заборонених для користувача компонентів, як показано на рис. 12 і 13.

7. Користування екраном «Відсканувати» мобільного застосунку

Мета даного екрану надати можливість користувачу мобільного застосунку відсканувати штрих-код продукту та отримати інформацію про склад продукту. Для цього необхідно перейти на екран «Відсканувати», навести камеру мобільного телефону на штрих-код і сфокусувати, натиснувши на екран, якщо ваш мобільний телефон не робить фокусування автоматично (див. рис. 14). За кілька секунд з’явиться екран з інформацією про продукт або екран з повідомленням, що продукту з таким штрих-кодом не знайдено. Щоб продовжити сканувати потрібно натиснути на елемент керування кнопку «Назад».

Рис. 14. Екран «Відсканувати»