Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка моніторингової системи для продажу меду та тестування його якості»

Виконав студент 4 курсу,
групи ЗПІ-18б
спеціальності
121 «Інженерія програмного забезпечення»
Тохи В.В.
(прізвище та ініціали)
Керівник к.т.н., доц. каф ПЗ Войтко В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Арсенюк І. Р.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ– 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань12 – інформаційні технології
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н
25 березня 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ
СТУДЕНТУ
Тосі Вадиму Валентиновичу

1. Тема роботи **Розробка моніторингової системи для продажу меду та тестування його якості**.

керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ,

затверджені наказом вищого навчального закладу від 24 березня 2022 року №66

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: операційна система – Windows 8/10; середовище програмування Brasket 2019; мови програмування – HTML, CSS, JS, PHP. Розмір вікна додатку не менше 1024x768. Кольоровий режим: TrueColor.

4. Зміст розрахунково-пояснювальної записки: аналіз використання систем для веб додатків, обґрунтування доцільності розробки; теоретичні основи розробленого методу; варіантний аналіз та обґрунтування вибору засобів реалізації системи; проєктування структур даних і модулів системи; розробка програмного коду веб системи; тестування роботи.

5. Перелік графічного матеріалу: мета, об'єкт та предмет дослідження; задачі дослідження; методи дослідження; наукова новизна; практичне значення; обґрунтування доцільності розробки; метод та модель реалізації веб системи; програмна реалізація веб системи; тестування; висновки, апробація, публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Войтко В.В., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану галузі, обґрунтування доцільності розробки	26.03.2022-05.05.2022	Вик.
2	Розробка методів і моделей реалізації веб системи	06.05.2022-12.05.2022	Вик.
3	Проектування структур даних та модулів системи	13.05.2022-20.05.2022	Вик.
4	Варіантний аналіз та обґрунтування вибору засобів веб системи	21.05.2022-23.05.2022	Вик.
5	Розробка програмного коду веб системи	24.05.2022-30.05.2022	Вик.
6	Тестування роботи	31.05.2022-03.06.2022	Вик.
7	Оформлення матеріалів до захисту БДР	04.06.2022-10.06.2022	Вик.

Студент _____ **Тоха В.В.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ **Войтко В.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 128 сторінок формату А4, на яких є 35 рисунків, 1 таблиця, список використаних джерел містить 19 найменувань.

У бакалаврській дипломній роботі розроблено веб систему, орієнтовану на продаж та моніторинг якості меду та продуктів бджільництва. Запропоновано метод для швидкого обрахування статистики та сортування товарів, який використовує функцію сортування Хоара (Quicksort), що дозволяє у режимі реального часу за відносно не великий проміжок часу відсортувати позиції за різними характеристиками. Розроблено модель роботи веб системи. Створена веб система була реалізована в інтернет-магазині «Трудяща бджілка».

Спроєктовано архітектуру веб системи. Створено працюючий відповідно до вимог інтернет-магазин, проведено його тестування. Розроблено керівництво користувача.

Розроблений метод і модель можуть бути використані при розробці сучасних веб систем, у яких передбачене сортування та робота з великими масивами даних.

Ключові слова: сортування Хоара, аналіз даних, моніторинг, веб-система.

ABSTRACT

The bachelor's thesis consists of 128 A4 pages, which have 35 figures, 1 table, a list of sources used contains 19 titles.

The web thesis focused on the sale and monitoring of the quality of honey and beekeeping products has been developed in the bachelor's thesis. A method for fast calculation of statistics and sorting of goods is proposed, which uses the Hoar sorting function (Quicksort), which allows in real time in a relatively short period of time to sort items by different characteristics.

A model of web system operation has been developed. The created web system was implemented in the online store "Working Bee".Web system architecture designed. An online store operating in accordance with the requirements was created and tested. Developed user manual.

The developed method and model can be used in the development of modern web systems, which provide sorting and work with large data sets.Keywords: Hoare sorting, data analysis, monitoring, web system.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасні люди користуються тими чи іншими веб-додатками. Розвиток веб-технологій не стоїть на місці, і з кожним днем потрібно удосконалювати технології, які використовуються. Зі збільшеним попиту на телефони та комп'ютери збільшується і попит та вимоги до сайтів та веб-систем, які там використовуються.

Зараз майже кожен підприємець переходить в онлайн, оскільки так можна охопити більшу аудиторію та налагодити логістичні системи підприємства. Якісно зроблений інтернет-магазин дає змогу ефективно презентувати товар і зменшити час для сортування та пошуку продуктів за визначеними потребами[1].

Інтернет-технології досить швидко розвиваються, тому кількість людей, які надають перевагу покупкам в Інтернеті, росте. Крім того, з популяризацією здорового харчування виріс попит на натуральні продукти природного походження [1], що обумовлює популяризацію меду як продукту здорового харчування.

Інтернет-магазини дають людям можливість знаходити кращі пропозиції за доступними цінами. Крім того, можна порівняти характеристики та провести сортування декількох товарів, між якими потрібно зробити вибір. Тому розробка веб-системи, яка спрямована на моніторинг якості меду та його продаж, є актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження.

Метою дослідження є підвищення ефективності роботи Інтернет-магазину з продажу меду шляхом розробки та використання моніторингової системи для продажу і тестування якості продукту, що дозволить

систематизувати товар та забезпечити високий рівень його якості.

Основними задачами дослідження є:

- аналіз поведінки користувача на сайті інтернет-магазину;
- розробка методу і моделі роботи веб системи;
- розробка креативного графічного оформлення інтернет-магазину;
- розробка інтуїтивно зрозумілого користувацького інтерфейсу;
- програмна реалізація інтернет-магазину «Трудяща бджілка» для продажу та моніторингу якості меду;
- тестування створеного веб ресурсу.

Об’єкт дослідження – процеси створення Інтернет-магазину з моніторинговою системою й тестуванням якості продукції.

Предметом дослідження є засоби реалізації інтернет-магазинів, орієнтованих на продаж і сортування товарів за ідентифікованими характеристиками.

Методи дослідження. У процесі розробки використовувалися:

- методи теорії алгоритмів для побудови алгоритмів сортування;
- методи створення графічних зображень для візуального оформлення інтернет-магазину;
- методи аналізу та синтезу для побудови ключових моделей моніторингової системи;
- методи мережевих технологій для забезпечення коректної роботи магазину онлайн;
- методи тестування для підтвердження працездатності системи та її відповідності заданим вимогам.

Новизна отриманих результатів.

1. Подальшого розвитку дістав метод аналізу та сортування товарів за станом їх якості, який, на відміну від існуючих, використовує варіантний аналіз асортименту у процесі формування рейтингової таблиці, що дозволяє контролювати та підвищувати якість продуктів.
2. Подальшого розвитку дістала модель роботи моніторингової

системи, яка, на відміну від існуючих, у режимі реального часу фіксує стан якості та рівень кристалізації товару, що дозволяє підвищити якість продукції й забезпечує ефективну роботу інтернет-магазину.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає у розробці моніторингої системі продукції інтернет-магазину, яку можна використовувати як готовий програмний продукт для продажу та моніторингу якості меду.

Достовірність теоретичних положень підтверджена результатами тестування розробленого інтернет-ресурсу.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці [2], опублікованій у співавторстві, автору належить розробка моделі моніторингової системи інтернет-магазину.

Апробація результатів роботи. Результати роботи доповідалися на Всеукраїнській науково-практичній інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп’ютерна інженерія.

Публікації. Основні результати досліджень опубліковані в науковій праці[2] у збірнику матеріалів конференції.

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та використано 19 літературних джерел.

1. ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

У час, коли телефон та комп’ютер є майже в кожного, та поширюється тенденція слідкуванням за своїм здоров’ям та якістю продуктів харчування, розширюються перспективи в інтернет-магазинів, які мають якісну продукцію, дають змогу моніторити їх якість й забезпечують користувачу можливість зручно і швидко придбати товар. Сучасні інтернет-магазини мають досить розширений функціонал у порівнянні зі звичайними сайтами. Вони включають в себе приємний та інтуїтивно зрозумілий інтерфейс, якісні графічні елементи та фото товарів, доступну систему каталогу товарів та коректну систему оплати і оформлення замовлення. Такі критерії описують якісний сайт з великим трафіком користувачів.

Насправді проблема класифікації сайтів розглядається не в одному, а відразу в кількох вимірах – функціональному та змістовному. Сайт, яким керує одна людина для поширення власних роздумів, — це індивідуальний блог. Але якщо один і той же продукт, навіть з однаковим дизайном, використовується для поширення, скажімо, новин, і наповнений групою професіоналів – це буде інформаційний ресурс.

1.1 Аналіз та класифікація веб продуктів.

Є досить багато різновидів веб продуктів. Це може бути відео хостинг, графічний редактор, онлайн агрегатор або сайт для створення тестування. Моніторингова система проводить аналіз характеристик товару, визначає якість продукту та формує для користувача рейтинг товарів за визначеними параметрами.

Від кількості товарів та функцій опублікування товарів в інтернет-магазині залежить його класифікація. Є звичайні інтернет-магазини, які може створити і використовувати індивідуальний користувач, а є онлайн агрегатори, які дають змогу продавати на своїй платформі товари інших продавців та

беруть невеликий відсоток з продажу або кошти за відображення карточок товарів. На таких онлайн агрегаторах працює швидка та продумана система, яка дає можливість порівнювати товари між собою та проаналізувати великі обсяги інформації за лічені секунди [3].

У наш час, коли відбувається масова діджиталізація в різних сферах діяльності, коли популярним є прагнення слідкувати за своїм здоров'я та споживати тільки натуральні продукти, зростають перспективи успішного розвитку веб-системи, яка надає змогу придбати якісну продукцію та здійснювати моніторинг якості товару з урахуванням часу додавання на сайт та відслідковування хімічного складову продуктів. Сучасні веб-системи мають широкий функціонал, якщо порівнювати їх зі звичайними сайтами. В них реалізований приємний та інтуїтивно зрозумілий інтерфейс, якісні графічні елементи та фото товарів, легка система каталогу та коректна система оплати й оформлення замовлення. Саме такі критерії описують якісний сайт з великим функціоналом та з великим трафіком користувачів.

За функціями сайти можна поділити на такі типи:

- сайти презентації;
- посадкові сайти;
- веб сервіси;
- інтернет-магазини;
- медіаресурси;
- соцмережі.

Ще є деякі додаткові типи веб-сайтів, зокрема:

- форуми;
- чати;
- портали.

Часто люди при слові «сайт» уявляють презентаційні сайти. Назва даного типу відповідає його функціоналу. Завдання таких сайтів — презентувати продукти компаній або самі компанії. Також можна сказати, що це сторінка

офіційного представництва бренду в інтернеті. Як правило, презентаційні сайти складаються з розділів по типу «Про компанію», «Послуги», «Продукти» та «Контакти». Якщо компанії мають напрям послуг пов’язаних з креативом, то може ще бути сторінка з портфоліо. До того ж, часто реалізовують новинну стручку зі всіма подіями які будуть корисні споживачу.

Є думка, що посадові сайти мають лиш одну сторінку, — це помилкове судження. Вони мають таку назву через те, що вони спрямовані на збір трафіку через рекламу. Можна сказати, що будь-яка сторінка, яка спрямована на збір трафіку та прив’язана до рекламного кабінету, автоматично вважається посадковим сайтом. Колись трафік, який генерувався рекламою, був спрямований на сторінку з каталогом продуктів компанії. З метою економії часу формуються шаблони сторінок продуктів. Це відбувається, оскільки структура є типовою, описи продуктів є типовими, загальними та беземоційними, тоді ефективність реклами буде низькою. Для підвищення ефективності продукти та послуги почали просувати окремими сторінками. Особиста сторінка дає змогу максимально індивідуально та якісно описати продукт для кращого просування.

Онлайн-сервіси [5] — це сайти, які дають змогу користувачам користуватись тими чи іншими онлайн послугами. Опис на такому сервісі є дещо загальним, але кількість послуг, які можуть бути реалізовані, досить велика, а тому самих онлайн сервісів може бути чимало. Наприклад, Gmail є онлайн сервісом для отримання та відправки листів електронної пошти. OllTV — онлайн сервіс для перегляду ліцензійного відеоконтенту. Всім відомий Google також онлайн-сервіс, який надає доступ з пошуку інформації. Для кращого розрізнення сервісів додають приставки «стрімінговий», «файлообмінний» або «платіжний».

Інтернет-магазин — це також онлайн сервіс, призначений для продажу послуг та товарів. Одні з перших інтернет-магазинів були створені ще у дев’яності роки. Тоді вони сприймались дуже дивно через невелику кількість аудиторії в інтернеті, і в споживача не було звички купувати товари онлайн.

Якщо поорівнювати великі книжкові магазини, які існували фізично, онлайн-книгарня Amazon у 1995 році, була як дрібненька крамниця.

Сьогодні Amazon — це один з найбільших онлайн агрегаторів, який продає тепер не тільки книги, а й купу інших товарів на будь-який запит. Таке ж можна розповісти про вітчизняний онлайн агрегатор Rozetka. Він також пройшов схожий шлях у розвитку зі звичайного магазину то найкрупнішого онлайн агрегатора в країні.

З точки зору зовнішнього вигляду в інтернет-магазині найважливішим є каталог товару, окремі сторінки з товарами та екрани, які пов’язані з покупками. Інтернет-магазин має багато нюансів при створені, тому фахівці, які створюють ці веб ресурси, як правило, працюють в подальшому виключно з ними.

Медіаресурси та блоги — це вебсайти, на яких публікуються матеріали інформаційного характеру. Зокрема, вони часто є цифровими представниками медіа, які існують в реальному житті. Багато газет, які публікували інформацію ще задовго до створення інтернету, пройшли шлях цифровізації всіх процесів. Таким чином, звичайні медіа з часом повністю перейшли до цифрових реалій [5].

З часом з’явилося таке явище як блогінг. Це онлайн-щоденник, де людина може розповідати про своє життя, збираючи читачів, які його життям цікавляться. Першим, хто почав вести блог, вважається Тім Бернерс Лі, засновник такої системи як інтернет [5].

Зараз межа між сучасними онлайн-медіа та блогами зникає, тому що все частіше блоги стають більш професійними, а ЗМІ починають часто публікувати інформацію з особистих блогів.

Соцмережі – це веб сайти, на яких можна створювати профілі для розміщення медіаресурсів і спілкування. Така концепція була темою довгих досліджень психологів та антропологів ще до того, як був створений інтернет. Дослідження базувались на системі взаємовідносин людей та їх груп.

Є думка, що першою соціальною мережею була Facebook, але це не так,

тому що за довго до нього існували такі мережі як GeoCities, Classmates і SixDegrees [5].

Також, крім соцмереж, є майданчики для поширення інформації за типом Instagram (став мережею №1 по поширенню світлин), YouTube (мережа №1 для поширення відео) Крім того, месенджери Telegram, Messanger та Viber також є месенджерами, але здебільшого поширюють саме текстову інформацію.

Було розглянуто та класифіковано основні види веб-ресурсів, які на даний момент існують. Було описано їх особливості і приклади таких ресурсів із інтернет-простору.

1.2 Порівняльний аналіз аналогів

Проведемо порівняльний аналіз інтернет-магазинів меду та продукції, яка з ним пов’язана.

MEDOVA DENASTIA – інтернет-магазин медової продукції [6]. На сайті подана різноманітна продукція. На рисунку 1.1 зображено зовнішній вигляд інтернет-магазину «MEDOVA DENASTIA».

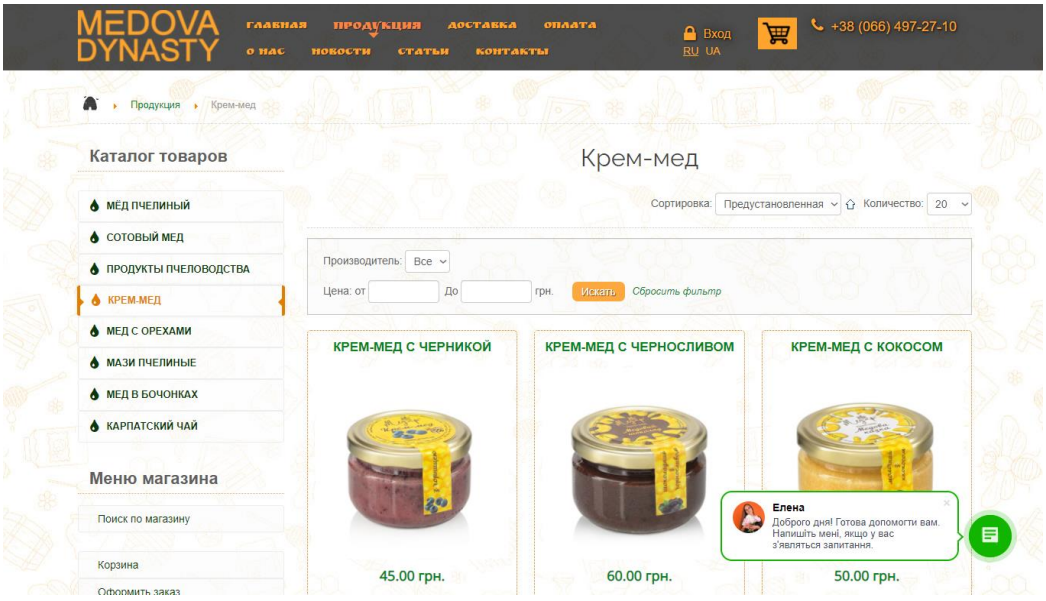

Рисунок 1.1 – Інтернет-магазин MEDOVA DENASTIA

ZolotaSota – інтернет-магазин який має в асортименті мед, крем-мед, віск, настоянки та інші продукти бджільництва. Каталог товарів сайту зображено на рисунку 1.2.

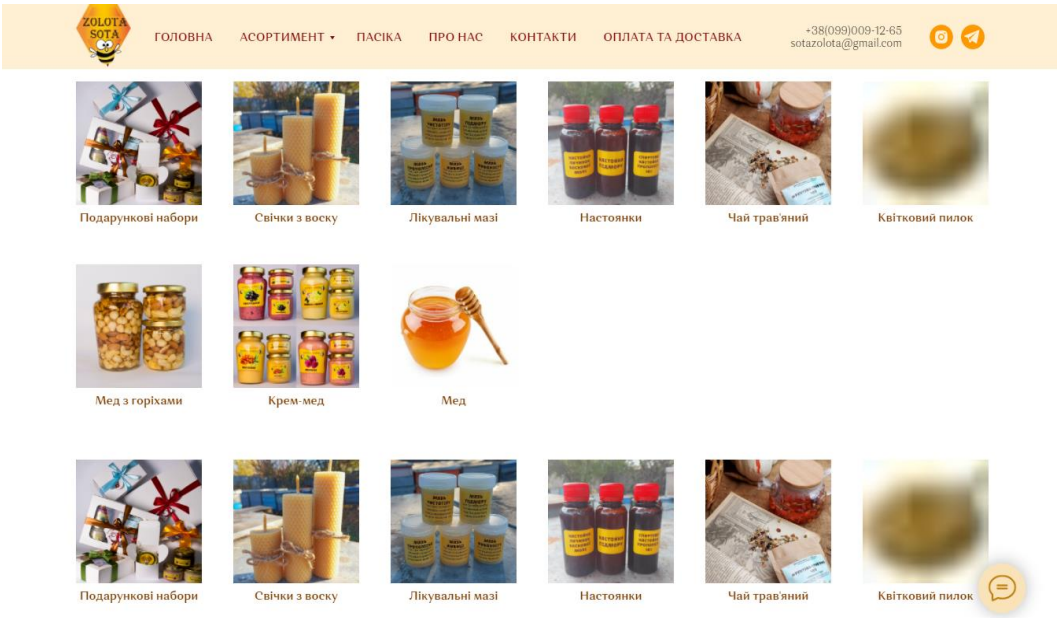

Рисунок 1.2 – Інтернет-магазин ZolotaSota

BeeLab – інтернет-магазин медової продукції, зокрема суміші меду та горіхів.

Зовнішній вигляд сайту BeeLab зображено на рисунку 1.3.

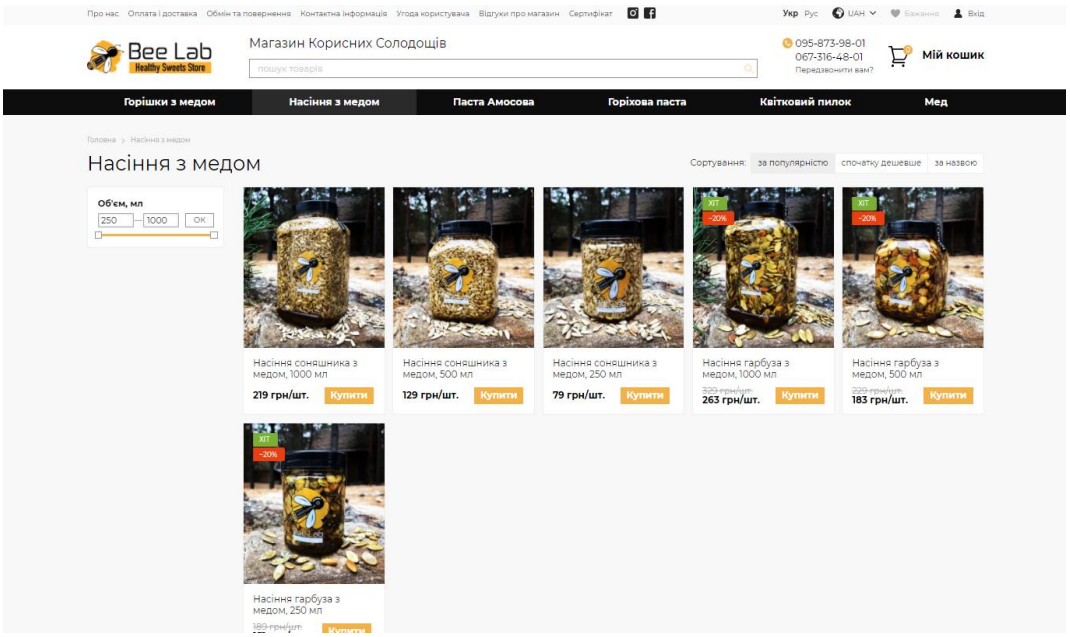

Рисунок 1.3 – Інтернет-магазин BeeLab

Еко МедОк – це сайт для збуту меду та продукції, що пов’язана з бджільництвом [7]. На рисунку 1.4 зображений каталог даного сайту.

Рисунок 1.4 – Інтернет-магазин Еко МедОк

Результати аналізу інтернет-магазинів, орієнтованих на продаж продуктів бджільництва, можна побачити у табл. 1.1, яка відображає головні характеристики розглянутих інтернет-ресурсів.

Таблиця 1.1 – Порівняння аналогів

Функції	MEDOVA DENASTIA	ZolotaSota	BeeLab	Еко МедОк	Трудяща бджілка
Зручний та інтуїтивний інтерфейс	+	-	+	+	+
Наповнення	-	-	+	-	+
Обширне	-	-	-	-	+

сортування					
Легка система оформлення замовлення	+	-	+	-	+
Адаптивність для мобільних платформ	-	-	+	-	+

Продовження таблиці 1.1

Отже, за табл. 1.1, інтернет-магазин MEDOVA DENASTIA надає можливість купити потрібний продукт, але має слабкий інтерфейс, малу кількість характеристик та налаштувань .

Інтернет-магазин MEDOVA DENASTIA надає можливість купити потрібний продукт, але має недосконалий інтерфейс, малу кількість характеристик та налаштувань .

Магазин Zolota Sota, крім малого функціоналу і поганого інтерфейсу, має слабке наповнення та нестабільну роботу.

Сайт для продажу медової продукції Bee Lab може конкурувати в плані інтерфейсу та наповнення. Дотримується загальний стиль та кольорова гама, але не контролює достатньо характеристик.

Інтернет платформа Еко МедОк має порівняно непоганий інтерфейс та більш широкий спектр фільтрації товару. Велика кількість характристик дає змогу вибрати продукти на будь який смак.

Відповідно до таблиці 1.1 обумовлюється перспективність створення веб ресурсу для аналізу якості та продажу меду, який матиме зручний і зрозумілий інтерфейс, гарний дизайн і широкий спектр аналізу та фільтрації товару за якістю.

1.3 Варіантний аналіз технологій та мов програмування

Мова розмітки гіпертексту HTML та мова стилів CSS є невід’ємною

частиною кожного сайту в мережі інтернет. А для бази даних та програмного функціоналу використовують JavaScript та PHP [8].

Мова розмітки гіперпосилань документів, іншими словами HTML. Була заснована на текстовій мові розміти SGLM, яка була створена для маркування документів, які містять деякий текст, зображення, гіперпосилання та інші графічні та функціональні елементи.

HTML розроблявся консорціумом W3C (остання версія 4.01). Є прогнози що незабаром XHTML займе нашу, так як є більш розширеною [8].

Cascading Style Sheets (скорочено CSS) – це мова стилів яка дає змогу описати вигляд документу. Дає змогу описати що де і як буде відображатись на тій чи інші сторінці[8].

Сторінки сайтів відображались не коректно на різних браузерах, так як в ті часи збереження стилю сайту було важким процесом, так як було мало контролю над відображенням контенту. Консорціуму W3C було запропоновано декілька варіантів таблиць стилів. Після тривалого обговорення було вибрано два з них, які лягли за основу сучасному CSS.

JavaScript – це мова програмування поведінки веб-сторінки. Хоча це не єдине застосування даної мови. На даний момент на платформі Node.js можна створювати продуктивні мережеві додатки. Такі бази даних як MongoDB та CouchDB використовують JS як мову програмування.

Брендон Айх в 1995 році створив Java Script, яка згодом в 1997 стала стандартом ECMA-262.

PHP – мова для скриптів, яка була створена для генерації сторінок зі сторони сервера. PHP це одна з найпоширеніших мов, які використовуються у сфері веб-розробок. Переважно саме вона підтримується хостинг-провадерами.

Розглянемо кожну мову зі сторони структури, функціоналу s особливостей і роботі.

HTML елементи, по суті, є будівельними блоками HTML сторінок. Мова HTML дає низку засобів для формування структурних документів, позначає семантику тексту (списки, посилання, абзаци, цитати, тощо). Самі ж елементи

описані тегами, які окреслені кутовими дужками.

HTML впровадив такі засоби як:

- позначення структурного складу тексту: абзацами, списками, таблицями та іншим;
- гіперпосилання, за допомогою яких відбувається отримання інформації в інтернеті;
- інтерактивні форми;
- включення до тексту таких елементів як відео, звук і зображення.

Використання CSS обумовлене описом шрифтів, кольорів, відступів та анімацій у верстці. Одна з особливостей це те що є можливість описувати все в окремому документі, що дає змогу краще сприймати код всієї сторінки. Крім того, це дає гнучкий контроль у відображені контенту для різних пристроїв.

Стандарти CSS визначають діапазон і порядок застосування стилів. Тобто застосовується каскадний принцип. Це коли елементам вказується лиш та інформація яка не визначена загальним стилем або змінилась.

Синтаксис CSS порівняно простий, так як використовується мало слів для назв складових стилю. В кожному стилі є список правил як можна використовувати. Правила містять в собі селектори (один або більше) та блок визначення. Сам ж блок складається зі списку властивостей, які оточені фігурними дужками/

Java Script використовується в:

- написанні сценаріїв для надання інтерактивності;
- створенні вебзастосунків;
- програмуванні сервера;
- створенні стаціонарних застосунків;
- мобільних застосунках;
- сценаріях прикладних програм.

Хоч і назва Java і Java Script схожі, але це кардинально різні мови. Вони мають різну семантику. Є лиш спільний синтаксис, так як походили вони з

мови C, але в більшій мірі вона є результатом впливу Self та Scheme.

Для кожного програміста PHP буде знайома, тому що мова взяла багато чого з C та Perl, це значно зменшує зусилля для вивчення. Хоч ця мова є молодою, але в нас час вона вже стала досить популярною серед веброзробників.

Дана підбірка є базовою не тільки для побудови інтернет магазинів, а й для створення сучасного сайту для великої трафікогенерації.

1.4 Обґрунтування вибору технологій та мов програмування

Щоб створити веб-додаток для моніторингу якості меду та його продажу, буде використано такі мови, як HTML (мова розмітки гіпертекстів), CSS (мова стилів документів), JavaScript (мова поведінки сайту) та PHP (скриптова мова програмування). Вказані мови задовольняють вимоги для створення інтернет-магазину з продажі меду, що підтримує моніторингову систему якості товару.

Середовищем для програмування було використано Brasket 2019[11].

Програма Brasket має велику популярність серед веб-розробників та верстальщиків, так як має підтримку великої кількості мов програмування, які дуже часто застосовуються в веб-розробці. Серед мов які підтримуються: HTML, CSS, Java, Java Script, C++, C#, SQL, PHP. Серед цих мов є ті які будуть використовуватись безпосередньо мною. Крім того, дана програма має зручний інтерфейс, гнучку взаємодією між файлами проекту, режим консолі та багато фішок які перекочували з текстових редакторів.

Отже, HTML, CSS, JS, PHP та середовище Brasket є ідеальною підбіркою для створення веб-додатку та інтернет-магазину «Трудяща бджілка».

1.5 Постановка задач роботи

Завданням бакалаврської дипломної роботи є створення веб-системи «Трудяща бджілка», орієнтованої моніторинг якості та продажу меду та продукції бджільництва.

Для злагодженої системи потрібно створити зручний та інтуїтивно зрозумілий інтерфейс для користувача. Потрібно розробити систему аналізу та моніторингу товару, а також забезпечити надійний спосіб оформлення покупок.

Необхідно розробити модель інтернет магазину «Трудяща бджілка», яка продемонструє весь функціонал та особливості даної веб-розробки.

Сайт та його система будуть розроблені на платформі Brasket для набуття навичок програмування та верстки.

Необхідно розробити систему моніторингу якості продуктів в залежності від характеристик та часу додавання продукту на сайт.

Серед графічних об’єктів потрібно підготувати зображення для оформлення та наповнення сайту.

Головними завданнями бакалаврської дипломної роботи є:

- аналіз поведінки користувача на сайті інтернет-магазину;
- розробка методу і моделі роботи веб системи;
- розробка креативного графічного оформлення інтернет-магазину;
- розробка інтуїтивно зрозумілого користувацького інтерфейсу;
- програмна реалізація інтернет-магазину «Трудяща бджілка» для продажу та моніторингу якості меду;
- тестування створеного веб ресурсу.

Отже, «Трудяща бджілка» – це інтернет-магазин з функцією глибокої оцінки та моніторингу якості товару, що дає змогу вибрати якісний та потрібний товар.

Для коректного відображення сайту необхідні такі характеристики:

- комп’ютер серії IBM PC з частотою 233 МГц і вище;
- 64МБ оперативної пам’яті;
- графічний адаптер SVGA (Super Video Graphic Adapter);
- відеокарта об’ємом пам’яті не менше 4МБ;
- клавіатура та комп’ютерна миша;

- ОС Windows Vista/7/XP;

Розмір дискового простору, що займає сайт: 15 300 000 байт (враховуючи всі графічні зображення). Розмір оперативної пам’яті, що займає програма: 1 500 КБайт.

1.6 Висновки

У першому розділі було проведено аналіз предметної області, розглянуто особливості створення інтернет-магазинів. Поставлено задачу розробити веб-систему «Трудяща бджілка», орієнтовану на моніторинг, аналіз якості меду і його продаж.

Для вирішення поставленої задачі необхідно удосконалити систему фільтрації та сортування сайту для обробки великої кількості даних.

Необхідно розробити модель веб-системи, моніторинг та аналіз якості продуктів. Розроблені метод, алгоритми, модель слід реалізувати у демонстраційному інтернет-магазині «Трудяща бджілка».

В якості мов програмування було обрано HTML, CSS, JS і PHP, а середовищем програмування – інструмент для веб-розробки Brasket.

2. РОЗРОБКА МЕТОДУ І МОДЕЛІ РОБОТИ ВЕБ-СИСТЕМИ

2.1 Аналіз завдання

Розробка бакалаврської дипломної роботи починається з визначення технічного завдання, тобто потрібно чітко сформулювати всі вимоги, які потрібні для створення веб-системи для продажу та моніторингу якості меду. Потрібно сформулювати вимоги до програмного наповнення, інтерфейсу, функціональні можливості та загальні принципи формування фінального продукту.

Інтернет-магазин «Трудяща бджілка» призначений для збуду медової продукції та глибокого моніторингу всіх характеристик для аналізу його якості.

Для розробки необхідно:

- провести аналіз поведінки користувача на сайті інтернет-магазину;
- розробити метод і модель роботи веб системи;
- розробити креативне графічне оформлення інтернет-магазину;
- розробити інтуїтивно зрозумілий користувацький інтерфейс;
- програмно реалізувати інтернет-магазин «Трудяща бджілка» для продажу та моніторингу якості меду;
- провести тестування створеного веб ресурсу.

.

2.2 Розробка стратегії для воронки продажу

Воронки продажу інтернет-магазину – це шлях користувача по сайту від моменту першого переходу до моменту здійснення покупки [12].

Воронка продажу необхідна як власникам бізнесу, і маркетологам – всім, хто хоче заробляти на своєму сайту та оптимізувати витрати на його просування.

Дана модель нагадує ворону через те що на різних етапах відсіюється якась кількість людей, і в результаті залишаються лиш ті, з якими компанія зробила продаж продукції [12].

Воронка інтернет-продажів ділиться на декілька рівнів аудиторії:

- цільова – люди будуть користуватись продуктом компанії;
- перспективна – частина цільової аудиторії, яка побачить сайт, тому всі зусилля по просуванню повинні відповідати запитам користувачів, щоб кошти на рекламу не бути потрачені даремно;
- зацікавлена – перспективні користувачі сайту, щоб покращити продажі в цій категорії, потрібно оптимізувати CRM та покращувати дизайн;
- потенційна – користувачі, які вивчили сайт і проявили цікавість до товарів;
- дійсна – покупці, що оформили замовлення і отримали замовлення;
- повторна – люди, які постійно взаємодіють з компанією та регулярно проходять етапи воронки.

Як правило, воронка сайту будується за класичною маркетинговою моделлю AIDA. Вона складається з такий етапів:

- увага;
- цікавість;
- бажання;
- дія.

Ціль аналізу воронки полягає в тому, щоб оцінити ефективність всіх етапів. Для цього необхідно зібрати кількісні дані в розрізі різних цільових груп, продуктів, каналів просування і так далі.

Портрет клієнта – усереднений образ клієнта з ключовими характеристиками. Щоб описати коректний образ, потрібно сформувати базові запитання: «Кого цікавить товар компанії?», «Де може перебувати покупець?», «Яку потребу вирішує покупець своєю покупкою?»

При формуванні портрета клієнта не можна брати інформацію з голови. Те, як компанія бачить свою аудиторію, може не співпадати з тим, що є насправді. Потрібно зібрати найлояльніших клієнтів і провести з ними невелике опитування. Якщо продажів ще не було, то можна опитати потенційних клієнтів: створити опитування в соціальних мережах і прорекламувати його.

Існує така теорія як «Сходинки усвідомленості Бена Ханта»: кожен клієнт знаходиться на деякому етапі знайомства з товаром чи послугою. За допомогою цієї теорії маркетологи можуть проробити деякі етапи взаємодії з потенційними клієнтами.

На першому етапі «сходинок Ханта» майбутній клієнт не ознайомлений з компанією і товаром. Головною задачею на цьому етапі – сформувати розуміння аудиторії про проблеми які вирішує та чи інша послуга.

Другий етап, користувач розуміє свою проблему і тому шукає способи їх вирішення в інтернеті для їх вирішення.

На третьому етапі здійснюється порівняння користувачем. Він порівнює варіанти вирішення проблеми: який буде найвигідніший саме йому.

Четвертий етап – час порівнювання декількох пропозицій на ринку. Які є переваги: ціна, якість, сервіс, швидкість доставки.

Під час фінального етапу клієнт оплачує вибраний товар/послугу.

Розглянемо, як може працювати воронка продажу в інтернет-магазині «Трудяща бджілка». Людина бачить контексну рекламу і згадує, що їй потрібна продукція, пов’язана з медом. Користувач переходить за посиланням та переглядає товари, знаходить потрібні товари і додає їх в козину. Відволікається на інші справи і закриває сайт. Покупка не є пріоритетною, тому покупець потенційно забуває про неї.

Згодом від бачить рекламу в соціальних мережах і згадує що переглядав товар на сайті. Через деякий час, клієнт бачить рекламу зі знижкою на цю позицію, повертається та оформлює замовлення. Після чого отримує постійну розсилку з вигідними пропозиціями магазину.

2.3 Розробка методу аналізу та сортування даних

При створені інтернет-магазину важливо, щоб користувач швидко отримував інформацію. Продуктивність сайту на пряму впливає на його ефективність, особливо, коли мова йде про обробку та аналіз великої кількості даних [14].

У наш час користувачі дуже вибагливі в плані часу. Якщо сайт завантажується довго, то є велика вірогідність того, що користувач не дочекається завантаження і перейде на інший портал. У випадку формування таблиці в режимі реального часу, потрібно забезпечити швидку підбірку даних. Затримка не буде помічатись, якщо товар має 10-15 характеристик, які не прив’язані до часу і не потребують постійного оновлення.

Використовуючи дані про хімічний склад, який на пряму пливає на якість продукції, будуть враховуватись такі параметри, як глюкоза, фруктоза, сахароза, зольні елементи та вода. В залежності від цих параметрів, буде враховуватись коефіцієнт якості.

Також, в залежності від часу збору визначається коефіцієнт кристалізації. Хоча кристалізація меду не впливає на його якість, але іноді користувач не до кінця розуміє, що це природній процес і вважає, що він визначається зайвим вмістом цукру. Тому на базі розрахунків, буде визначатися ступінь кристалізації. Все рахується на середніх значеннях, але якщо хімічний склад відходить від норми, то загальний час буде множитись на відсоток відходу від норми.

В залежності від сорту меду відбуваються різні темпи кристалізації. Наприклад, соняшниковий мед може кристалізуватися за декілька тижнів. Акацієвий мед може не кристалізовуватися декілька років. Але от гречаний або липовий мед в середньому темпі кристалізується.

Норма хімічного складі становить 21.7% - 53.9% фруктози, 20.4% - 44.4 % глюкози, 15% - 20% вода, а все інше це природні домішки.

Метод сортування та обробки якості та засахареності товарів містить таку послідовність операцій:

1. Для отримання більш детальної інформації про якість продуктів користувач переходить на сторінку «Моніторинг якості».
2. Відбувається аналіз та перерозподіл всіх товарів функцією SortGoods, яка аналізує всі наявні товари та перерозподіляє їх якості.
3. Згодом викликається функція SugarcContent, яка визначає рівень засахареності меду в залежності від його виду, часу перебування на складі та рівня цукрів та води в складі
4. Після аналізу формується таблиця, яка ілюструє всю необхідну інформацію. Користувач має змогу перейти до довару, який сподобався, та здійснити покупку.

2.4 Розробка моделі веб-системи

Модель реалізації інтернет-магазину «Трудяща Бджілка» візуалізує карту користувача, що завантажує сайт і починає шукати потрібну йому продукцію. Користувача зустрічає «Головна сторінка», яка має загальну інформацію про сайт, гіперпосилання на особливі пропозиції та рекомендації.

В компоненті «Меню» є можливість переміщуватись по всіх сторінках сайту. Серед них «Каталог продукції», «Моніторинг якості», «Блог» та «Контакти».

Натиснувши на «Каталог продукції», переходимо в таблицю з картками товарів та бічною панеллю для сортування і фільтрації товарів.

На сторінці «Моніторинг якості» бачимо таблицю товарів за рейтингом, який постійно оновлюється автоматично. Наприклад, вибравши потрібну

категорію, можна побачити позиції, які максимально якісні з точки зору збору та хімічного складу.

«Блог» містить в собі різні статті (для генерації трафіку) і новини про роботу магазину.

На сторінці «Контакти» зібрана інформація для зворотного зв’язку.

На рисунку 2.1 наведено модель реалізації веб-системи «Трудяща бджілка».

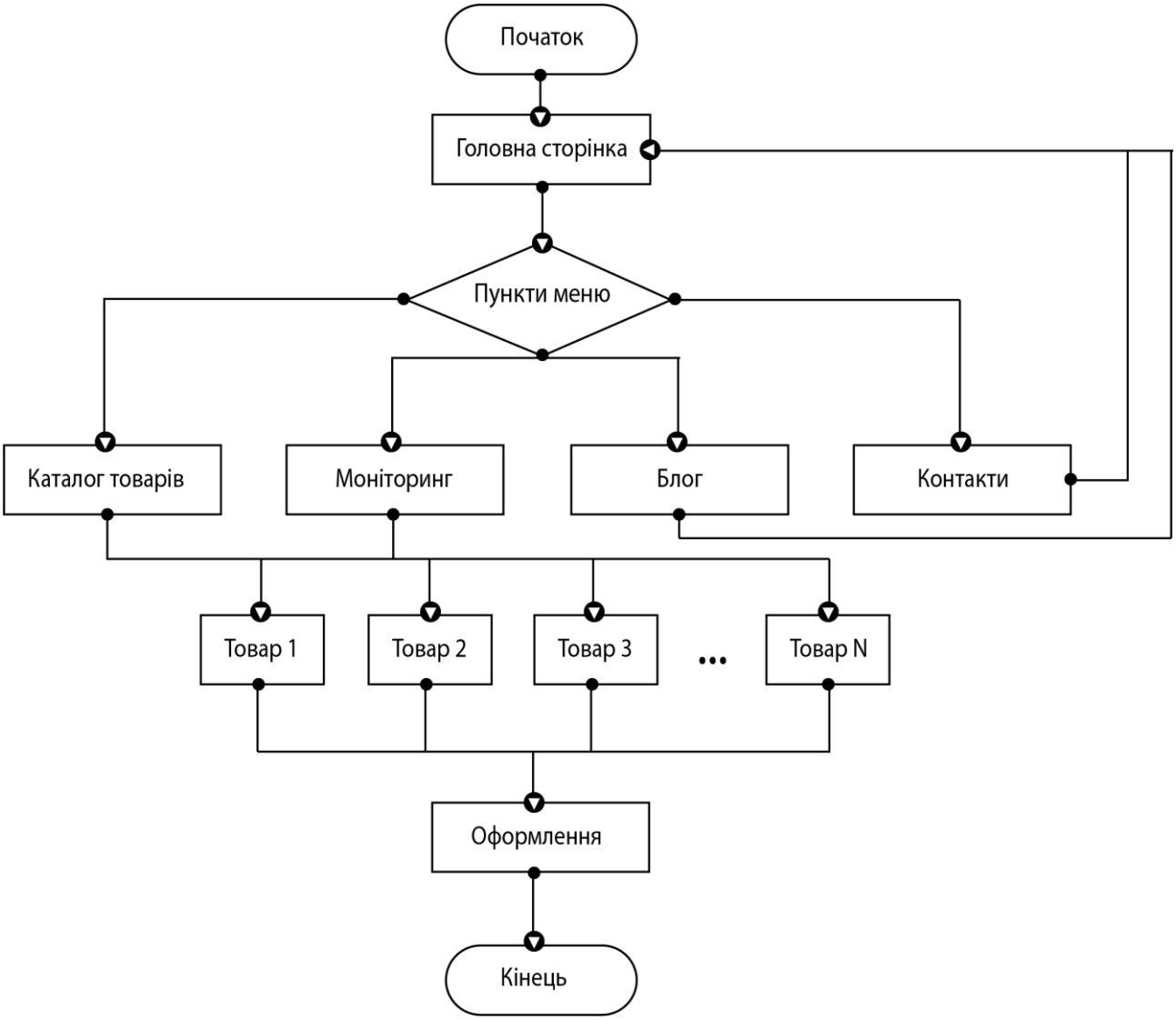

Рисунок 2.1 – Модель реалізації інтернет-магазину «Трудяща бджілка»

Розглянемо систему фільтрації і сортування на сторінці «Каталог товарів». Після завантаження сторінки відображається панель з критеріями

сортування та таблицею з карточок товарів. Користувач має змогу вибрати категорію продукції, її сорт, ціну, грамовки та матеріал упаковки. Фільтрація автоматично відбувається за якістю (яка визначається алгоритмом зі сторінки «Моніторинг якості»), але є можливість замінити цей пункт на фільтрацію за ціною.

На рисунку 2.2 наведено модель реалізації системи фільтрації та сортування.

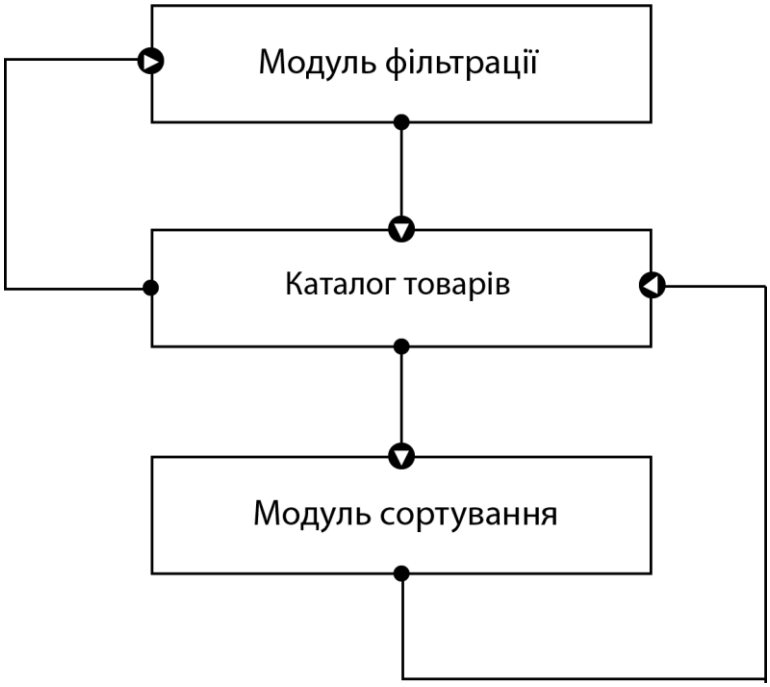

Рисунок 2.2 – Модель реалізації сортування і фільтрації

2.5 Висновки

У другому розділі розглянуто стратегії для воронки продажу інтернет-магазину.

Удосконалено метод аналізу якості меду на основі його характеристик, хімічного складу та часу збору.

Подальшого розвитку дістали моделі реалізації глибокої фільтрації та сортування товарів в каталозі.

Подальшого розвитку дістали моделі реалізації сортування та фільтрації

меду за заданими характеристиками в каталозі.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СИСТЕМИ

3.1 Проєктування інтерфейсу інтернет-магазину

Проєктуючи інтерфейс, потрібно враховувати, щоб навіть недосвідчений користувач міг розібратися та знайти потрібну позицію.

Сайт розраховується як на мобільні платформи, так і для комп'ютера. Тому відображення не суттєво зміниться в залежності від того, який розмір екрану у користувача.[15].

Для проєктування інтерфейсу веб-системи складено список необхідних сторінок, на яких буде відображено контент:

1. Сторінка «Головна», що складається з навігації, головного банера, інфографіки та цікавих пропозиції.
2. Сторінка «Каталог товару», на якій є головна навігація, панель для сортування та фільтрації і самі картки товару.
3. Сторінка «Моніторинг якості» з навігацією та рейтинг-таблицею товарів.
4. Сторінка «Блог», в якій, знову ж такий, є навігація та відображені новини компанії.
5. Сторінка «Контакти», де є навігація та основна інформація для зворотного зв'язку.

На основі висунутих вимог розроблено схематичні зображення інтерфейсу інтернет магазину.

Схему сторінки «Головна» наведено на рисунку 3.1.

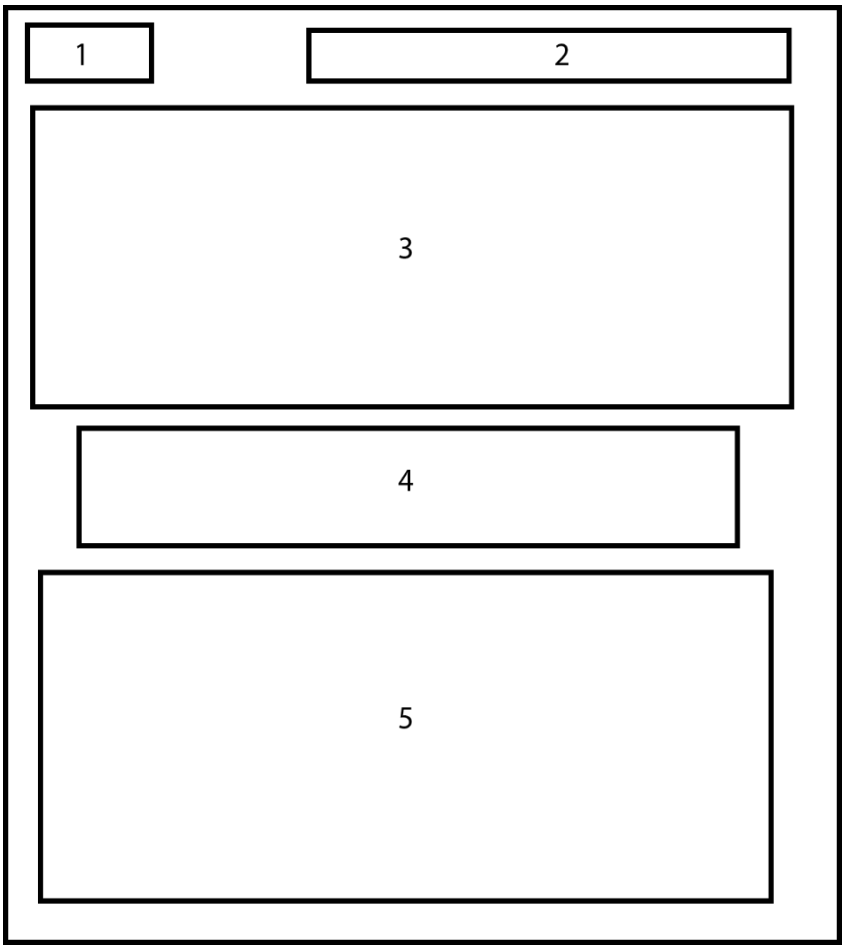

Рисунок 3.1 – Схема сторінки «Головна»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.1:

1. Логотип, який відображає назву та головного маскота «Трудяща бджілка».
2. Елементи головного меню для навігації по сайту.
3. Банер з найактуальнішими новинами.
4. Інфографіка та фото (мед, пасіка, бджоляр).
5. Підбір новинок та хітів продаж.

Сторінка «Каталог товару» дає можливість відфільтрувати та відсортувати товари по різним характеристикам. Схему цієї сторінки наведено на рисунку 3.2.

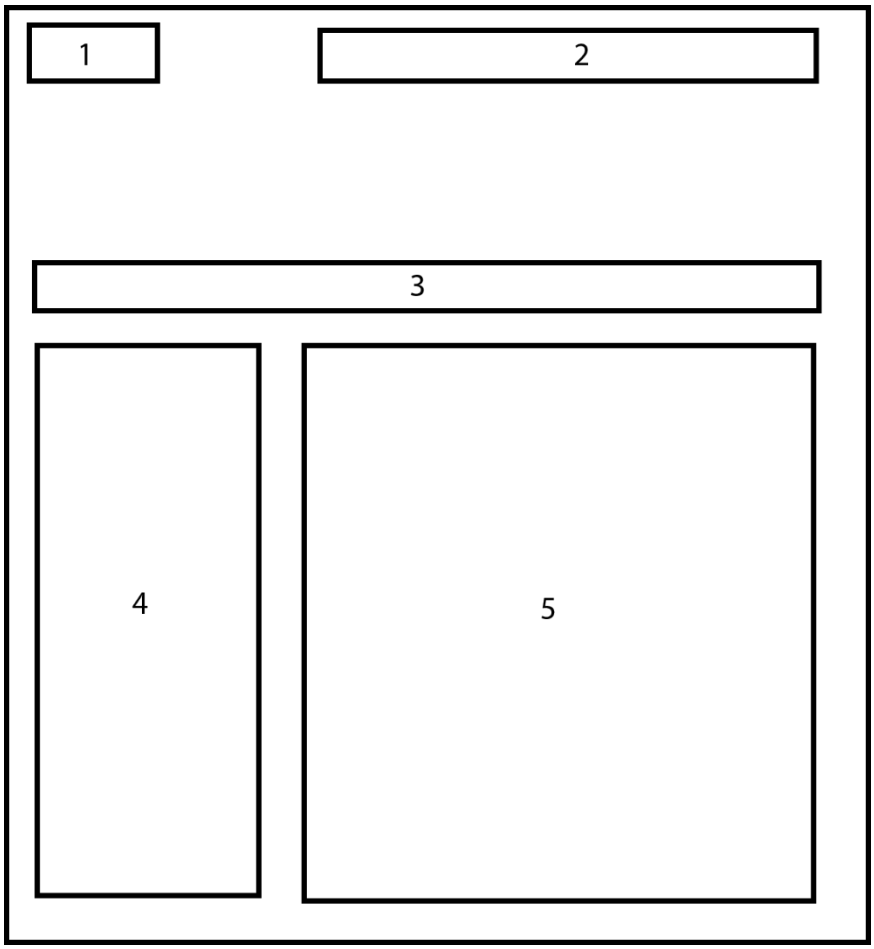

Рисунок 3.2 – Схема сторінки «Каталог товарів»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.2:

1. Логотип компанії.
2. Навігація сайту.
3. Панель для фільтрації товару.
4. Панель для сортування товару за ціною, видом продукції та вагою.
5. Відсортовані та відфільтровані карточки товару.

Сторінка «Моніторинг якості» дає розгорнуту інформацію про хімічний склад товарів у розрізі та показує ступінь кристалізації в залежності від часу зберігання. Схему цієї сторінки наведено на рисунку 3.3.

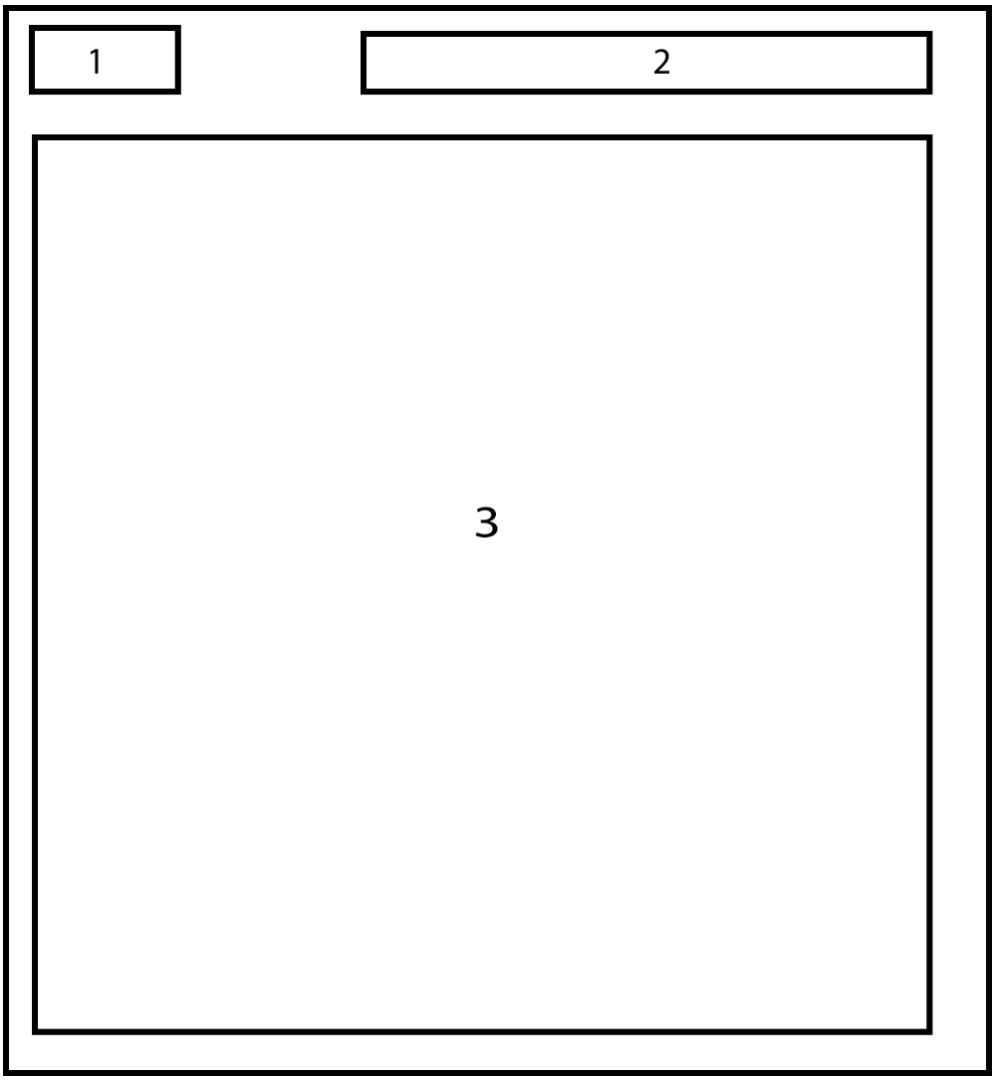

Рисунок 3.3 – Схема сторінки «Моніторинг якості»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.3:

1. Логотип компанії.
2. Навігація по сайту.
3. Таблиця з інформацією про товари з найкращою якістю.

Сторінка «Блог» відображає статті та новини сайту. Схему наведено на рисунку 3.4.

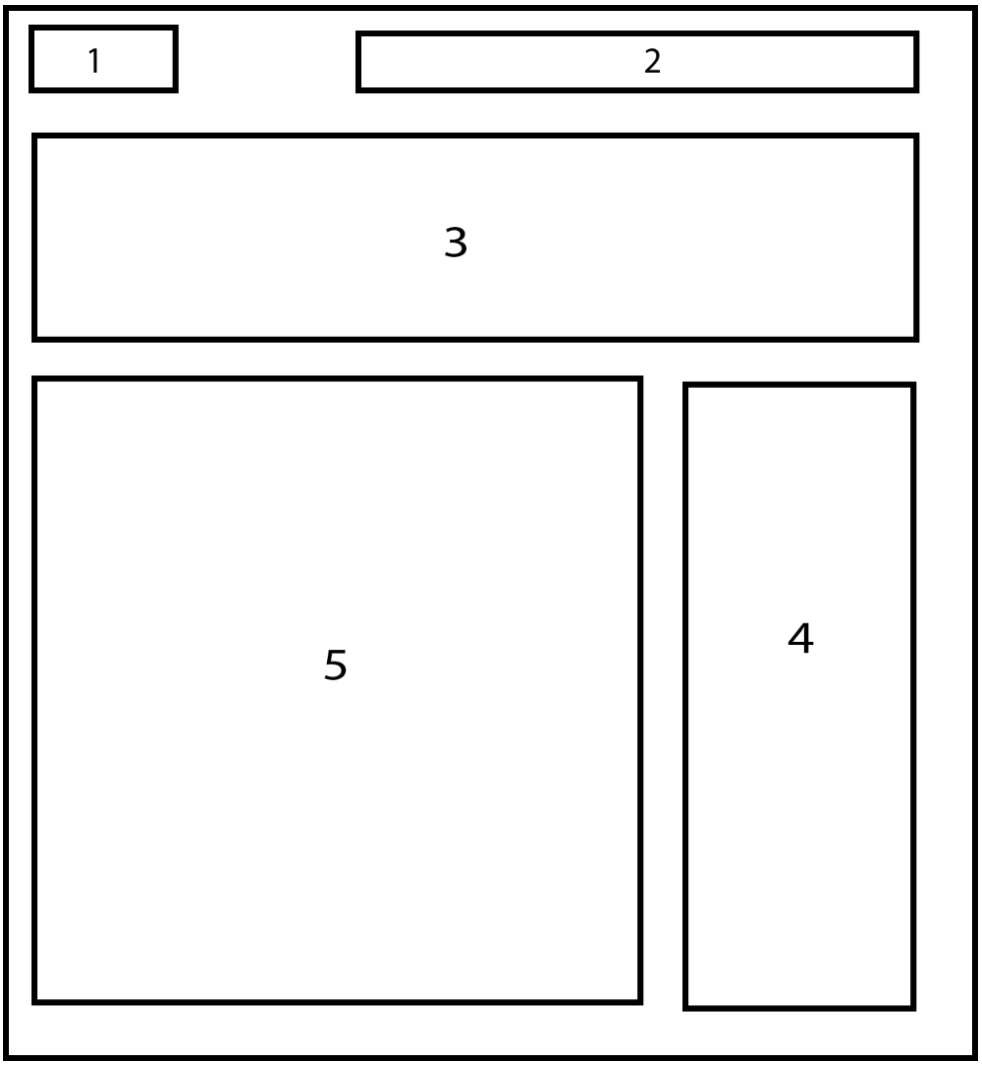

Рисунок 3.4 – Схема сторінки «Блог»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.4:

1. Логотип.
2. Навігація
3. Категорії новин та статей.
4. Коротка інформація про блог.
5. Статті.

Схему сторінки «Контакти» наведено на рисунку 3.5.

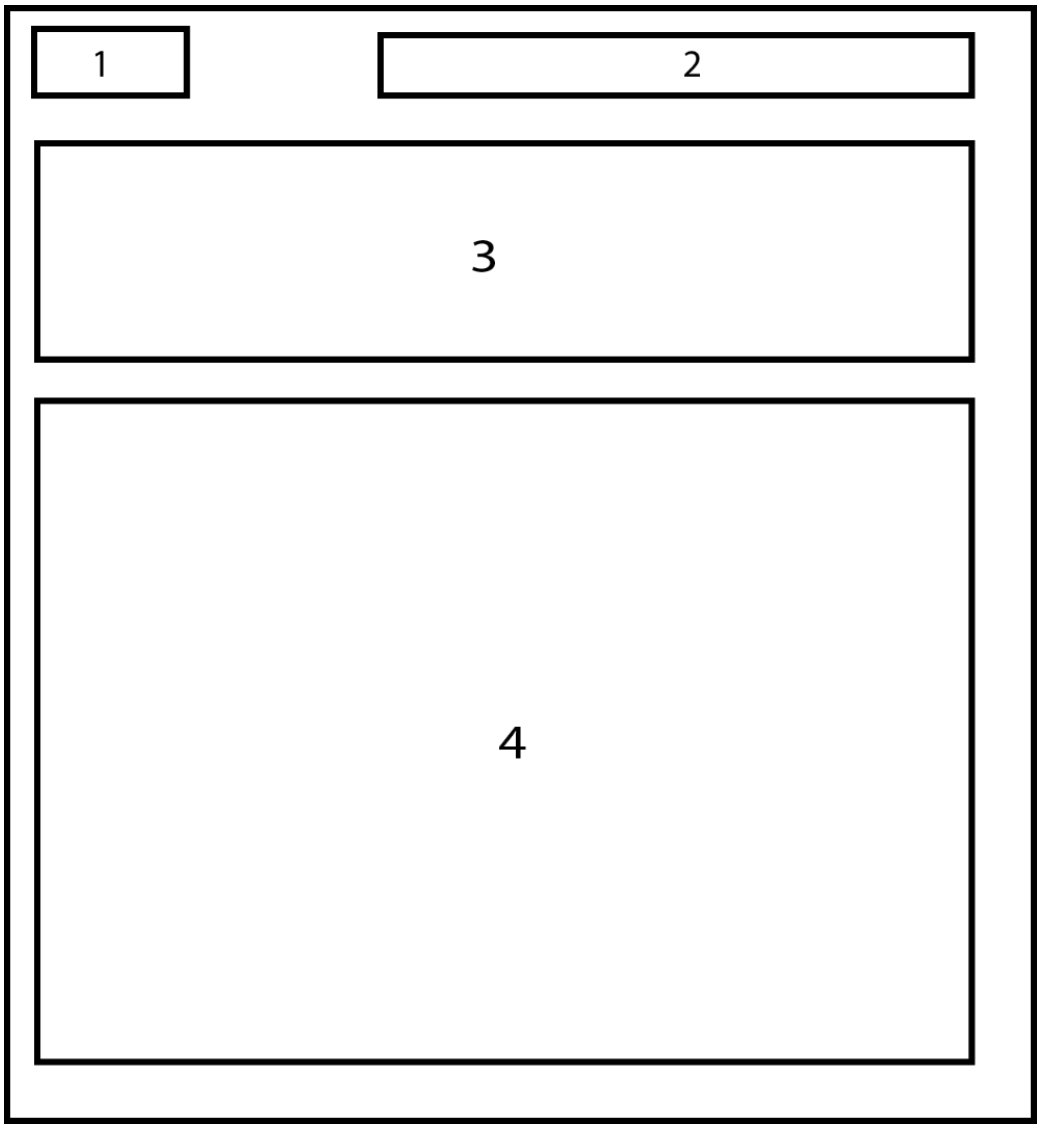

Рисунок 3.5 – Схема сторінки «Контакти»

Опис елементів схеми відповідно до їх нумерації на рисунку 3.5:

1. Логотип.
2. Навігація.
3. Номер телефону, пошта і інші контактна інформація.
4. Карта з місце знаходженням компанії

Також було створено структуру окремої сторінки для карток товару. Схему сторінки яка відображає картки товарів наведено на рисунку 3.6.

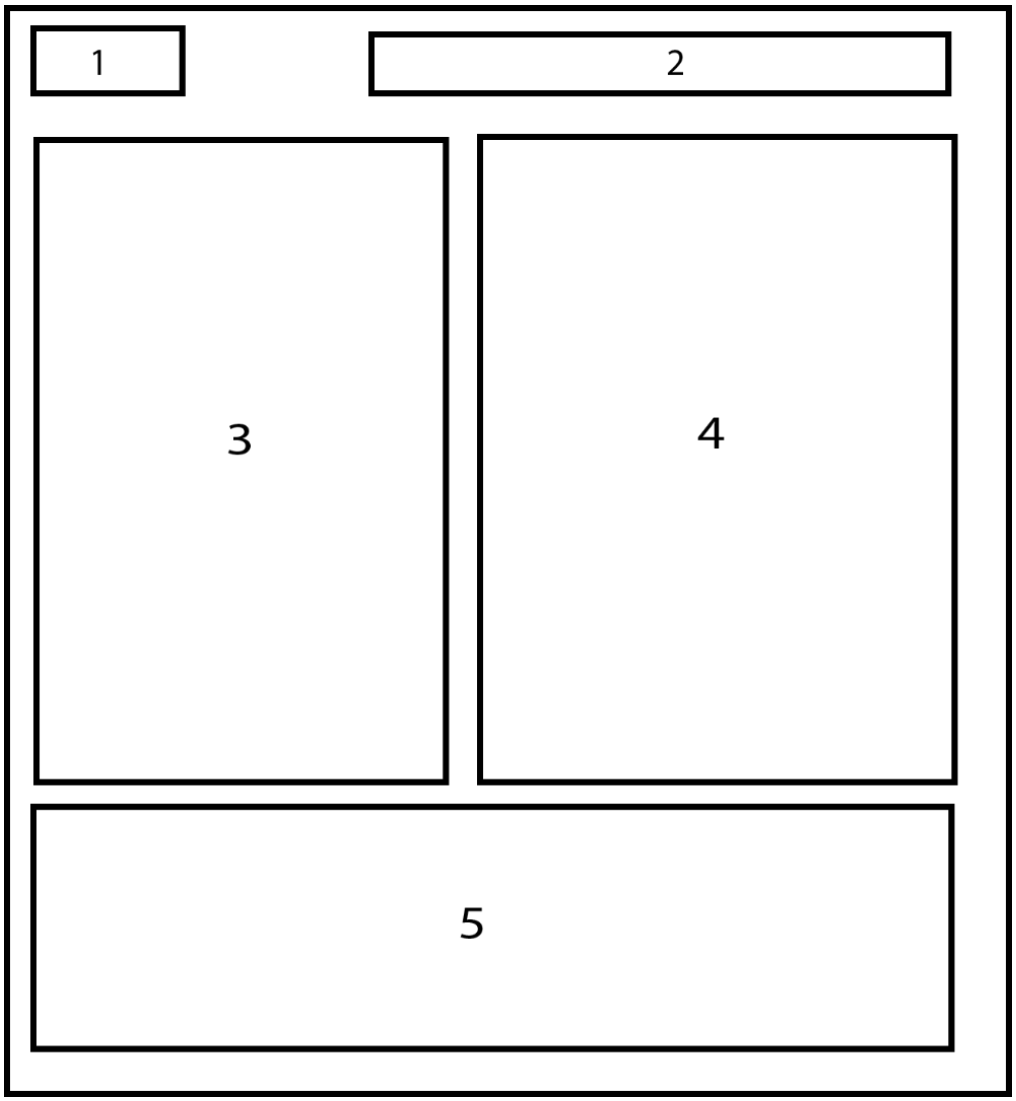

Рисунок 3.6 – Схема сторінки з карткою товару.

Опис елементів схеми відповідно до їх нумерації на рисунку 3.6:

1. Лого.
2. Навігації.
3. Фото товару
4. Назва, опис та кнопка «Купити»
5. Відгуки користувачів

Розроблений користувацький інтерфейс є макетом і має бути наповненим графічними матеріалами для забезпечення зручності у використанні та підвищення рівня зрозумілості користувачу.

3.2 Розробка графічних матеріалів

Однією з основних частин розробки веб-систем є дизайн та графічне наповнення. Кожен відвідувач сайту, в першу чергу звертає увагу на те, як той чи інший веб ресурс зображає і представляє матеріал, яким хоче залучити споживача до покупки.

Вдало підібраний дизайн може на пряму маніпулювати користувачем для виконання потрібної для підприємця дії. Влучно розставлені елементи, їх кольори та загалом дизайн відіграє ключову роль в трафікогенерації сайту.

Графічний редактор — це програма, яка наділена необхідними інструментами для створення та редагування графічного зображення, а також його експортування в потрібний для користувача формат.

Figma — графічний редактор для створення та редагування графічних креативів у векторній і растровій графіці. Крім того, доступна функція прототипування, яка дає змогу зробити проєкт з анімаціями та переходом по посиланям [16].

Розробляючи макет сторінки, потрібно дотримуватись установлених норм для оптимізації процесу. Отож першим етапом потрібно створити ескізи, які в подальшому будуть розглядатись керівництвом. Це потрібно для того, щоб не робити зайву роботу і мінімізувати затрати часу на виробництві. Лиш тільки після затвердження ескізу можна приступати до повноцінної роботи

Як правило, кількість схвалених варіантів значно менша за ту, яка надавалася на початку.

Після того, як затверджено кінцевий варіант, відбувається розробка макету сторінки. В даному процесі вже задіяні такі програми, як Figma або Sketch для зручного і інформативного подання моделі сайту. Розглянемо процес розробки на прикладі графічного зображення сторінки «Головна».

Розробка сторінки складається з побудови векторних, растрових та текстових об’єктів. В першу чергу було створено графічне представлення головного меню та скроллера з головними пропозиціями.

Створення макету супроводжується поєднанням текст, векторних та растрових зображень. Зокрема зображення кнопок або якихось декоративних елементів відбувається поєднанням тексту та геометричної фігури.

На рисунку 3.7 наведено візуалізацію процесу створеної частини зі скроллером та головним меню.

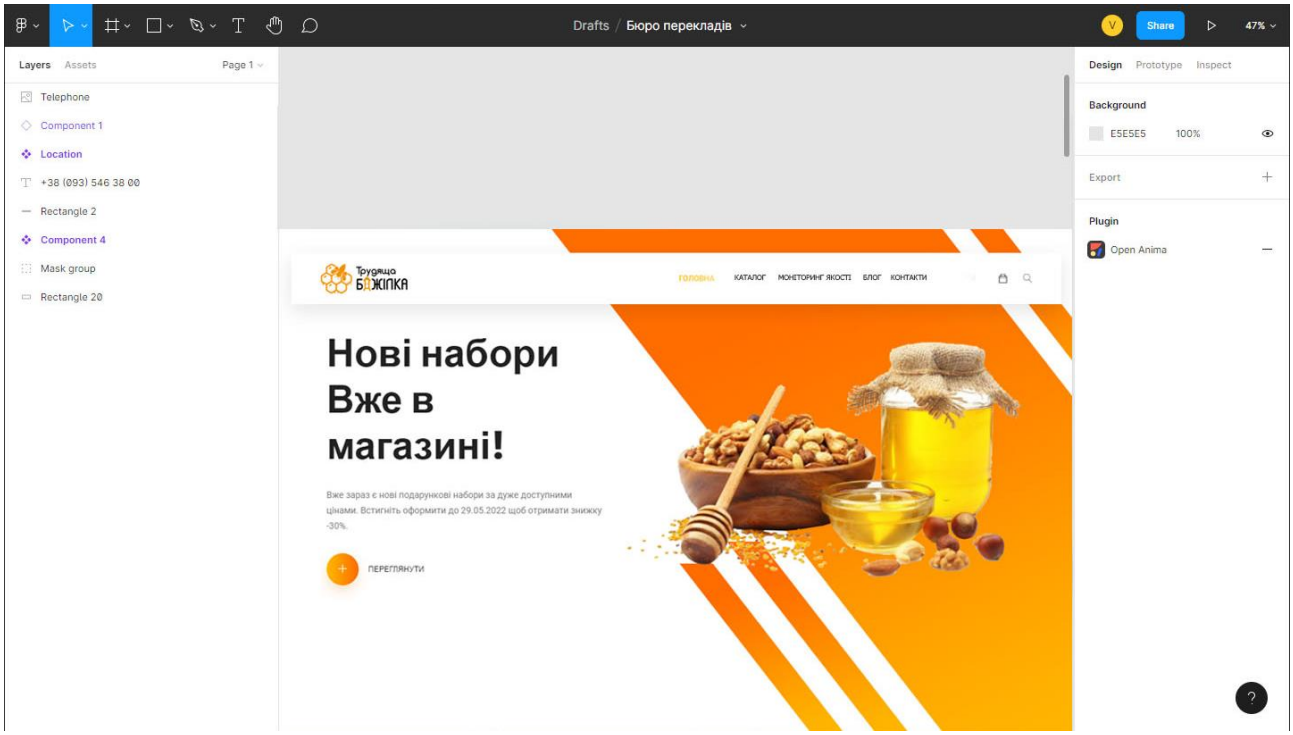

Рисунок 3.7 – Розроблене головне меню, логотип, та скроллер.

Крім того, при створені макету в Figma, є можливість розробити та продумати анімації перехотів та появи різних елементів, що дає змогу краще зрозуміти суть проекту майбутньому розробнику.

Використовуючи поєднання простих фігур, для сторінки «Головна» було створено векторні об’єкти, які зображують категорії товарів, які є в асортименті.

На рисунку 3.8 наведено вигляд векторних зображень «Категорії товарів».

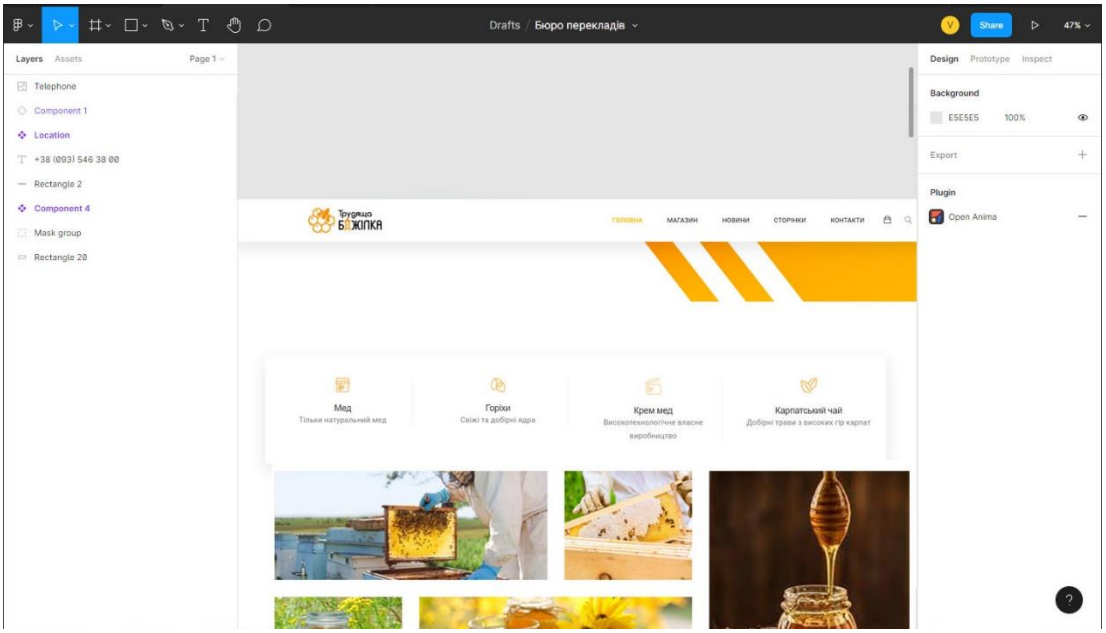

Рисунок 3.8 – Графічні елементи– «Категорії товарів».

Наступний блок був створений через поєднання відрендерених 3D моделей (фото товару), тексту і векторної геометричних фігур. На рисунку 3.9 наведено вигляд створеного графічного блоку «Переглянуті».

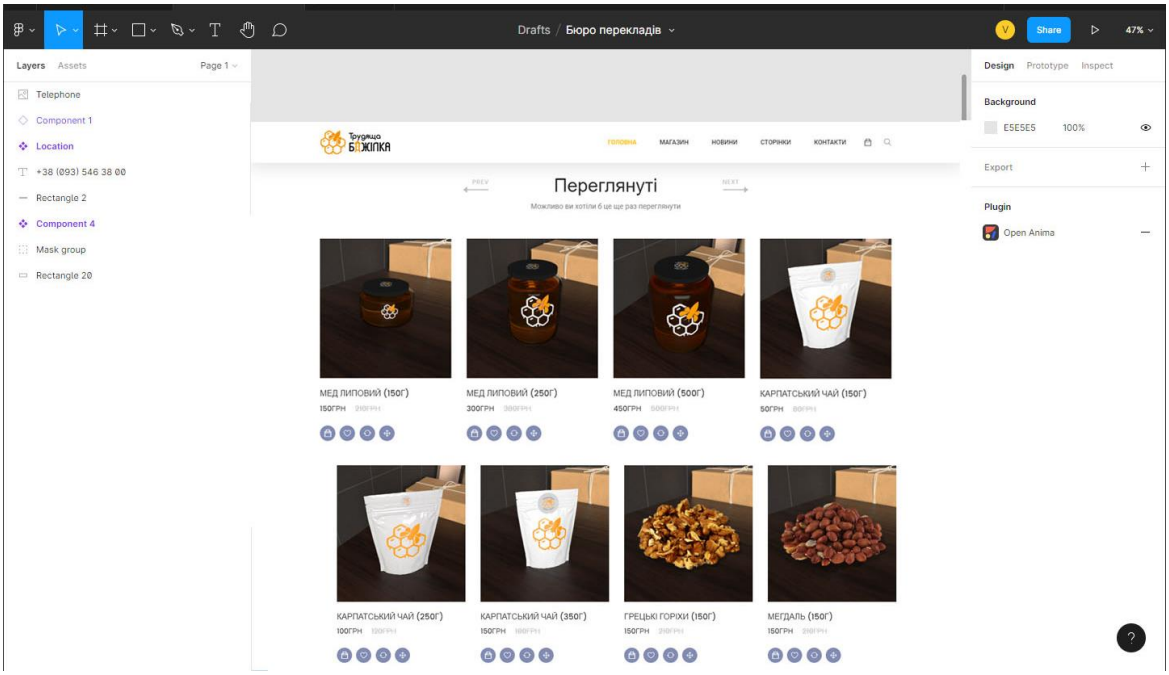

Рисунок 3.9 – Графічний блок «Переглянуті»

Також, було створено блок з таймером та ще одним скроллером для пропозицій пов’язаних з наборами.

На рисунку 3.10 наведено вигляд даного блок.

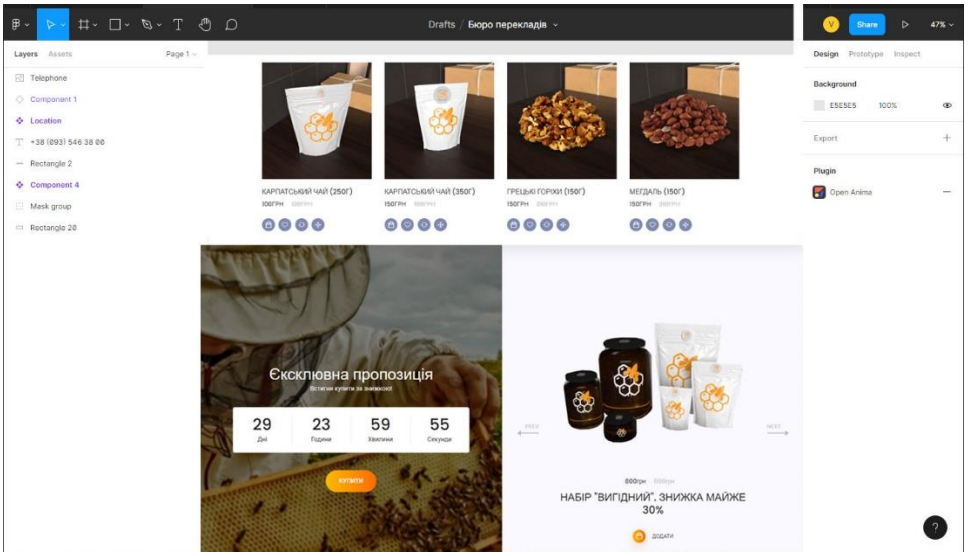

Рисунок 3.10 – Блок з таймером та скроллером.

Було створено ще 2 блок. Зокрема, блок з інфографікою брендів, пропозицій товарів та футер. Поєднавши всі блоки ми отримаємо макет сторінки «Головна» Вигляд поєднаних блоків наведено на рисунку 3.11.

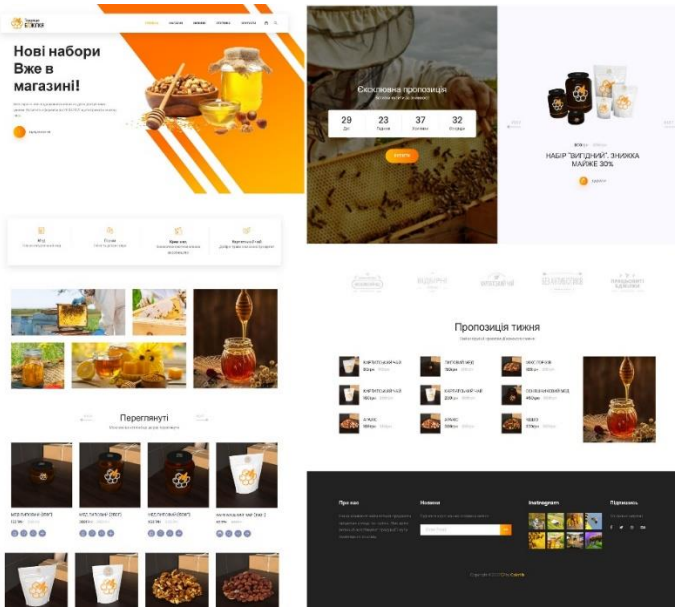

Рисунок 3.11 – Поєднання розроблених блоків на сторінці «Головна»

Після цього все потрібно експортувати для подальшої верстки. Для Bracket це графічні формати PNG, BMP, TIF, TGA, JPG.

Формат PNG дозволяє зберегти прозорість, тому саме він буде використаний у рамках формату для експорту елементів для веб-системи «Трудяща бджілка» [17].

3.3 Розробка основних модулів додатку

Для реалізації багатьох функцій сайту було використано Java Script. Зокрема бібліотеку JQuery яка є найпопулярнішою бібліотекою з відкритим кодом.

При завантажений сайт зустрічає скроллер/банер з найвигіднішими пропозиціями.

Частина коду реалізації даного елемента наведена на рисунку 3.12.

```
$(".active-banner-slider").owlCarousel({
    items:1,
    autoplay:false,
    autoplayTimeout: 5000,
    loop:true,
    nav:true,
    navText:["<img src='img/banner/prev.png'>","<img src='img/banner/next.png'>"],
    dots:false
});
```

Рисунок 3.12 – Частина коду реалізації слайдера

Нижче є блоки товарів «Пропозиції» та «Переглянуті», вони також змінюються між собою за принципом слайдера.

Частина коду реалізації наведена на рисунку 3.13.

```
$(function(){

    if(document.getElementById("price-range")){

    var nonLinearSlider = document.getElementById('price-range');

    noUiSlider.create(nonLinearSlider, {
        connect: true,
        behaviour: 'tap',
        start: [ 500, 4000 ],
        range: {
            // Starting at 500, step the value by 500,
            // until 4000 is reached. From there, step by 1000.
            'min': [ 0 ],
            '10%': [ 500, 500 ],
            '50%': [ 4000, 1000 ],
            'max': [ 10000 ]
        }
    });


    var nodes = [
        document.getElementById('lower-value'), // 0
        document.getElementById('upper-value')  // 1
    ];

    // Display the slider value and how far the handle moved
    // from the left edge of the slider.
    nonLinearSlider.noUiSlider.on('update', function ( values, handle, unencoded, isTap, positions ) {
        nodes[handle].innerHTML = values[handle];
    });

    }

});
```

Рисунок 3.13 – Частина коду реалізації слайдера блоків товарів

При наведенні на кнопки під картками товарів, відбувається ефект акордеону. Коли відображається підпис тої чи іншої кнопки. Частина коду наведена на рисунку 3.14.

```
$('.collapse').on('shown.bs.collapse', function(){
    $(this).parent().find(".lnr-arrow-right").removeClass("lnr-arrow-right").addClass("lnr-arrow-left");
}).on('hidden.bs.collapse', function(){
    $(this).parent().find(".lnr-arrow-left").removeClass("lnr-arrow-left").addClass("lnr-arrow-right");
});
```

Рисунок 3.14 – Частина коду реалізації акордеону

Після блоку «Пропозиції» та «Переглянуті» реалізований таймер, який показує скільки часу залишилось до закінчення акції. Частина коду функції наведена на рисунку 3.15.

```
$(document).ready(function() {
    $('#mc_embed_signup').find('form').ajaxChimp();
});



 if(document.getElementById("js-countdown")){

    var countdown = new Date("October 17, 2018");

    function getRemainingTime(endtime) {
        var milliseconds = Date.parse(endtime) - Date.parse(new Date());
        var seconds = Math.floor(milliseconds / 1000 % 60);
        var minutes = Math.floor(milliseconds / 1000 / 60 % 60);
        var hours = Math.floor(milliseconds / (1000 * 60 * 60) % 24);
        var days = Math.floor(milliseconds / (1000 * 60 * 60 * 24));

    return {
        'total': milliseconds,
        'seconds': seconds,
        'minutes': minutes,
        'hours': hours,
        'days': days
        };
    }

    function initClock(id, endtime) {
        var counter = document.getElementById(id);
        var daysItem = counter.querySelector('.js-countdown-days');
        var hoursItem = counter.querySelector('.js-countdown-hours');
        var minutesItem = counter.querySelector('.js-countdown-minutes');
        var secondsItem = counter.querySelector('.js-countdown-seconds');

    function updateClock() {
        var time = getRemainingTime(endtime);

        daysItem.innerHTML = time.days;
        hoursItem.innerHTML = ('0' + time.hours).slice(-2);
        minutesItem.innerHTML = ('0' + time.minutes).slice(-2);
        secondsItem.innerHTML = ('0' + time.seconds).slice(-2);

        if (time.total <= 0) {
          clearInterval(timeinterval);
        }
        }

        updateClock();
        var timeinterval = setInterval(updateClock, 1000);
    }

    initClock('js-countdown', countdown);
```

Рисунок 3.15 – Частина функції реалізації таймера

При прокручуванні сторінки головне меню «приклеєне» до екрану, що дає можливість користувачам зручніше переключатися між сторінками.

Частина коду реалізації «приклеєного» меню наведена на рисунку 3.16.

```
(function($) {
  var defaults = {
      topSpacing: 0,
      bottomSpacing: 0,
      className: 'is-sticky',
      wrapperClassName: 'sticky-wrapper',
      center: false,
      getWidthFrom: '',
      responsiveWidth: false
    },
    $window = $(window),
    $document = $(document),
    sticked = [],
    windowHeight = $window.height(),
    scroller = function() {
      var scrollTop = $window.scrollTop(),
        documentHeight = $document.height(),
        dwh = documentHeight - windowHeight,
        extra = (scrollTop > dwh) ? dwh - scrollTop : 0;

      for (var i = 0; i < sticked.length; i++) {
        var s = sticked[i],
          elementTop = s.stickyWrapper.offset().top,
          etse = elementTop - s.topSpacing - extra;

        if (scrollTop <= etse) {
          if (s.currentTop !== null) {
            s.stickyElement
              .css('width', '')
              .css('position', '')
              .css('top', '');
            s.stickyElement.trigger('sticky-end', [s]).parent().removeClass(s.className);
            s.currentTop = null;
          }
        }
        else {
          var newTop = documentHeight - s.stickyElement.outerHeight()
            - s.topSpacing - s.bottomSpacing - scrollTop - extra;
          if (newTop < 0) {
            newTop = newTop + s.topSpacing;
          } else {
            newTop = s.topSpacing;
          }
          if (s.currentTop != newTop) {
            s.stickyElement
              .css('width', s.stickyElement.width())
              .css('position', 'fixed')
              .css('top', newTop);

            if (typeof s.getWidthFrom !== 'undefined') {
              s.stickyElement.css('width', $(s.getWidthFrom).width());
            }
```

Рисунок 3.16 – Частина коду реалізації приклеїного меню

На сторінці «Контакти» є реалізована карта для відображення місця знаходження компанії.

Частина коду з реалізації карти наведена на рисунку 3.17.

```
if ($("#mapBox").length) {
    var $lat = $("#mapBox").data("lat");
    var $lon = $("#mapBox").data("lon");
    var $zoom = $("#mapBox").data("zoom");
    var $marker = $("#mapBox").data("marker");
    var $info = $("#mapBox").data("info");
    var $markerLat = $("#mapBox").data("mlat");
    var $markerLon = $("#mapBox").data("mlon");
    var map = new GMaps({
      el: "#mapBox",
      lat: $lat,
      lng: $lon,
      scrollwheel: false,
      scaleControl: true,
      streetViewControl: false,
      panControl: true,
      disableDoubleClickZoom: true,
      mapTypeControl: false,
      zoom: $zoom,
      styles: [
        {
          featureType: "water",
          elementType: "geometry.fill",
          stylers: [
            {
              color: "#dcdfe6"
            }
          ]
        },
        {
          featureType: "transit",
          stylers: [
            {
              color: "#808080"
            },
            {
              visibility: "off"
            }
          ]
        },
        {
          featureType: "road.highway",
          elementType: "geometry.stroke",
          stylers: [
            {
              visibility: "on"
            },
            {
              color: "#dcdfe6"
```

Рисунок 3.17 – Частина коду реалізації карти

Для кращої мобільності на сайті реалізований пошук. Щоб він не займа багато місця та не заважав користувачу, було зроблено спливаюче вікно пошуку. Частина коду наведена на рисунку 3.18.

```
$("#search_input_box").hide();
$("#search").on("click", function () {
    $("#search_input_box").slideToggle();
    $("#search_input").focus();
});
$("#close_search").on("click", function () {
    $('#search_input_box').slideUp(500);
});
```

Рисунок 3.18 – Частина коду скритого пошуку

Для фільтрації об’єктів на сторінці «Каталог товарів», було використано алгоритм сортування. Частина коду сортування наведена на рис. 3.19.

```
const filterBox = document.querySelectorAll('.box');

document.querySelector('nav').addEventListener('click', event => {
    if (event.target.tagName !== 'LI') return false;

    let filterClass = event.target.dataset['f'];

    filterBox.forEach(elem => {
        elem.classList.remove('hide');
        if (!elem.classList.contains(filterClass) && filterClass!== 'all') {
            elem.classList.add('hide');
        }
    });

});
```

Рисунок 3.19 – Частина коду сортування

Для сортування та реалізації рейтингу на сторінці «Моніторинг якості» було реалізовано сортування. Частина коду процедури сортування наведена на рисунку 3.20.

```
function mySort() {
    let nav = document.querySelector('#nav');
    for (let i = 0; i < nav.children.length; i++) {
        for (let j = i; j < nav.children.length; j++) {
            if (+nav.children[i].getAttribute('data-sort') > +nav.children[j].getAttribute('data-sort')) {
                replacedNode = nav.replaceChild(nav.children[j], nav.children[i]);
                insertAfter(replacedNode, nav.children[i]);
            }
        }
    }
}

function mySortDesc() {
    let nav = document.querySelector('#nav');
    for (let i = 0; i < nav.children.length; i++) {
        for (let j = i; j < nav.children.length; j++) {
            if (+nav.children[i].getAttribute('data-sort') < +nav.children[j].getAttribute('data-sort')) {
                replacedNode = nav.replaceChild(nav.children[j], nav.children[i]);
                insertAfter(replacedNode, nav.children[i]);
            }
        }
    }
}
```

Рисунок 3.20 – Частина коду процедури сортування

3.4 Висновки

У третьому розділі спроєктовано користувацький інтерфейс веб-системи, розроблена схема переходів між сторінками.

Розроблено графічний матеріал для наповнення веб-системи «Трудяща бджілка».

Виконано розробку програмного забезпечення веб-системи, описано розробку основних її модулів.

4. ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Вибір методів тестування програмного забезпечення

Тестування програмного забезпечення – технологічно-дослідницький процес, призначений для виявлення даних про якість продукту [18].

Внутрішня поведінка програми відбувається методом «білої скриньки». Даний метод перевіряє правильність взаємодії елементів та їх будову.

«Біла скринька» – це тестування, яке базується на системі аналізу структури для управління програми. Повністю перевірена програма вважається тоді, коли перевіряються всі маршрути, які можуть бути реалізовані в програмі.

Формуються такі варіанти тестування:

- перевірка всіх незалежних програмних маршрутів;
- перевірка всіх гілок True, False для всіх логічних рішень;
- виконання всіх циклів (в межах їх меж і діапазонів);
- аналіз правильності внутрішніх структур даних.

Переваги тестування «білої скриньки»:

- Мінімальна кількість помилок в «центрі» і максимальна кількість помилок на «периферії» програми.

- Потоки контролю даних і попередніх припущення на їх основі часто не вірні. Тому маршрут може бути зазвичай, через це модель обчислень жахливо обробляється.

- Коли пишеться алгоритм у вигляді тексту, можна допустити різноманітні помилки в синтаксисі.

- Є результати в програмах, які залежать не від вихідних даних, а від внутрішніх станів.

Недоліки, з якими стикаються при тестуванні:

- велика кількість незалежних маршрутів, яка може бути занадто великою;

- після повного тестування немає гарантій що програма 100% працює;

- неможливість виявити помилки, якщо їх поява не обумовлена даними.

Метод тестування «чорної скриньки» допомагає дослідити можливості програмних функцій (в усіх областях). Під час проходження тестування відбувається ігнорування внутрішньої логічної структури функцій. Методи випробування демонструють:

- виконання програмних функцій;

- приймання вихідних даних;

- отримані результати;

- зберігання цілісності збереженої інформації.

Тестування «чорної скриньки» забезпечує пошук таких категорій помилок:

- відсутність або не правильність функцій;

- інтерфейсні помилки;

- помилки, які виникли в зовнішніх даних або у доступі до зовнішньої бази даних;

- помилкові характеристики (необхідна ємність пам'яті тощо);

- помилки завершення або ініціалізації.

Проаналізувавши плюси та мінуси методів тестування «чорної скриньки» та «білої скриньки» з урахуванням мети веб-системи, вибрано метод «чорної скриньки» для тестування власної веб-системи

4.2 Тестування розробленого додатку

Для початку, запустимо веб-систему щоб переконатись у її роботі. Після запуску має відобразитись головне меню зі скролером. Результат запуску додатку наведено на рисунку 4.1.

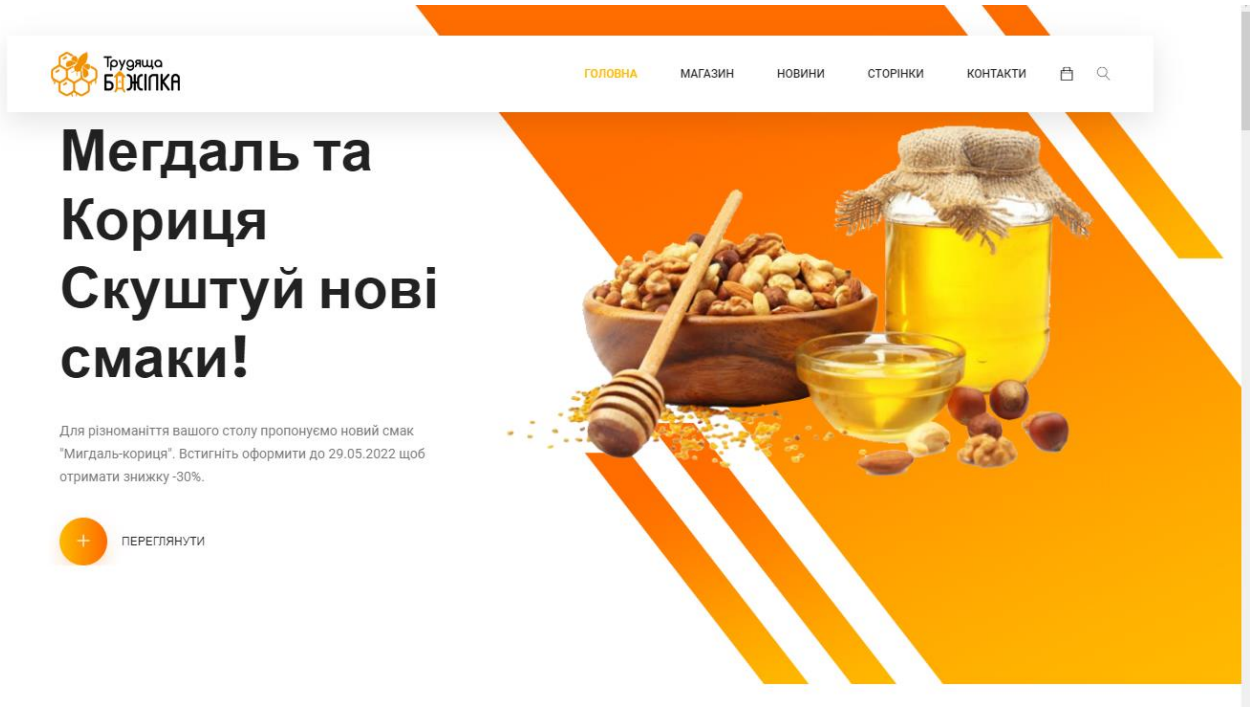

Рисунок 4.1 – Головна сторінка «Трудяща бджілка»

Після запуску відбувається перевірка функціоналу. Для веб-системи потрібно перевірити функціонування скролерів. Для цього потрібно перетягнути відображену пропозицію вбік для відображення наступної. Результат перетягування відповідної позиції наведено на рисунку 4.2

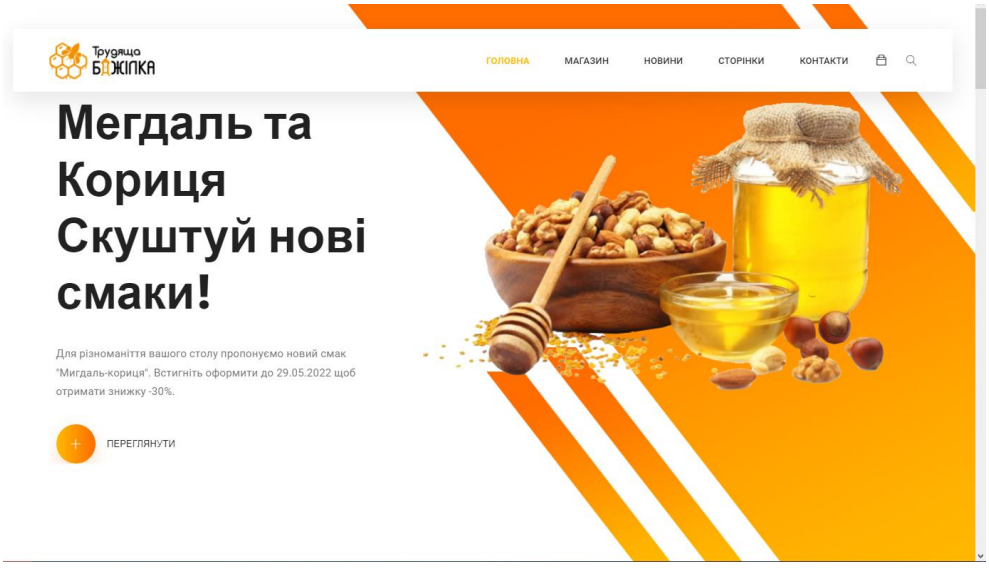

Рисунок 4.2 – Результат перетягування позиції скроллера

Перевірка скроллера з блоками пропозиції товару. Щоб виконати перехід необхідно натиснути відповідні кнопки «Next» і «Prev», або ж просто перетягнути блок у потрібну сторону.

Результат переходу між блоками пропозиції на рисунку 4.3.

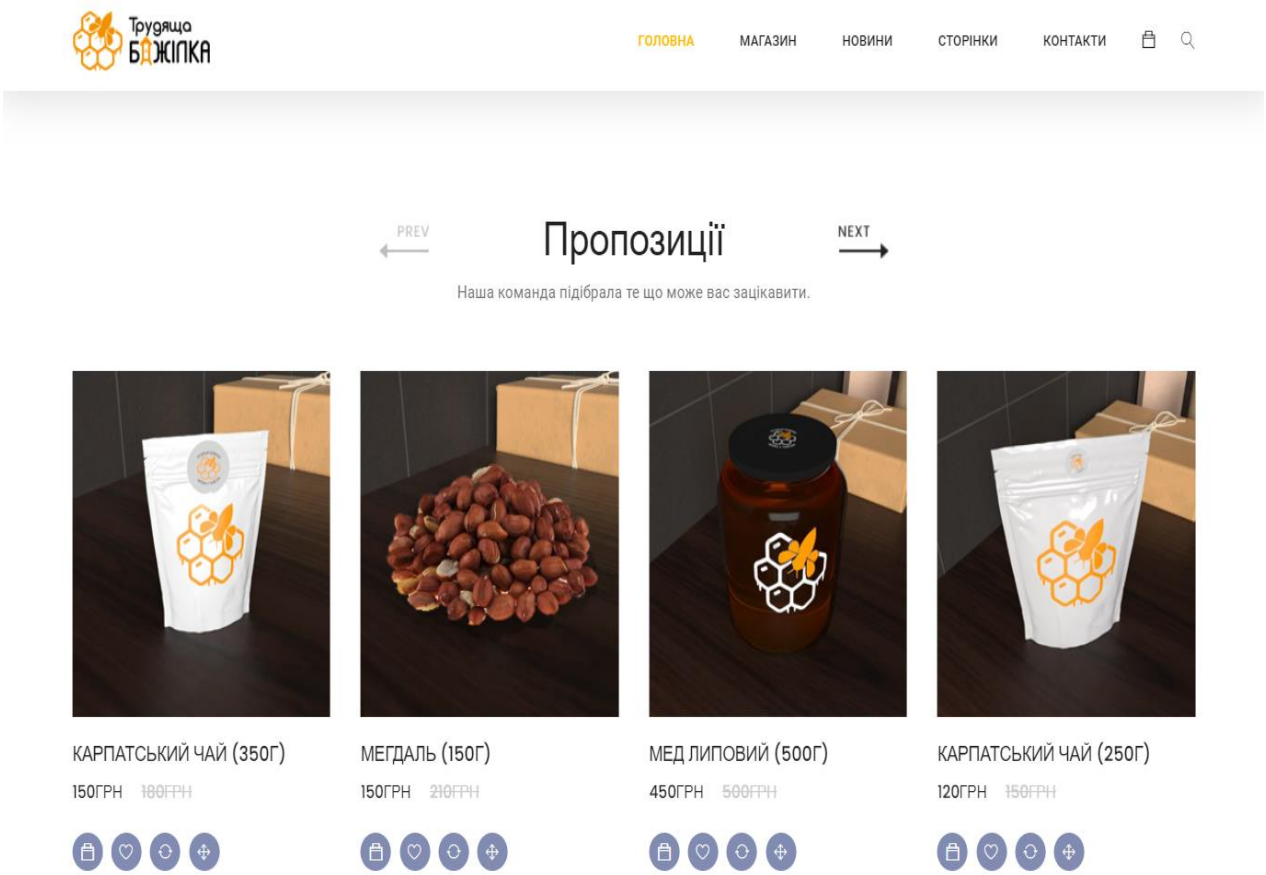

Рисунок 4.3 – Результат натискання кнопки «Next»

Після блоку з «Пропозиціями» реалізований таймер закінчення акції. Пропозиція має закінчуватись через місяць

Результат відображення таймера наведено на рисунку 4.4.

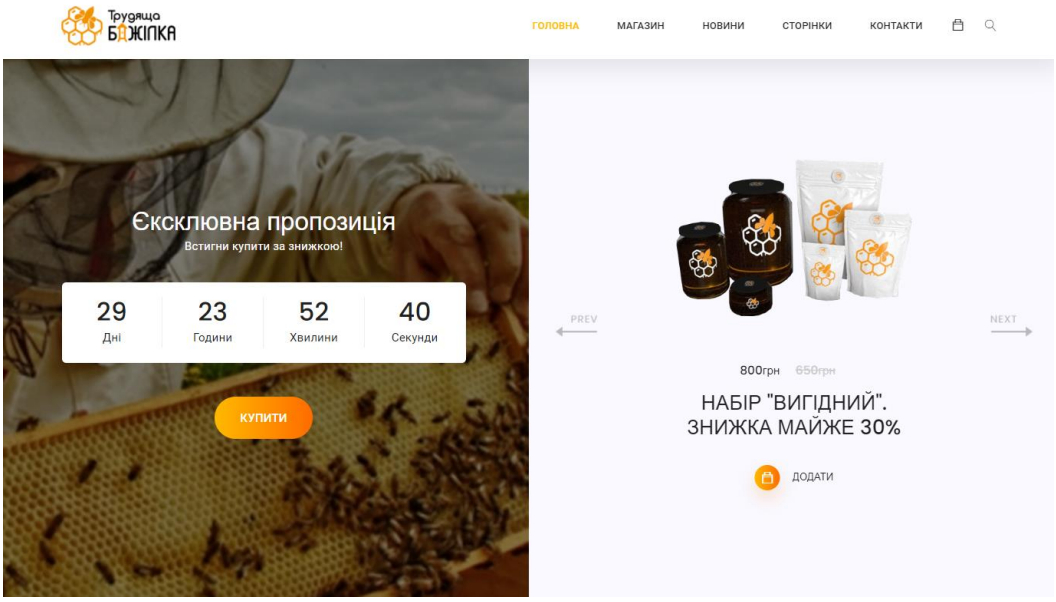

Рисунок 4.4 – Результат відображення таймера

Однією з основних функцій є фільтрація продукції. Для фільтрування потрібно вибрати, за яким принципом буде здійснюватися фільтрація і після чого відбудеться завантаження результатів. Для тестування відфільтруємо товари за рейтингом. Результат сортування за рейтингом наведено на рисунку 4.5.

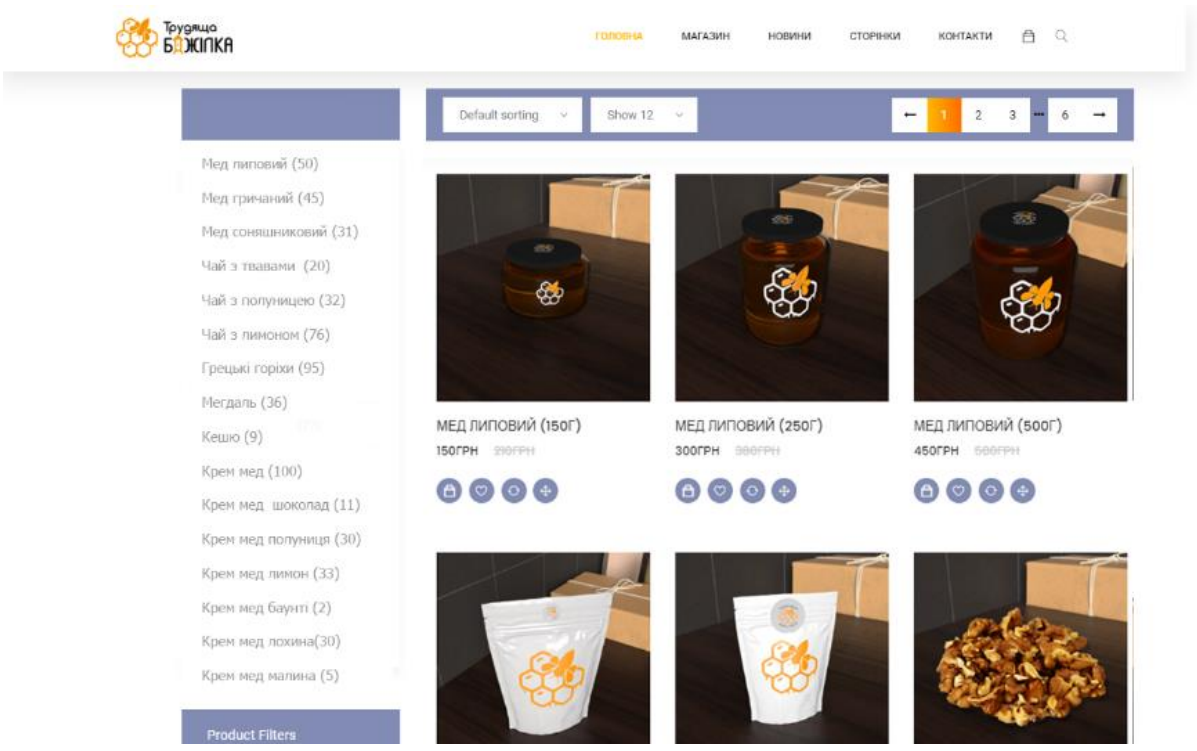

Рисунок 4.5 – Результат сортування по рейтингу

Якщо товари відфільтровані, то можна обрати потрібний та перейти до оформлення замовлення.

Перейшовши на товар, бачимо індикатор якості та кристалізації.

Результат переходу до сторінки з товаром наведено на рисунку 4.6.

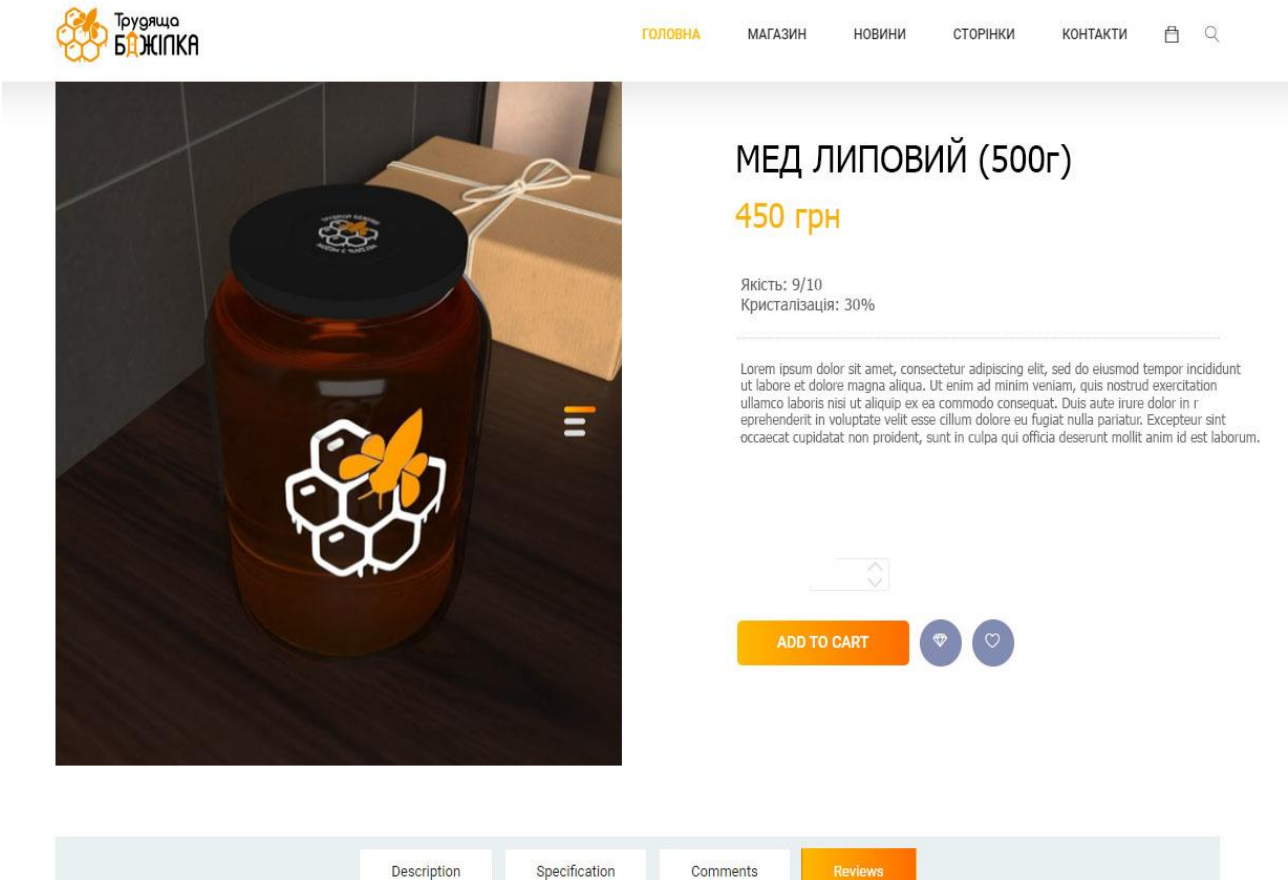

Рисунок 4.6 – Результат переходу до сторінки з товаром

Потрібно додати товар в кошик і оформити замовлення для того, щоб пересвідчитись у працездатності цієї функції.

Результат додавання товару до пошуку наведено на рисунку 4.7.

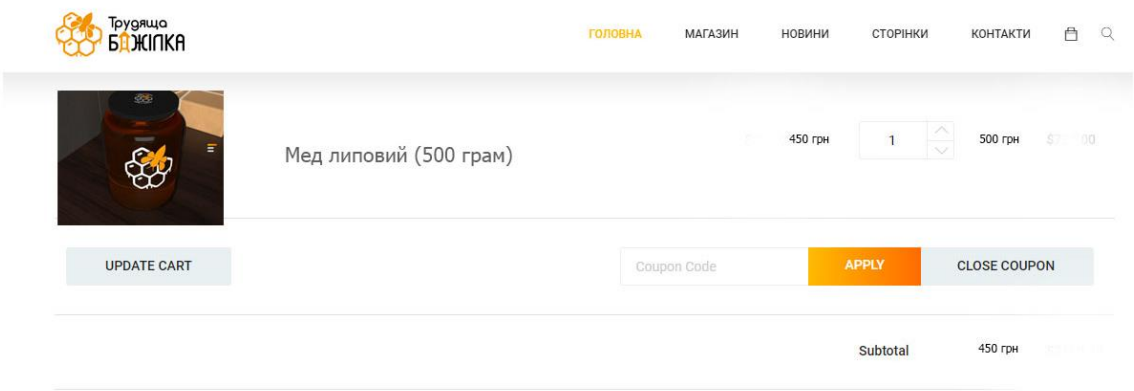

Рисунок 4.7 – Результат додавання товару в кошик

Після оформлення замовлення потрібно перевірити, як працює система моніторингу якості меду. Потрібно перейти на сторінку «Моніторинг якості» та пересвідчитися, що сторінка відображає товари за рейтингом та показує оцінку якості і кристалізації. Результат вигляду сторінки наведено на рис. 4.8.

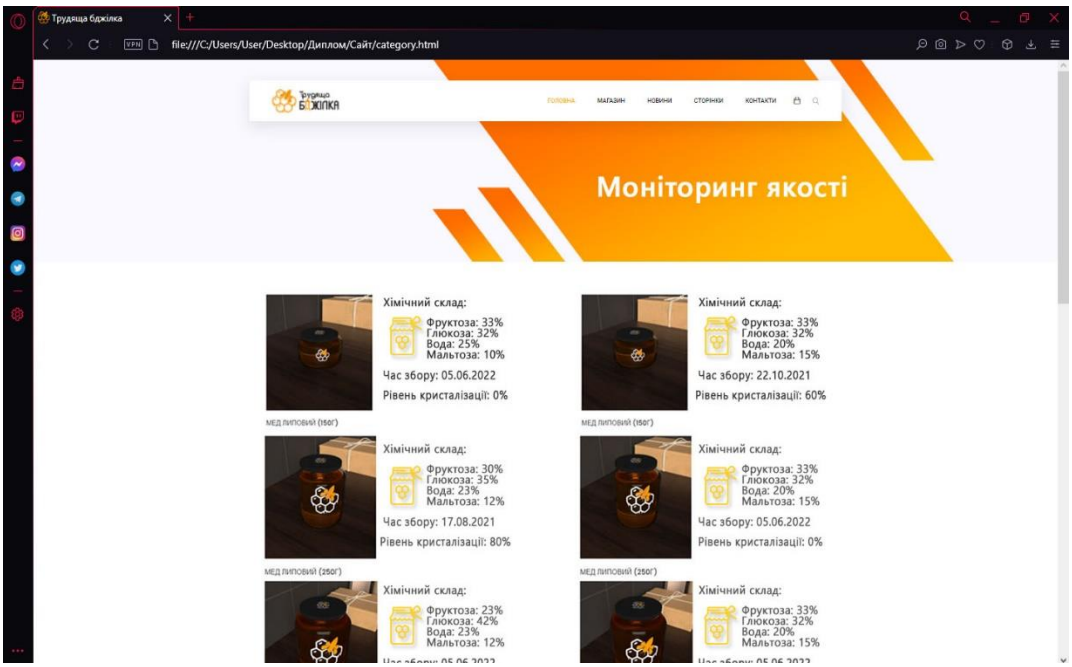

Рисунок 4.8 – Результат оцінки якості товару

На сайті передбачена система додавання статей та ведення блогу для кращої комунікації зі споживачами.

На рис. 4.9 можна побачити інтерфейс та блок з додаванням нових статей.

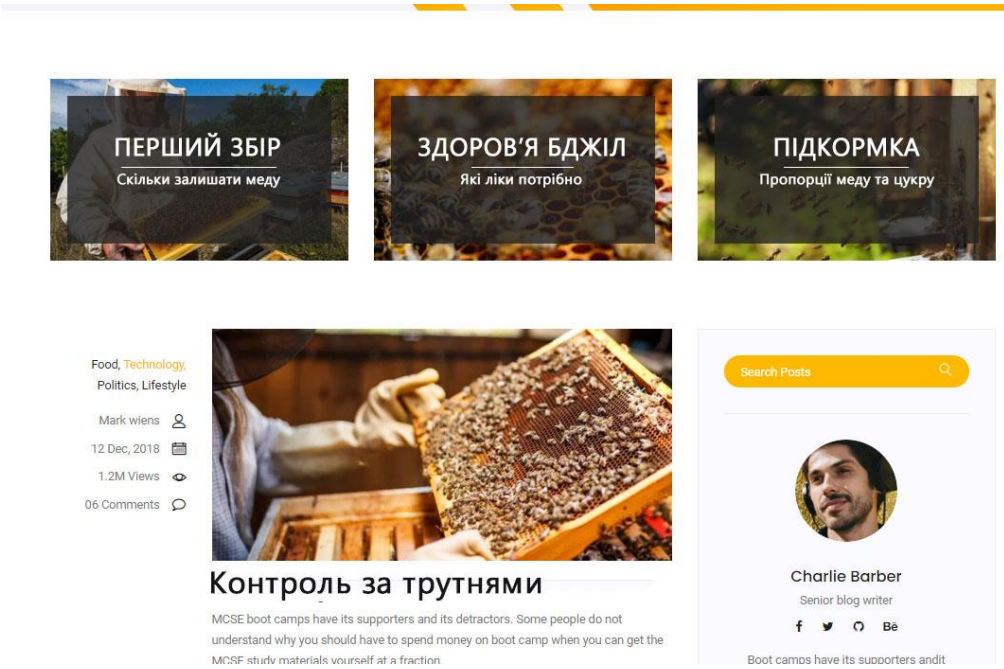

Рисунок 4.8 – Результат додавання товару в кошик

Результатами тестування веб-системи «Трудяща бджілка» підтверджено працездатність усіх заявлених функцій.

4.3 Висновки

Виконано аналіз методів тестування, і в результаті було обрано метод тестування «чорної скриньки». За цим методом проведено тестування розробленої веб-системи, яке підтвердило його повну працездатність.

ВИСНОВКИ

Проаналізовано предметну область, визначено основне призначення веб систем в інтернет просторі.

Виконано аналіз аналогів розробки, визначено їх переваги та недоліки.

Виконано аналіз існуючих реалізацій схожих веб-систем із функцією моніторингу якості та кристалізації меду. Виконано постановку вимог до розроблюваної веб-системи. Розроблено метод і модель роботи системи з урахуванням наявного функціоналу інтернет-магазину.

Запропоновано метод швидкого сортування великої кількості даних, що дозволяє за короткий проміжок часу обробити велику кількість даних та подати повний аналіз якості й хімічного складу меду.

Розроблено функцію, яка в залежності від виду меду і його хімічного складу та перебування його на складі, визначає рівень кристалізації, щоб користувачу було зручніше обрати продукт, який задовільнить саме його.

Спроєктовано користувацький інтерфейс веб-системи, розроблена карта сайту, зверстано всі потрібні сторінки у зручному та приємному оку дизайні. Виконано розробку програмного коду веб-системи. Розроблено графічний матеріал для наповнення частини для продажу.

Проведене тестування програмної розробки підтверджує працездатність роботи системи та дає змогу перевірити якість меду, вибрати необхідний товар та придбати його.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка моніторингової системи для продажу меду. Веб-системи – [Електронний ресурс] – Режим доступу до матеріалу: http://websecurity.com.ua/security/chapter1/
2. Войтко В.В. Розробка моніторингової системи для продажу меду В.В. Войтко, Г.О. Черноволик, Л.М. Круподьорова, В.В. Тоха // Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці:дослідження, проблеми, перспективи -2022", Секція - Інформаційні технології та комп‘ютерна інженерія. [Електронний ресурс] – Режим доступу до матеріалу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/viewFile/16211/136470.
3. Портал знань: програмування сайтів – [Електронний ресурс]: Режим доступу до матеріалу: http://www.znannya.org/?view=WebDev
4. Довідник по Bracket – [Електронний ресурс]: Режим доступу до матеріалу: https://winsoft.com.ua/windows/rozrobnikam/redaktori-kodu/brackets
5. Cases. Види веб-ресурсів – [Електронний ресурс]: Режим доступу до матеріалу: https://cases.media/creativepractice/article/klasifikaciya-vebsaitiv
6. BeeLab [Електронний ресурс] – Режим доступу до ресурсу: https://beelab.in.ua/ua/
7. Zolota Sota [Електронний ресурс] – Режим доступу до ресурсу: https://zolotasota.com
8. Ієрархічні методи – [Електронний ресурс]: Режим доступу до матеріалу: http://wiki.kspu.kr.ua/index.php/Ієрархічні_методи.
9. Білоусова Л.І. Інформатика. Навчальний посібник // Білоусова Л.І., Муравка А.С., Олефіренко Н.В. - Харків: Факт, 2009. – 352 с.
10. Norton T. Learning JS by Developing Games with Unity 3D Beginner's Guide / T. Norton – Birmingham: Packt Publishing, 2013. – 271 p. – ISBN 978-1-84969-658-6.

11. Creighton R. H. Brasket by Example Beginner's Guide / R.H. Creighton – Birmingham: Packt Publishing, 2010. – 364 p. – ISBN 978-1-849690-54-6.

12. Воронки продажу на сайті – [Електронний ресурс]: Режим доступу до матеріалу: https://www.mango-office.ua/newsletter/chto-takoe-voronka-prodazh-v-internet-magazine/

13. Підвищення відвідування сайту – [Електронний ресурс]: Режим доступу до матеріалу: https://hostiq.ua/blog/ukr/35-tools-for-web-analytics/

14. Сортування товарів в каталозі інтернет-магазину – [Електронний ресурс]: Режим доступу до матеріалу: http://programer.org.ua/sortuvannya-tovariv-u-katalozi-internet-magazinu/

15. User experience – [Електронний ресурс]: Режим доступу до матеріалу: https://en.wikipedia.org/wiki/User_experience.

16. Adobe Illustrator – [Електронний ресурс]: Режим доступу до матеріалу: https://ru.wikipedia.org/wiki/Adobe_Illustrator.

17. Importing Assets – [Електронний ресурс]: Режим доступу до матеріалу: https://docs.unity3d.com/560/Documentation/Manual/ImportingAssets.html

18. Тестування програмного забезпечення – [Електронний ресурс]: Режим доступу до матеріалу: http://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.

19. DOU – [Електронний ресурс]: Режим доступу до матеріалу: https://dou.ua/.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. Романюк О.Н
25 березня 2022р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка моніторингової системи для продажу меду та тестування його якості» студенту Тосі Вадиму Валентиновичу
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доцент кафедри ПЗ, В.В. Войтко
25 березня 2022 р.
Виконав:студент гр. 3ПІ-18б,
В.В. Тоха
25 березня 2022 р.

Вінниця ВНТУ 2022

1. Найменування та галузь застосування

Розробка має назву «Трудяща бджілка» і виконується в рамках бакалаврської дипломної роботи.

Згідно отриманого завдання кінцевий програмний продукт може використовуватись офіційною організацією.

2. Підстава для розробки

Підставою для розробки даної бакалаврської дипломної роботи є рішення засідання кафедри програмного забезпечення (протокол №12 від «7» лютого 2022 року).

3. Мета та призначення розробки

Метою роботи є підвищення ефективності роботи Інтернет-магазину з продажу меду шляхом розробки та використання моніторингової системи для продажу і тестування якості продукту, що дозволить систематизувати товар та забезпечити високий рівень його якості.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

- аналіз поведінки користувача на сайті інтернет-магазину;
- - розробка методу і моделі роботи веб системи;
- - розробка креативного графічного оформлення інтернет-магазину;
- - розробка інтуїтивно зрозумілого користувацького інтерфейсу;
- - програмна реалізація інтернет-магазину «Трудяща бджілка» для продажу та моніторингу якості меду;
- - тестування створеного веб ресурсу.

4. Технічні вимоги

Склад комплексу *технічних засобів*

Для розв’язку задачі буде використовуватися персональна електронно-обчислювальна машина (ПЕОМ).

Вимоги до складових *частин комплексу технічних засобів.*

Для нормальної роботи програми, необхідний персональний комп’ютер з наступною мінімальною конфігурацією:

1. Процесор з частотою не менше 2.0 Гц;

2. Оперативна пам’ять ємністю не менше 4 Гб;

3. Запам’ятовуючий пристрій ємністю 500 Gb.

Вимоги до програмного *забезпечення.*

Програмний продукт розробляється на мовах HTML, CSS, JS, PHP на візуалізаторі Bractet 2018.

5. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка;

– технічне обґрунтування доцільності розробки;

– лістинги програми.

6. Стадії і етапи розробки

Завдання на проектування видане 26.05.2022 року. Проектування та дослідження повинно бути завершеним до 10.06.2022 року.

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану галузі, обґрунтування доцільності розробки	26.03.2022-05.05.2022	Вик.
2	Розробка методів і моделей реалізації веб системи	06.05.2022-12.05.2022	Вик.
3	Проектування структур даних та модулів системи	13.05.2022-20.05.2022	Вик.
4	Варіантний аналіз та обґрунтування вибору засобів веб системи	21.05.2022-23.05.2022	Вик.
5	Розробка програмного коду веб системи	24.05.2022-30.05.2022	Вик.
6	Тестування роботи	31.05.2022-03.06.2022	Вик.
7	Оформлення матеріалів до захисту БДР	04.06.2022-10.06.2022	Вик.

7. Порядок контролю і приймання

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б. Лістинг програми

Main.js

```
$(document).ready(function(){
	"use strict";

	var window_width 	= $(window).width(),
	window_height 		= window.innerHeight,
	header_height 		= $(".default-header").height(),
	header_height_static = $(".site-header.static").outerHeight(),
	fitscreen 		= window_height - header_height;


	$(".fullscreen").css("height", window_height)
  $(".fitscreen").css("height", fitscreen);

  $('select').niceSelect();


  $('.navbar-nav li.dropdown').hover(function() {
  $(this).find('.dropdown-menu').stop(true, true).delay(200).fadeIn(500);
  }, function() {
  $(this).find('.dropdown-menu').stop(true, true).delay(200).fadeOut(500);
  });

  $('.img-pop-up').magnificPopup({
    type: 'image',
    gallery:{
    enabled:true
    }
  });

  // Search Toggle
  $("#search_input_box").hide();
  $("#search").on("click", function () {
    $("#search_input_box").slideToggle();
    $("#search_input").focus();
  });
  $("#close_search").on("click", function () {
    $('#search_input_box').slideUp(500);
  });

			$(".sticky-header").sticky();

  $(".active-banner-slider").owlCarousel({
    items:1,
    autoplay:false,
    autoplayTimeout: 5000,
    loop:true,
    nav:true,
    navText:["<img src='img/banner/prev.png'>","<img src='img/banner/next.png'>"],
```

```
        dots:false
    });

    $(".active-product-area").owlCarousel({
        items:1,
        autoplay:false,
        autoplayTimeout: 5000,
        loop:true,
        nav:true,
        navText:["<img src='img/product/prev.png'>","<img src='img/product/next.png'>"],
        dots:false
    });

    $(".s_Product_carousel").owlCarousel({
      items:1,
      autoplay:false,
      autoplayTimeout: 5000,
      loop:true,
      nav:false,
      dots:true
    });

    $(".active-exclusive-product-slider").owlCarousel({
        items:1,
        autoplay:false,
        autoplayTimeout: 5000,
        loop:true,
        nav:true,
        navText:["<img src='img/product/prev.png'>","<img src='img/product/next.png'>"],
        dots:false
    });

    $('.collapse').on('shown.bs.collapse', function(){
        $(this).parent().find(".lnr-arrow-right").removeClass("lnr-arrow-right").addClass("lnr-arrow-
left");
    }).on('hidden.bs.collapse', function(){
        $(this).parent().find(".lnr-arrow-left").removeClass("lnr-arrow-left").addClass("lnr-arrow-
right");
    });

  // Select all links with hashes
  $('.main-menubar a[href*="#"]')
    // Remove links that don't actually link to anything
    .not('[href="#"]')
    .not('[href="#0"]')
    .click(function(event) {
      // On-page links
      if (
        location.pathname.replace(/^\//, '') == this.pathname.replace(/^\//, '')
        &&
        location.hostname == this.hostname
      ) {
```

```
    // Figure out element to scroll to
    var target = $(this.hash);
    target = target.length ? target : $('[name=' + this.hash.slice(1) + ']');
    // Does a scroll target exist?
    if (target.length) {
      // Only prevent default if animation is actually gonna happen
      event.preventDefault();
      $('html, body').animate({
        scrollTop: target.offset().top-70
      }, 1000, function() {
        // Callback after animation
        // Must change focus!
        var $target = $(target);
        $target.focus();
        if ($target.is(":focus")) { // Checking if the target was focused
          return false;
        } else {
          $target.attr('tabindex','-1'); // Adding tabindex for elements not focusable
          $target.focus(); // Set focus again
        };
      });
    }
  }
});

    $(document).ready(function() {
      var form = $('#booking'); // contact form
      var submit = $('.submit-btn'); // submit button
      var alert = $('.alert-msg'); // alert div for show alert message

      // form submit event
      form.on('submit', function(e) {
        e.preventDefault(); // prevent default form submit

        $.ajax({
          url: 'booking.php', // form action url
          type: 'POST', // form submit method get/post
          dataType: 'html', // request type html/json/xml
          data: form.serialize(), // serialize form data
          beforeSend: function() {
            alert.fadeOut();
            submit.html('Sending....'); // change submit button text
          },
          success: function(data) {
            alert.html(data).fadeIn(); // fade in response data
            form.trigger('reset'); // reset form
            submit.attr("style", "display: none !important");; // reset submit button text
          },
          error: function(e) {
            console.log(e)
          }
        });
```

```
    });
  });



$(document).ready(function() {
  $('#mc_embed_signup').find('form').ajaxChimp();
});



 if(document.getElementById("js-countdown")){

  var countdown = new Date("October 17, 2018");

  function getRemainingTime(endtime) {
    var milliseconds = Date.parse(endtime) - Date.parse(new Date());
    var seconds = Math.floor(milliseconds / 1000 % 60);
    var minutes = Math.floor(milliseconds / 1000 / 60 % 60);
    var hours = Math.floor(milliseconds / (1000 * 60 * 60) % 24);
    var days = Math.floor(milliseconds / (1000 * 60 * 60 * 24));

  return {
    'total': milliseconds,
    'seconds': seconds,
    'minutes': minutes,
    'hours': hours,
    'days': days
    };
  }

  function initClock(id, endtime) {
    var counter = document.getElementById(id);
    var daysItem = counter.querySelector('.js-countdown-days');
    var hoursItem = counter.querySelector('.js-countdown-hours');
    var minutesItem = counter.querySelector('.js-countdown-minutes');
    var secondsItem = counter.querySelector('.js-countdown-seconds');

  function updateClock() {
    var time = getRemainingTime(endtime);

    daysItem.innerHTML = time.days;
    hoursItem.innerHTML = ('0' + time.hours).slice(-2);
    minutesItem.innerHTML = ('0' + time.minutes).slice(-2);
    secondsItem.innerHTML = ('0' + time.seconds).slice(-2);

    if (time.total <= 0) {
      clearInterval(timeinterval);
    }
    }
```

```
        updateClock();
        var timeinterval = setInterval(updateClock, 1000);
    }

    initClock('js-countdown', countdown);

};



  $('.quick-view-carousel-details').owlCarousel({
      loop: true,
      dots: true,
      items: 1,
  })


 $(function(){

    if(document.getElementById("price-range")){

    var nonLinearSlider = document.getElementById('price-range');

    noUiSlider.create(nonLinearSlider, {
        connect: true,
        behaviour: 'tap',
        start: [ 500, 4000 ],
        range: {
            // Starting at 500, step the value by 500,
            // until 4000 is reached. From there, step by 1000.
            'min': [ 0 ],
            '10%': [ 500, 500 ],
            '50%': [ 4000, 1000 ],
            'max': [ 10000 ]
        }
    });


    var nodes = [
        document.getElementById('lower-value'), // 0
        document.getElementById('upper-value')  // 1
    ];

    // Display the slider value and how far the handle moved
    // from the left edge of the slider.
    nonLinearSlider.noUiSlider.on('update', function ( values, handle, unencoded, isTap, positions
) {
        nodes[handle].innerHTML = values[handle];
    });

    }
```

```
});

$('.have-btn').on('click', function(e){
   e.preventDefault();
   $('.have-btn span').text(function(i, text){
     return text === "Have a Coupon?" ? "Close Coupon" : "Have a Coupon?";
   })
   $('.cupon-code').fadeToggle("slow");
});

$('.load-more-btn').on('click', function(e){
   e.preventDefault();
   $('.load-product').fadeIn('slow');
   $(this).fadeOut();
});


var value,
   quantity = document.getElementsByClassName('quantity-container');

function createBindings(quantityContainer) {
   var quantityAmount = quantityContainer.getElementsByClassName('quantity-amount')[0];
   var increase = quantityContainer.getElementsByClassName('increase')[0];
   var decrease = quantityContainer.getElementsByClassName('decrease')[0];
   increase.addEventListener('click', function () { increaseValue(quantityAmount); });
   decrease.addEventListener('click', function () { decreaseValue(quantityAmount); });
}

function init() {
   for (var i = 0; i < quantity.length; i++ ) {
      createBindings(quantity[i]);
   }
};

function increaseValue(quantityAmount) {
   value = parseInt(quantityAmount.value, 10);

   console.log(quantityAmount, quantityAmount.value);

   value = isNaN(value) ? 0 : value;
   value++;
   quantityAmount.value = value;
}

function decreaseValue(quantityAmount) {
   value = parseInt(quantityAmount.value, 10);

   value = isNaN(value) ? 0 : value;
   if (value > 0) value--;

   quantityAmount.value = value;
}
```

```
init();


  if ($("#mapBox").length) {
     var $lat = $("#mapBox").data("lat");
     var $lon = $("#mapBox").data("lon");
     var $zoom = $("#mapBox").data("zoom");
     var $marker = $("#mapBox").data("marker");
     var $info = $("#mapBox").data("info");
     var $markerLat = $("#mapBox").data("mlat");
     var $markerLon = $("#mapBox").data("mlon");
     var map = new GMaps({
       el: "#mapBox",
       lat: $lat,
       lng: $lon,
       scrollwheel: false,
       scaleControl: true,
       streetViewControl: false,
       panControl: true,
       disableDoubleClickZoom: true,
       mapTypeControl: false,
       zoom: $zoom,
       styles: [
         {
           featureType: "water",
           elementType: "geometry.fill",
           stylers: [
             {
               color: "#dcdfe6"
             }
           ]
         },
         {
           featureType: "transit",
           stylers: [
             {
               color: "#808080"
             },
             {
               visibility: "off"
             }
           ]
         },
         {
           featureType: "road.highway",
           elementType: "geometry.stroke",
           stylers: [
             {
               visibility: "on"
             },
             {
```

```
      color: "#dcdfe6"
    }
  ]
},
{
  featureType: "road.highway",
  elementType: "geometry.fill",
  stylers: [
    {
      color: "#ffffff"
    }
  ]
},
{
  featureType: "road.local",
  elementType: "geometry.fill",
  stylers: [
    {
      visibility: "on"
    },
    {
      color: "#ffffff"
    },
    {
      weight: 1.8
    }
  ]
},
{
  featureType: "road.local",
  elementType: "geometry.stroke",
  stylers: [
    {
      color: "#d7d7d7"
    }
  ]
},
{
  featureType: "poi",
  elementType: "geometry.fill",
  stylers: [
    {
      visibility: "on"
    },
    {
      color: "#ebebeb"
    }
  ]
},
{
  featureType: "administrative",
  elementType: "geometry",
```

```
  stylers: [
    {
      color: "#a7a7a7"
    }
  ]
},
{
  featureType: "road.arterial",
  elementType: "geometry.fill",
  stylers: [
    {
      color: "#ffffff"
    }
  ]
},
{
  featureType: "road.arterial",
  elementType: "geometry.fill",
  stylers: [
    {
      color: "#ffffff"
    }
  ]
},
{
  featureType: "landscape",
  elementType: "geometry.fill",
  stylers: [
    {
      visibility: "on"
    },
    {
      color: "#efefef"
    }
  ]
},
{
  featureType: "road",
  elementType: "labels.text.fill",
  stylers: [
    {
      color: "#696969"
    }
  ]
},
{
  featureType: "administrative",
  elementType: "labels.text.fill",
  stylers: [
    {
      visibility: "on"
    },
```

```
      {
        color: "#737373"
      }
    ]
  },
  {
    featureType: "poi",
    elementType: "labels.icon",
    stylers: [
      {
        visibility: "off"
      }
    ]
  },
  {
    featureType: "poi",
    elementType: "labels",
    stylers: [
      {
        visibility: "off"
      }
    ]
  },
  {
    featureType: "road.arterial",
    elementType: "geometry.stroke",
    stylers: [
      {
        color: "#d6d6d6"
      }
    ]
  },
  {
    featureType: "road",
    elementType: "labels.icon",
    stylers: [
      {
        visibility: "off"
      }
    ]
  },
  {},
  {
    featureType: "poi",
    elementType: "geometry.fill",
    stylers: [
      {
        color: "#dadada"
      }
    ]
  }
]
```

```
      });
    }



});
```

CountDuwn. js

```
function getTimeRemaining(endtime) {
  var t = Date.parse(endtime) - Date.parse(new Date());
  var seconds = Math.floor((t / 1000) % 60);
  var minutes = Math.floor((t / 1000 / 60) % 60);
  var hours = Math.floor((t / (1000 * 60 * 60)) % 24);
  var days = Math.floor(t / (1000 * 60 * 60 * 24));
  return {
    'total': t,
    'days': days,
    'hours': hours,
    'minutes': minutes,
    'seconds': seconds
  };
}

function initializeClock(id, endtime) {
  var clock = document.getElementById(id);
  var daysSpan = clock.querySelector('.days');
  var hoursSpan = clock.querySelector('.hours');
  var minutesSpan = clock.querySelector('.minutes');
  var secondsSpan = clock.querySelector('.seconds');

  function updateClock() {
    var t = getTimeRemaining(endtime);

    daysSpan.innerHTML = t.days;
    hoursSpan.innerHTML = ('0' + t.hours).slice(-2);
    minutesSpan.innerHTML = ('0' + t.minutes).slice(-2);
    secondsSpan.innerHTML = ('0' + t.seconds).slice(-2);

    if (t.total <= 0) {
      clearInterval(timeinterval);
    }
  }

  updateClock();
  var timeinterval = setInterval(updateClock, 1000);
}
```

```
var deadline = new Date(Date.parse(new Date()) + 30 * 24 * 60 * 60 * 1000);
initializeClock('clockdiv', deadline);
```

Ion.rangeSlider.js

```
;(function(factory) {
    if (typeof define === "function" && define.amd) {
        define(["jquery"], function (jQuery) {
            return factory(jQuery, document, window, navigator);
        });
    } else if (typeof exports === "object") {
        factory(require("jquery"), document, window, navigator);
    } else {
        factory(jQuery, document, window, navigator);
    }
} (function ($, document, window, navigator, undefined) {
    "use strict";

    var plugin_count = 0;

    // IE8 fix
    var is_old_ie = (function () {
        var n = navigator.userAgent,
            r = /msie\s\d+/i,
            v;
        if (n.search(r) > 0) {
            v = r.exec(n).toString();
            v = v.split(" ")[1];
            if (v < 9) {
                $("html").addClass("lt-ie9");
                return true;
            }
        }
        return false;
    } ());
    if (!Function.prototype.bind) {
        Function.prototype.bind = function bind(that) {

            var target = this;
            var slice = [].slice;

            if (typeof target != "function") {
                throw new TypeError();
            }
```

```
        var args = slice.call(arguments, 1),
            bound = function () {

                if (this instanceof bound) {

                    var F = function(){};
                    F.prototype = target.prototype;
                    var self = new F();

                    var result = target.apply(
                        self,
                        args.concat(slice.call(arguments))
                    );
                    if (Object(result) === result) {
                        return result;
                    }
                    return self;

                } else {

                    return target.apply(
                        that,
                        args.concat(slice.call(arguments))
                    );

                }

            };

        return bound;
    };
}
if (!Array.prototype.indexOf) {
    Array.prototype.indexOf = function(searchElement, fromIndex) {
        var k;
        if (this == null) {
            throw new TypeError('"this" is null or not defined');
        }
        var O = Object(this);
        var len = O.length >>> 0;
        if (len === 0) {
            return -1;
        }
        var n = +fromIndex || 0;
        if (Math.abs(n) === Infinity) {
            n = 0;
        }
        if (n >= len) {
            return -1;
        }
        k = Math.max(n >= 0 ? n : len - Math.abs(n), 0);
```

```
            while (k < len) {
                if (k in O && O[k] === searchElement) {
                    return k;
                }
                k++;
            }
            return -1;
        };
    }

    var base_html =
        '<span class="irs">' +
        '<span class="irs-line" tabindex="0"><span class="irs-line-left"></span><span class="irs-line-
mid"></span><span class="irs-line-right"></span></span>' +
        '<span class="irs-min">0</span><span class="irs-max">1</span>' +
        '<span class="irs-from">0</span><span class="irs-to">0</span><span class="irs-
single">0</span>' +
        '</span>' +
        '<span class="irs-grid"></span>' +
        '<span class="irs-bar"></span>';

    var single_html =
        '<span class="irs-bar-edge"></span>' +
        '<span class="irs-shadow shadow-single"></span>' +
        '<span class="irs-slider single"></span>';

    var double_html =
        '<span class="irs-shadow shadow-from"></span>' +
        '<span class="irs-shadow shadow-to"></span>' +
        '<span class="irs-slider from"></span>' +
        '<span class="irs-slider to"></span>';

    var disable_html =
        '<span class="irs-disable-mask"></span>';


    var IonRangeSlider = function (input, options, plugin_count) {
        this.VERSION = "2.2.0";
        this.input = input;
        this.plugin_count = plugin_count;
        this.current_plugin = 0;
        this.calc_count = 0;
        this.update_tm = 0;
        this.old_from = 0;
        this.old_to = 0;
        this.old_min_interval = null;
        this.raf_id = null;
        this.dragging = false;
        this.force_redraw = false;
        this.no_diapason = false;
        this.has_tab_index = true;
        this.is_key = false;
```

```
this.is_update = false;
this.is_start = true;
this.is_finish = false;
this.is_active = false;
this.is_resize = false;
this.is_click = false;

options = options || {};

// cache for links to all DOM elements
this.$cache = {
    win: $(window),
    body: $(document.body),
    input: $(input),
    cont: null,
    rs: null,
    min: null,
    max: null,
    from: null,
    to: null,
    single: null,
    bar: null,
    line: null,
    s_single: null,
    s_from: null,
    s_to: null,
    shad_single: null,
    shad_from: null,
    shad_to: null,
    edge: null,
    grid: null,
    grid_labels: []
};

// storage for measure variables
this.coords = {
    // left
    x_gap: 0,
    x_pointer: 0,

    // width
    w_rs: 0,
    w_rs_old: 0,
    w_handle: 0,

    // percents
    p_gap: 0,
    p_gap_left: 0,
    p_gap_right: 0,
    p_step: 0,
    p_pointer: 0,
    p_handle: 0,
```

```
    p_single_fake: 0,
    p_single_real: 0,
    p_from_fake: 0,
    p_from_real: 0,
    p_to_fake: 0,
    p_to_real: 0,
    p_bar_x: 0,
    p_bar_w: 0,

    // grid
    grid_gap: 0,
    big_num: 0,
    big: [],
    big_w: [],
    big_p: [],
    big_x: []
};

// storage for labels measure variables
this.labels = {
    // width
    w_min: 0,
    w_max: 0,
    w_from: 0,
    w_to: 0,
    w_single: 0,

    // percents
    p_min: 0,
    p_max: 0,
    p_from_fake: 0,
    p_from_left: 0,
    p_to_fake: 0,
    p_to_left: 0,
    p_single_fake: 0,
    p_single_left: 0
};


var $inp = this.$cache.input,
    val = $inp.prop("value"),
    config, config_from_data, prop;

// default config
config = {
    type: "single",

    min: 10,
    max: 100,
    from: null,
    to: null,
    step: 1,
```

```
min_interval: 0,
max_interval: 0,
drag_interval: false,

values: [],
p_values: [],

from_fixed: false,
from_min: null,
from_max: null,
from_shadow: false,

to_fixed: false,
to_min: null,
to_max: null,
to_shadow: false,

prettify_enabled: true,
prettify_separator: " ",
prettify: null,

force_edges: false,

keyboard: true,

grid: false,
grid_margin: true,
grid_num: 4,
grid_snap: false,

hide_min_max: false,
hide_from_to: false,

prefix: "",
postfix: "",
max_postfix: "",
decorate_both: true,
values_separator: " — ",

input_values_separator: ";",

disable: false,
block: false,

extra_classes: "",

scope: null,
onStart: null,
onChange: null,
onFinish: null,
onUpdate: null
```

```
};


// check if base element is input
if ($inp[0].nodeName !== "INPUT") {
    console && console.warn && console.warn("Base element should be <input>!", $inp[0]);
}


// config from data-attributes extends js config
config_from_data = {
    type: $inp.data("type"),

    min: $inp.data("min"),
    max: $inp.data("max"),
    from: $inp.data("from"),
    to: $inp.data("to"),
    step: $inp.data("step"),

    min_interval: $inp.data("minInterval"),
    max_interval: $inp.data("maxInterval"),
    drag_interval: $inp.data("dragInterval"),

    values: $inp.data("values"),

    from_fixed: $inp.data("fromFixed"),
    from_min: $inp.data("fromMin"),
    from_max: $inp.data("fromMax"),
    from_shadow: $inp.data("fromShadow"),

    to_fixed: $inp.data("toFixed"),
    to_min: $inp.data("toMin"),
    to_max: $inp.data("toMax"),
    to_shadow: $inp.data("toShadow"),

    prettify_enabled: $inp.data("prettifyEnabled"),
    prettify_separator: $inp.data("prettifySeparator"),

    force_edges: $inp.data("forceEdges"),

    keyboard: $inp.data("keyboard"),

    grid: $inp.data("grid"),
    grid_margin: $inp.data("gridMargin"),
    grid_num: $inp.data("gridNum"),
    grid_snap: $inp.data("gridSnap"),

    hide_min_max: $inp.data("hideMinMax"),
    hide_from_to: $inp.data("hideFromTo"),

    prefix: $inp.data("prefix"),
    postfix: $inp.data("postfix"),
```

```
            max_postfix: $inp.data("maxPostfix"),
            decorate_both: $inp.data("decorateBoth"),
            values_separator: $inp.data("valuesSeparator"),

            input_values_separator: $inp.data("inputValuesSeparator"),

            disable: $inp.data("disable"),
            block: $inp.data("block"),

            extra_classes: $inp.data("extraClasses"),
        };
        config_from_data.values = config_from_data.values && config_from_data.values.split(",");

        for (prop in config_from_data) {
            if (config_from_data.hasOwnProperty(prop)) {
                if (config_from_data[prop] === undefined || config_from_data[prop] === "") {
                    delete config_from_data[prop];
                }
            }
        }


        // input value extends default config
        if (val !== undefined && val !== "") {
            val = val.split(config_from_data.input_values_separator || options.input_values_separator ||
";");

            if (val[0] && val[0] == +val[0]) {
                val[0] = +val[0];
            }
            if (val[1] && val[1] == +val[1]) {
                val[1] = +val[1];
            }

            if (options && options.values && options.values.length) {
                config.from = val[0] && options.values.indexOf(val[0]);
                config.to = val[1] && options.values.indexOf(val[1]);
            } else {
                config.from = val[0] && +val[0];
                config.to = val[1] && +val[1];
            }
        }



        // js config extends default config
        $.extend(config, options);


        // data config extends config
        $.extend(config, config_from_data);
        this.options = config;
```

```
    // validate config, to be sure that all data types are correct
    this.update_check = {};
    this.validate();



    // default result object, returned to callbacks
    this.result = {
        input: this.$cache.input,
        slider: null,

        min: this.options.min,
        max: this.options.max,

        from: this.options.from,
        from_percent: 0,
        from_value: null,

        to: this.options.to,
        to_percent: 0,
        to_value: null
    };



    this.init();
};

IonRangeSlider.prototype = {

    init: function (is_update) {
        this.no_diapason = false;
        this.coords.p_step = this.convertToPercent(this.options.step, true);

        this.target = "base";

        this.toggleInput();
        this.append();
        this.setMinMax();

        if (is_update) {
            this.force_redraw = true;
            this.calc(true);

            // callbacks called
            this.callOnUpdate();
        } else {
            this.force_redraw = true;
            this.calc(true);
```

```
            // callbacks called
            this.callOnStart();
        }

        this.updateScene();
    },

    append: function () {
        var container_html = '<span class="irs js-irs-' + this.plugin_count + ' ' +
this.options.extra_classes + '"></span>';
        this.$cache.input.before(container_html);
        this.$cache.input.prop("readonly", true);
        this.$cache.cont = this.$cache.input.prev();
        this.result.slider = this.$cache.cont;

        this.$cache.cont.html(base_html);
        this.$cache.rs = this.$cache.cont.find(".irs");
        this.$cache.min = this.$cache.cont.find(".irs-min");
        this.$cache.max = this.$cache.cont.find(".irs-max");
        this.$cache.from = this.$cache.cont.find(".irs-from");
        this.$cache.to = this.$cache.cont.find(".irs-to");
        this.$cache.single = this.$cache.cont.find(".irs-single");
        this.$cache.bar = this.$cache.cont.find(".irs-bar");
        this.$cache.line = this.$cache.cont.find(".irs-line");
        this.$cache.grid = this.$cache.cont.find(".irs-grid");

        if (this.options.type === "single") {
            this.$cache.cont.append(single_html);
            this.$cache.edge = this.$cache.cont.find(".irs-bar-edge");
            this.$cache.s_single = this.$cache.cont.find(".single");
            this.$cache.from[0].style.visibility = "hidden";
            this.$cache.to[0].style.visibility = "hidden";
            this.$cache.shad_single = this.$cache.cont.find(".shadow-single");
        } else {
            this.$cache.cont.append(double_html);
            this.$cache.s_from = this.$cache.cont.find(".from");
            this.$cache.s_to = this.$cache.cont.find(".to");
            this.$cache.shad_from = this.$cache.cont.find(".shadow-from");
            this.$cache.shad_to = this.$cache.cont.find(".shadow-to");

            this.setTopHandler();
        }

        if (this.options.hide_from_to) {
            this.$cache.from[0].style.display = "none";
            this.$cache.to[0].style.display = "none";
            this.$cache.single[0].style.display = "none";
        }

        this.appendGrid();
```

```
    if (this.options.disable) {
        this.appendDisableMask();
        this.$cache.input[0].disabled = true;
    } else {
        this.$cache.input[0].disabled = false;
        this.removeDisableMask();
        this.bindEvents();
    }

    // block only if not disabled
    if (!this.options.disable) {
        if (this.options.block) {
            this.appendDisableMask();
        } else {
            this.removeDisableMask();
        }
    }

    if (this.options.drag_interval) {
        this.$cache.bar[0].style.cursor = "ew-resize";
    }
},

setTopHandler: function () {
    var min = this.options.min,
        max = this.options.max,
        from = this.options.from,
        to = this.options.to;

    if (from > min && to === max) {
        this.$cache.s_from.addClass("type_last");
    } else if (to < max) {
        this.$cache.s_to.addClass("type_last");
    }
},

changeLevel: function (target) {
    switch (target) {
        case "single":
            this.coords.p_gap = this.toFixed(this.coords.p_pointer - this.coords.p_single_fake);
            this.$cache.s_single.addClass("state_hover");
            break;
        case "from":
            this.coords.p_gap = this.toFixed(this.coords.p_pointer - this.coords.p_from_fake);
            this.$cache.s_from.addClass("state_hover");
            this.$cache.s_from.addClass("type_last");
            this.$cache.s_to.removeClass("type_last");
            break;
        case "to":
            this.coords.p_gap = this.toFixed(this.coords.p_pointer - this.coords.p_to_fake);
            this.$cache.s_to.addClass("state_hover");
            this.$cache.s_to.addClass("type_last");
```

```
            this.$cache.s_from.removeClass("type_last");
            break;
        case "both":
            this.coords.p_gap_left = this.toFixed(this.coords.p_pointer - this.coords.p_from_fake);
            this.coords.p_gap_right = this.toFixed(this.coords.p_to_fake - this.coords.p_pointer);
            this.$cache.s_to.removeClass("type_last");
            this.$cache.s_from.removeClass("type_last");
            break;
    }
},

appendDisableMask: function () {
    this.$cache.cont.append(disable_html);
    this.$cache.cont.addClass("irs-disabled");
},

removeDisableMask: function () {
    this.$cache.cont.remove(".irs-disable-mask");
    this.$cache.cont.removeClass("irs-disabled");
},

remove: function () {
    this.$cache.cont.remove();
    this.$cache.cont = null;

    this.$cache.line.off("keydown.irs_" + this.plugin_count);

    this.$cache.body.off("touchmove.irs_" + this.plugin_count);
    this.$cache.body.off("mousemove.irs_" + this.plugin_count);

    this.$cache.win.off("touchend.irs_" + this.plugin_count);
    this.$cache.win.off("mouseup.irs_" + this.plugin_count);

    if (is_old_ie) {
        this.$cache.body.off("mouseup.irs_" + this.plugin_count);
        this.$cache.body.off("mouseleave.irs_" + this.plugin_count);
    }

    this.$cache.grid_labels = [];
    this.coords.big = [];
    this.coords.big_w = [];
    this.coords.big_p = [];
    this.coords.big_x = [];

    cancelAnimationFrame(this.raf_id);
},

bindEvents: function () {
    if (this.no_diapason) {
        return;
    }
```

```
        this.$cache.body.on("touchmove.irs_" + this.plugin_count, this.pointerMove.bind(this));
        this.$cache.body.on("mousemove.irs_" + this.plugin_count, this.pointerMove.bind(this));

        this.$cache.win.on("touchend.irs_" + this.plugin_count, this.pointerUp.bind(this));
        this.$cache.win.on("mouseup.irs_" + this.plugin_count, this.pointerUp.bind(this));

        this.$cache.line.on("touchstart.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));
        this.$cache.line.on("mousedown.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));

        this.$cache.line.on("focus.irs_" + this.plugin_count, this.pointerFocus.bind(this));

        if (this.options.drag_interval && this.options.type === "double") {
            this.$cache.bar.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"both"));
            this.$cache.bar.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"both"));
        } else {
            this.$cache.bar.on("touchstart.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));
            this.$cache.bar.on("mousedown.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));
        }

        if (this.options.type === "single") {
            this.$cache.single.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"single"));
            this.$cache.s_single.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"single"));
            this.$cache.shad_single.on("touchstart.irs_" + this.plugin_count,
this.pointerClick.bind(this, "click"));

            this.$cache.single.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"single"));
            this.$cache.s_single.on("mousedown.irs_" + this.plugin_count,
this.pointerDown.bind(this, "single"));
            this.$cache.edge.on("mousedown.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));
            this.$cache.shad_single.on("mousedown.irs_" + this.plugin_count,
this.pointerClick.bind(this, "click"));
        } else {
            this.$cache.single.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
null));
            this.$cache.single.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
null));

            this.$cache.from.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"from"));
            this.$cache.s_from.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"from"));
            this.$cache.to.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this, "to"));
```

```
            this.$cache.s_to.on("touchstart.irs_" + this.plugin_count, this.pointerDown.bind(this,
"to"));
            this.$cache.shad_from.on("touchstart.irs_" + this.plugin_count,
this.pointerClick.bind(this, "click"));
            this.$cache.shad_to.on("touchstart.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));

            this.$cache.from.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"from"));
            this.$cache.s_from.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"from"));
            this.$cache.to.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"to"));
            this.$cache.s_to.on("mousedown.irs_" + this.plugin_count, this.pointerDown.bind(this,
"to"));
            this.$cache.shad_from.on("mousedown.irs_" + this.plugin_count,
this.pointerClick.bind(this, "click"));
            this.$cache.shad_to.on("mousedown.irs_" + this.plugin_count, this.pointerClick.bind(this,
"click"));
        }

        if (this.options.keyboard) {
            this.$cache.line.on("keydown.irs_" + this.plugin_count, this.key.bind(this, "keyboard"));
        }

        if (is_old_ie) {
            this.$cache.body.on("mouseup.irs_" + this.plugin_count, this.pointerUp.bind(this));
            this.$cache.body.on("mouseleave.irs_" + this.plugin_count, this.pointerUp.bind(this));
        }
    },


    pointerFocus: function (e) {
        if (!this.target) {
            var x;
            var $handle;

            if (this.options.type === "single") {
                $handle = this.$cache.single;
            } else {
                $handle = this.$cache.from;
            }

            x = $handle.offset().left;
            x += ($handle.width() / 2) - 1;

            this.pointerClick("single", {preventDefault: function () {}, pageX: x});
        }
    },

    pointerMove: function (e) {
        if (!this.dragging) {
```

```
        return;
    }

    var x = e.pageX || e.originalEvent.touches && e.originalEvent.touches[0].pageX;
    this.coords.x_pointer = x - this.coords.x_gap;

    this.calc();
},

pointerUp: function (e) {
    if (this.current_plugin !== this.plugin_count) {
        return;
    }

    if (this.is_active) {
        this.is_active = false;
    } else {
        return;
    }

    this.$cache.cont.find(".state_hover").removeClass("state_hover");

    this.force_redraw = true;

    if (is_old_ie) {
        $("*").prop("unselectable", false);
    }

    this.updateScene();
    this.restoreOriginalMinInterval();

    // callbacks call
    if ($.contains(this.$cache.cont[0], e.target) || this.dragging) {
        this.callOnFinish();
    }

    this.dragging = false;
},


pointerDown: function (target, e) {
    e.preventDefault();
    var x = e.pageX || e.originalEvent.touches && e.originalEvent.touches[0].pageX;
    if (e.button === 2) {
        return;
    }

    if (target === "both") {
        this.setTempMinInterval();
    }

    if (!target) {
```

```
            target = this.target || "from";
        }

        this.current_plugin = this.plugin_count;
        this.target = target;

        this.is_active = true;
        this.dragging = true;

        this.coords.x_gap = this.$cache.rs.offset().left;
        this.coords.x_pointer = x - this.coords.x_gap;

        this.calcPointerPercent();
        this.changeLevel(target);

        if (is_old_ie) {
            $("*").prop("unselectable", true);
        }

        this.$cache.line.trigger("focus");

        this.updateScene();
    },


    pointerClick: function (target, e) {
        e.preventDefault();
        var x = e.pageX || e.originalEvent.touches && e.originalEvent.touches[0].pageX;
        if (e.button === 2) {
            return;
        }

        this.current_plugin = this.plugin_count;
        this.target = target;

        this.is_click = true;
        this.coords.x_gap = this.$cache.rs.offset().left;
        this.coords.x_pointer = +(x - this.coords.x_gap).toFixed();

        this.force_redraw = true;
        this.calc();

        this.$cache.line.trigger("focus");
    },

    key: function (target, e) {
        if (this.current_plugin !== this.plugin_count || e.altKey || e.ctrlKey || e.shiftKey || e.metaKey)
{
            return;
        }

        switch (e.which) {
```

```
            case 83: // W
            case 65: // A
            case 40: // DOWN
            case 37: // LEFT
                e.preventDefault();
                this.moveByKey(false);
                break;

            case 87: // S
            case 68: // D
            case 38: // UP
            case 39: // RIGHT
                e.preventDefault();
                this.moveByKey(true);
                break;
        }

        return true;
    },

    moveByKey: function (right) {
        var p = this.coords.p_pointer;
        var p_step = (this.options.max - this.options.min) / 100;
        p_step = this.options.step / p_step;

        if (right) {
            p += p_step;
        } else {
            p -= p_step;
        }

        this.coords.x_pointer = this.toFixed(this.coords.w_rs / 100 * p);
        this.is_key = true;
        this.calc();
    },


    setMinMax: function () {
        if (!this.options) {
            return;
        }

        if (this.options.hide_min_max) {
            this.$cache.min[0].style.display = "none";
            this.$cache.max[0].style.display = "none";
            return;
        }

        if (this.options.values.length) {
            this.$cache.min.html(this.decorate(this.options.p_values[this.options.min]));
            this.$cache.max.html(this.decorate(this.options.p_values[this.options.max]));
        } else {
```

```
            var min_pretty = this._prettify(this.options.min);
            var max_pretty = this._prettify(this.options.max);

            this.result.min_pretty = min_pretty;
            this.result.max_pretty = max_pretty;

            this.$cache.min.html(this.decorate(min_pretty, this.options.min));
            this.$cache.max.html(this.decorate(max_pretty, this.options.max));
        }

        this.labels.w_min = this.$cache.min.outerWidth(false);
        this.labels.w_max = this.$cache.max.outerWidth(false);
    },

    setTempMinInterval: function () {
        var interval = this.result.to - this.result.from;

        if (this.old_min_interval === null) {
            this.old_min_interval = this.options.min_interval;
        }

        this.options.min_interval = interval;
    },

    restoreOriginalMinInterval: function () {
        if (this.old_min_interval !== null) {
            this.options.min_interval = this.old_min_interval;
            this.old_min_interval = null;
        }
    },


    calc: function (update) {
        if (!this.options) {
            return;
        }

        this.calc_count++;

        if (this.calc_count === 10 || update) {
            this.calc_count = 0;
            this.coords.w_rs = this.$cache.rs.outerWidth(false);

            this.calcHandlePercent();
        }

        if (!this.coords.w_rs) {
            return;
        }

        this.calcPointerPercent();
        var handle_x = this.getHandleX();
```

```
        if (this.target === "both") {
            this.coords.p_gap = 0;
            handle_x = this.getHandleX();
        }

        if (this.target === "click") {
            this.coords.p_gap = this.coords.p_handle / 2;
            handle_x = this.getHandleX();

            if (this.options.drag_interval) {
                this.target = "both_one";
            } else {
                this.target = this.chooseHandle(handle_x);
            }
        }

        switch (this.target) {
            case "base":
                var w = (this.options.max - this.options.min) / 100,
                    f = (this.result.from - this.options.min) / w,
                    t = (this.result.to - this.options.min) / w;

                this.coords.p_single_real = this.toFixed(f);
                this.coords.p_from_real = this.toFixed(f);
                this.coords.p_to_real = this.toFixed(t);

                this.coords.p_single_real = this.checkDiapason(this.coords.p_single_real,
this.options.from_min, this.options.from_max);
                this.coords.p_from_real = this.checkDiapason(this.coords.p_from_real,
this.options.from_min, this.options.from_max);
                this.coords.p_to_real = this.checkDiapason(this.coords.p_to_real, this.options.to_min,
this.options.to_max);

                this.coords.p_single_fake = this.convertToFakePercent(this.coords.p_single_real);
                this.coords.p_from_fake = this.convertToFakePercent(this.coords.p_from_real);
                this.coords.p_to_fake = this.convertToFakePercent(this.coords.p_to_real);

                this.target = null;

                break;

            case "single":
                if (this.options.from_fixed) {
                    break;
                }

                this.coords.p_single_real = this.convertToRealPercent(handle_x);
                this.coords.p_single_real = this.calcWithStep(this.coords.p_single_real);
                this.coords.p_single_real = this.checkDiapason(this.coords.p_single_real,
this.options.from_min, this.options.from_max);
```

```
                this.coords.p_single_fake = this.convertToFakePercent(this.coords.p_single_real);

                break;

            case "from":
                if (this.options.from_fixed) {
                    break;
                }

                this.coords.p_from_real = this.convertToRealPercent(handle_x);
                this.coords.p_from_real = this.calcWithStep(this.coords.p_from_real);
                if (this.coords.p_from_real > this.coords.p_to_real) {
                    this.coords.p_from_real = this.coords.p_to_real;
                }
                this.coords.p_from_real = this.checkDiapason(this.coords.p_from_real,
this.options.from_min, this.options.from_max);
                this.coords.p_from_real = this.checkMinInterval(this.coords.p_from_real,
this.coords.p_to_real, "from");
                this.coords.p_from_real = this.checkMaxInterval(this.coords.p_from_real,
this.coords.p_to_real, "from");

                this.coords.p_from_fake = this.convertToFakePercent(this.coords.p_from_real);

                break;

            case "to":
                if (this.options.to_fixed) {
                    break;
                }

                this.coords.p_to_real = this.convertToRealPercent(handle_x);
                this.coords.p_to_real = this.calcWithStep(this.coords.p_to_real);
                if (this.coords.p_to_real < this.coords.p_from_real) {
                    this.coords.p_to_real = this.coords.p_from_real;
                }
                this.coords.p_to_real = this.checkDiapason(this.coords.p_to_real, this.options.to_min,
this.options.to_max);
                this.coords.p_to_real = this.checkMinInterval(this.coords.p_to_real,
this.coords.p_from_real, "to");
                this.coords.p_to_real = this.checkMaxInterval(this.coords.p_to_real,
this.coords.p_from_real, "to");

                this.coords.p_to_fake = this.convertToFakePercent(this.coords.p_to_real);

                break;

            case "both":
                if (this.options.from_fixed || this.options.to_fixed) {
                    break;
                }
```

```
                handle_x = this.toFixed(handle_x + (this.coords.p_handle * 0.001));

                this.coords.p_from_real = this.convertToRealPercent(handle_x) -
this.coords.p_gap_left;
                this.coords.p_from_real = this.calcWithStep(this.coords.p_from_real);
                this.coords.p_from_real = this.checkDiapason(this.coords.p_from_real,
this.options.from_min, this.options.from_max);
                this.coords.p_from_real = this.checkMinInterval(this.coords.p_from_real,
this.coords.p_to_real, "from");
                this.coords.p_from_fake = this.convertToFakePercent(this.coords.p_from_real);

                this.coords.p_to_real = this.convertToRealPercent(handle_x) +
this.coords.p_gap_right;
                this.coords.p_to_real = this.calcWithStep(this.coords.p_to_real);
                this.coords.p_to_real = this.checkDiapason(this.coords.p_to_real, this.options.to_min,
this.options.to_max);
                this.coords.p_to_real = this.checkMinInterval(this.coords.p_to_real,
this.coords.p_from_real, "to");
                this.coords.p_to_fake = this.convertToFakePercent(this.coords.p_to_real);

                break;

            case "both_one":
                if (this.options.from_fixed || this.options.to_fixed) {
                    break;
                }

                var real_x = this.convertToRealPercent(handle_x),
                    from = this.result.from_percent,
                    to = this.result.to_percent,
                    full = to - from,
                    half = full / 2,
                    new_from = real_x - half,
                    new_to = real_x + half;

                if (new_from < 0) {
                    new_from = 0;
                    new_to = new_from + full;
                }

                if (new_to > 100) {
                    new_to = 100;
                    new_from = new_to - full;
                }

                this.coords.p_from_real = this.calcWithStep(new_from);
                this.coords.p_from_real = this.checkDiapason(this.coords.p_from_real,
this.options.from_min, this.options.from_max);
                this.coords.p_from_fake = this.convertToFakePercent(this.coords.p_from_real);

                this.coords.p_to_real = this.calcWithStep(new_to);
```

```
                this.coords.p_to_real = this.checkDiapason(this.coords.p_to_real, this.options.to_min,
this.options.to_max);
                this.coords.p_to_fake = this.convertToFakePercent(this.coords.p_to_real);

                break;
            }

            if (this.options.type === "single") {
                this.coords.p_bar_x = (this.coords.p_handle / 2);
                this.coords.p_bar_w = this.coords.p_single_fake;

                this.result.from_percent = this.coords.p_single_real;
                this.result.from = this.convertToValue(this.coords.p_single_real);
                this.result.from_pretty = this._prettify(this.result.from);

                if (this.options.values.length) {
                    this.result.from_value = this.options.values[this.result.from];
                }
            } else {
                this.coords.p_bar_x = this.toFixed(this.coords.p_from_fake + (this.coords.p_handle / 2));
                this.coords.p_bar_w = this.toFixed(this.coords.p_to_fake - this.coords.p_from_fake);

                this.result.from_percent = this.coords.p_from_real;
                this.result.from = this.convertToValue(this.coords.p_from_real);
                this.result.from_pretty = this._prettify(this.result.from);
                this.result.to_percent = this.coords.p_to_real;
                this.result.to = this.convertToValue(this.coords.p_to_real);
                this.result.to_pretty = this._prettify(this.result.to);

                if (this.options.values.length) {
                    this.result.from_value = this.options.values[this.result.from];
                    this.result.to_value = this.options.values[this.result.to];
                }
            }

            this.calcMinMax();
            this.calcLabels();
        },


        /**
         * calculates pointer X in percent
         */
        calcPointerPercent: function () {
            if (!this.coords.w_rs) {
                this.coords.p_pointer = 0;
                return;
            }

            if (this.coords.x_pointer < 0 || isNaN(this.coords.x_pointer)  ) {
                this.coords.x_pointer = 0;
            } else if (this.coords.x_pointer > this.coords.w_rs) {
```

```
            this.coords.x_pointer = this.coords.w_rs;
        }

        this.coords.p_pointer = this.toFixed(this.coords.x_pointer / this.coords.w_rs * 100);
    },

    convertToRealPercent: function (fake) {
        var full = 100 - this.coords.p_handle;
        return fake / full * 100;
    },

    convertToFakePercent: function (real) {
        var full = 100 - this.coords.p_handle;
        return real / 100 * full;
    },

    getHandleX: function () {
        var max = 100 - this.coords.p_handle,
            x = this.toFixed(this.coords.p_pointer - this.coords.p_gap);

        if (x < 0) {
            x = 0;
        } else if (x > max) {
            x = max;
        }

        return x;
    },

    calcHandlePercent: function () {
        if (this.options.type === "single") {
            this.coords.w_handle = this.$cache.s_single.outerWidth(false);
        } else {
            this.coords.w_handle = this.$cache.s_from.outerWidth(false);
        }

        this.coords.p_handle = this.toFixed(this.coords.w_handle / this.coords.w_rs * 100);
    },


    chooseHandle: function (real_x) {
        if (this.options.type === "single") {
            return "single";
        } else {
            var m_point = this.coords.p_from_real + ((this.coords.p_to_real -
this.coords.p_from_real) / 2);
            if (real_x >= m_point) {
                return this.options.to_fixed ? "from" : "to";
            } else {
                return this.options.from_fixed ? "to" : "from";
            }
        }
```

```
    },

    /**
     * Measure Min and Max labels width in percent
     */
    calcMinMax: function () {
        if (!this.coords.w_rs) {
            return;
        }

        this.labels.p_min = this.labels.w_min / this.coords.w_rs * 100;
        this.labels.p_max = this.labels.w_max / this.coords.w_rs * 100;
    },

    /**
     * Measure labels width and X in percent
     */
    calcLabels: function () {
        if (!this.coords.w_rs || this.options.hide_from_to) {
            return;
        }

        if (this.options.type === "single") {

            this.labels.w_single = this.$cache.single.outerWidth(false);
            this.labels.p_single_fake = this.labels.w_single / this.coords.w_rs * 100;
            this.labels.p_single_left = this.coords.p_single_fake + (this.coords.p_handle / 2) -
(this.labels.p_single_fake / 2);
            this.labels.p_single_left = this.checkEdges(this.labels.p_single_left,
this.labels.p_single_fake);

        } else {

            this.labels.w_from = this.$cache.from.outerWidth(false);
            this.labels.p_from_fake = this.labels.w_from / this.coords.w_rs * 100;
            this.labels.p_from_left = this.coords.p_from_fake + (this.coords.p_handle / 2) -
(this.labels.p_from_fake / 2);
            this.labels.p_from_left = this.toFixed(this.labels.p_from_left);
            this.labels.p_from_left = this.checkEdges(this.labels.p_from_left,
this.labels.p_from_fake);

            this.labels.w_to = this.$cache.to.outerWidth(false);
            this.labels.p_to_fake = this.labels.w_to / this.coords.w_rs * 100;
            this.labels.p_to_left = this.coords.p_to_fake + (this.coords.p_handle / 2) -
(this.labels.p_to_fake / 2);
            this.labels.p_to_left = this.toFixed(this.labels.p_to_left);
            this.labels.p_to_left = this.checkEdges(this.labels.p_to_left, this.labels.p_to_fake);

            this.labels.w_single = this.$cache.single.outerWidth(false);
            this.labels.p_single_fake = this.labels.w_single / this.coords.w_rs * 100;
            this.labels.p_single_left = ((this.labels.p_from_left + this.labels.p_to_left +
this.labels.p_to_fake) / 2) - (this.labels.p_single_fake / 2);
```

```
            this.labels.p_single_left = this.toFixed(this.labels.p_single_left);
            this.labels.p_single_left = this.checkEdges(this.labels.p_single_left,
this.labels.p_single_fake);

        }
    },



    /**
     * Main function called in request animation frame
     * to update everything
     */
    updateScene: function () {
        if (this.raf_id) {
            cancelAnimationFrame(this.raf_id);
            this.raf_id = null;
        }

        clearTimeout(this.update_tm);
        this.update_tm = null;

        if (!this.options) {
            return;
        }

        this.drawHandles();

        if (this.is_active) {
            this.raf_id = requestAnimationFrame(this.updateScene.bind(this));
        } else {
            this.update_tm = setTimeout(this.updateScene.bind(this), 300);
        }
    },


    drawHandles: function () {
        this.coords.w_rs = this.$cache.rs.outerWidth(false);

        if (!this.coords.w_rs) {
            return;
        }

        if (this.coords.w_rs !== this.coords.w_rs_old) {
            this.target = "base";
            this.is_resize = true;
        }

        if (this.coords.w_rs !== this.coords.w_rs_old || this.force_redraw) {
            this.setMinMax();
            this.calc(true);
            this.drawLabels();
```

```
            if (this.options.grid) {
                this.calcGridMargin();
                this.calcGridLabels();
            }
            this.force_redraw = true;
            this.coords.w_rs_old = this.coords.w_rs;
            this.drawShadow();
        }

        if (!this.coords.w_rs) {
            return;
        }

        if (!this.dragging && !this.force_redraw && !this.is_key) {
            return;
        }

        if (this.old_from !== this.result.from || this.old_to !== this.result.to || this.force_redraw ||
this.is_key) {

            this.drawLabels();

            this.$cache.bar[0].style.left = this.coords.p_bar_x + "%";
            this.$cache.bar[0].style.width = this.coords.p_bar_w + "%";

            if (this.options.type === "single") {
                this.$cache.s_single[0].style.left = this.coords.p_single_fake + "%";

                this.$cache.single[0].style.left = this.labels.p_single_left + "%";
            } else {
                this.$cache.s_from[0].style.left = this.coords.p_from_fake + "%";
                this.$cache.s_to[0].style.left = this.coords.p_to_fake + "%";

                if (this.old_from !== this.result.from || this.force_redraw) {
                    this.$cache.from[0].style.left = this.labels.p_from_left + "%";
                }
                if (this.old_to !== this.result.to || this.force_redraw) {
                    this.$cache.to[0].style.left = this.labels.p_to_left + "%";
                }

                this.$cache.single[0].style.left = this.labels.p_single_left + "%";
            }

            this.writeToInput();

            if ((this.old_from !== this.result.from || this.old_to !== this.result.to) && !this.is_start) {
                this.$cache.input.trigger("change");
                this.$cache.input.trigger("input");
            }

            this.old_from = this.result.from;
            this.old_to = this.result.to;
```

```
        // callbacks call
        if (!this.is_resize && !this.is_update && !this.is_start && !this.is_finish) {
            this.callOnChange();
        }
        if (this.is_key || this.is_click) {
            this.is_key = false;
            this.is_click = false;
            this.callOnFinish();
        }

        this.is_update = false;
        this.is_resize = false;
        this.is_finish = false;
    }

    this.is_start = false;
    this.is_key = false;
    this.is_click = false;
    this.force_redraw = false;
},

/**
 * Draw labels
 * measure labels collisions
 * collapse close labels
 */
drawLabels: function () {
    if (!this.options) {
        return;
    }

    var values_num = this.options.values.length;
    var p_values = this.options.p_values;
    var text_single;
    var text_from;
    var text_to;
    var from_pretty;
    var to_pretty;

    if (this.options.hide_from_to) {
        return;
    }

    if (this.options.type === "single") {

        if (values_num) {
            text_single = this.decorate(p_values[this.result.from]);
            this.$cache.single.html(text_single);
        } else {
            from_pretty = this._prettify(this.result.from);
```

```
                text_single = this.decorate(from_pretty, this.result.from);
                this.$cache.single.html(text_single);
            }

            this.calcLabels();

            if (this.labels.p_single_left < this.labels.p_min + 1) {
                this.$cache.min[0].style.visibility = "hidden";
            } else {
                this.$cache.min[0].style.visibility = "visible";
            }

            if (this.labels.p_single_left + this.labels.p_single_fake > 100 - this.labels.p_max - 1) {
                this.$cache.max[0].style.visibility = "hidden";
            } else {
                this.$cache.max[0].style.visibility = "visible";
            }

        } else {

            if (values_num) {

                if (this.options.decorate_both) {
                    text_single = this.decorate(p_values[this.result.from]);
                    text_single += this.options.values_separator;
                    text_single += this.decorate(p_values[this.result.to]);
                } else {
                    text_single = this.decorate(p_values[this.result.from] + this.options.values_separator
+ p_values[this.result.to]);
                }
                text_from = this.decorate(p_values[this.result.from]);
                text_to = this.decorate(p_values[this.result.to]);

                this.$cache.single.html(text_single);
                this.$cache.from.html(text_from);
                this.$cache.to.html(text_to);

            } else {
                from_pretty = this._prettify(this.result.from);
                to_pretty = this._prettify(this.result.to);

                if (this.options.decorate_both) {
                    text_single = this.decorate(from_pretty, this.result.from);
                    text_single += this.options.values_separator;
                    text_single += this.decorate(to_pretty, this.result.to);
                } else {
                    text_single = this.decorate(from_pretty + this.options.values_separator + to_pretty,
this.result.to);
                }
                text_from = this.decorate(from_pretty, this.result.from);
                text_to = this.decorate(to_pretty, this.result.to);
```

```
    this.$cache.single.html(text_single);
    this.$cache.from.html(text_from);
    this.$cache.to.html(text_to);

}

this.calcLabels();

var min = Math.min(this.labels.p_single_left, this.labels.p_from_left),
    single_left = this.labels.p_single_left + this.labels.p_single_fake,
    to_left = this.labels.p_to_left + this.labels.p_to_fake,
    max = Math.max(single_left, to_left);

if (this.labels.p_from_left + this.labels.p_from_fake >= this.labels.p_to_left) {
    this.$cache.from[0].style.visibility = "hidden";
    this.$cache.to[0].style.visibility = "hidden";
    this.$cache.single[0].style.visibility = "visible";

    if (this.result.from === this.result.to) {
        if (this.target === "from") {
            this.$cache.from[0].style.visibility = "visible";
        } else if (this.target === "to") {
            this.$cache.to[0].style.visibility = "visible";
        } else if (!this.target) {
            this.$cache.from[0].style.visibility = "visible";
        }
        this.$cache.single[0].style.visibility = "hidden";
        max = to_left;
    } else {
        this.$cache.from[0].style.visibility = "hidden";
        this.$cache.to[0].style.visibility = "hidden";
        this.$cache.single[0].style.visibility = "visible";
        max = Math.max(single_left, to_left);
    }
} else {
    this.$cache.from[0].style.visibility = "visible";
    this.$cache.to[0].style.visibility = "visible";
    this.$cache.single[0].style.visibility = "hidden";
}

if (min < this.labels.p_min + 1) {
    this.$cache.min[0].style.visibility = "hidden";
} else {
    this.$cache.min[0].style.visibility = "visible";
}

if (max > 100 - this.labels.p_max - 1) {
    this.$cache.max[0].style.visibility = "hidden";
} else {
    this.$cache.max[0].style.visibility = "visible";
}
```

```
    }
},

drawShadow: function () {
    var o = this.options,
        c = this.$cache,

        is_from_min = typeof o.from_min === "number" && !isNaN(o.from_min),
        is_from_max = typeof o.from_max === "number" && !isNaN(o.from_max),
        is_to_min = typeof o.to_min === "number" && !isNaN(o.to_min),
        is_to_max = typeof o.to_max === "number" && !isNaN(o.to_max),

        from_min,
        from_max,
        to_min,
        to_max;

    if (o.type === "single") {
        if (o.from_shadow && (is_from_min || is_from_max)) {
            from_min = this.convertToPercent(is_from_min ? o.from_min : o.min);
            from_max = this.convertToPercent(is_from_max ? o.from_max : o.max) - from_min;
            from_min = this.toFixed(from_min - (this.coords.p_handle / 100 * from_min));
            from_max = this.toFixed(from_max - (this.coords.p_handle / 100 * from_max));
            from_min = from_min + (this.coords.p_handle / 2);

            c.shad_single[0].style.display = "block";
            c.shad_single[0].style.left = from_min + "%";
            c.shad_single[0].style.width = from_max + "%";
        } else {
            c.shad_single[0].style.display = "none";
        }
    } else {
        if (o.from_shadow && (is_from_min || is_from_max)) {
            from_min = this.convertToPercent(is_from_min ? o.from_min : o.min);
            from_max = this.convertToPercent(is_from_max ? o.from_max : o.max) - from_min;
            from_min = this.toFixed(from_min - (this.coords.p_handle / 100 * from_min));
            from_max = this.toFixed(from_max - (this.coords.p_handle / 100 * from_max));
            from_min = from_min + (this.coords.p_handle / 2);

            c.shad_from[0].style.display = "block";
            c.shad_from[0].style.left = from_min + "%";
            c.shad_from[0].style.width = from_max + "%";
        } else {
            c.shad_from[0].style.display = "none";
        }

        if (o.to_shadow && (is_to_min || is_to_max)) {
            to_min = this.convertToPercent(is_to_min ? o.to_min : o.min);
            to_max = this.convertToPercent(is_to_max ? o.to_max : o.max) - to_min;
            to_min = this.toFixed(to_min - (this.coords.p_handle / 100 * to_min));
            to_max = this.toFixed(to_max - (this.coords.p_handle / 100 * to_max));
            to_min = to_min + (this.coords.p_handle / 2);
```

```
                c.shad_to[0].style.display = "block";
                c.shad_to[0].style.left = to_min + "%";
                c.shad_to[0].style.width = to_max + "%";
            } else {
                c.shad_to[0].style.display = "none";
            }
        }
    },

    writeToInput: function () {
        if (this.options.type === "single") {
            if (this.options.values.length) {
                this.$cache.input.prop("value", this.result.from_value);
            } else {
                this.$cache.input.prop("value", this.result.from);
            }
            this.$cache.input.data("from", this.result.from);
        } else {
            if (this.options.values.length) {
                this.$cache.input.prop("value", this.result.from_value +
this.options.input_values_separator + this.result.to_value);
            } else {
                this.$cache.input.prop("value", this.result.from + this.options.input_values_separator +
this.result.to);
            }
            this.$cache.input.data("from", this.result.from);
            this.$cache.input.data("to", this.result.to);
        }
    },


    callOnStart: function () {
        this.writeToInput();

        if (this.options.onStart && typeof this.options.onStart === "function") {
            if (this.options.scope) {
                this.options.onStart.call(this.options.scope, this.result);
            } else {
                this.options.onStart(this.result);
            }
        }
    },
    callOnChange: function () {
        this.writeToInput();

        if (this.options.onChange && typeof this.options.onChange === "function") {
            if (this.options.scope) {
                this.options.onChange.call(this.options.scope, this.result);
            } else {
                this.options.onChange(this.result);
            }
```

```
    }
},
callOnFinish: function () {
    this.writeToInput();

    if (this.options.onFinish && typeof this.options.onFinish === "function") {
        if (this.options.scope) {
            this.options.onFinish.call(this.options.scope, this.result);
        } else {
            this.options.onFinish(this.result);
        }
    }
},
callOnUpdate: function () {
    this.writeToInput();

    if (this.options.onUpdate && typeof this.options.onUpdate === "function") {
        if (this.options.scope) {
            this.options.onUpdate.call(this.options.scope, this.result);
        } else {
            this.options.onUpdate(this.result);
        }
    }
},


toggleInput: function () {
    this.$cache.input.toggleClass("irs-hidden-input");

    if (this.has_tab_index) {
        this.$cache.input.prop("tabindex", -1);
    } else {
        this.$cache.input.removeProp("tabindex");
    }

    this.has_tab_index = !this.has_tab_index;
},


convertToPercent: function (value, no_min) {
    var diapason = this.options.max - this.options.min,
        one_percent = diapason / 100,
        val, percent;

    if (!diapason) {
        this.no_diapason = true;
        return 0;
    }

    if (no_min) {
        val = value;
    } else {
```

```
        val = value - this.options.min;
    }

    percent = val / one_percent;

    return this.toFixed(percent);
},


convertToValue: function (percent) {
    var min = this.options.min,
        max = this.options.max,
        min_decimals = min.toString().split(".")[1],
        max_decimals = max.toString().split(".")[1],
        min_length, max_length,
        avg_decimals = 0,
        abs = 0;

    if (percent === 0) {
        return this.options.min;
    }
    if (percent === 100) {
        return this.options.max;
    }


    if (min_decimals) {
        min_length = min_decimals.length;
        avg_decimals = min_length;
    }
    if (max_decimals) {
        max_length = max_decimals.length;
        avg_decimals = max_length;
    }
    if (min_length && max_length) {
        avg_decimals = (min_length >= max_length) ? min_length : max_length;
    }

    if (min < 0) {
        abs = Math.abs(min);
        min = +(min + abs).toFixed(avg_decimals);
        max = +(max + abs).toFixed(avg_decimals);
    }

    var number = ((max - min) / 100 * percent) + min,
        string = this.options.step.toString().split(".")[1],
        result;

    if (string) {
        number = +number.toFixed(string.length);
    } else {
        number = number / this.options.step;
```

```
            number = number * this.options.step;

            number = +number.toFixed(0);
        }

        if (abs) {
            number -= abs;
        }

        if (string) {
            result = +number.toFixed(string.length);
        } else {
            result = this.toFixed(number);
        }

        if (result < this.options.min) {
            result = this.options.min;
        } else if (result > this.options.max) {
            result = this.options.max;
        }

        return result;
    },

    calcWithStep: function (percent) {
        var rounded = Math.round(percent / this.coords.p_step) * this.coords.p_step;

        if (rounded > 100) {
            rounded = 100;
        }
        if (percent === 100) {
            rounded = 100;
        }

        return this.toFixed(rounded);
    },

    checkMinInterval: function (p_current, p_next, type) {
        var o = this.options,
            current,
            next;

        if (!o.min_interval) {
            return p_current;
        }

        current = this.convertToValue(p_current);
        next = this.convertToValue(p_next);

        if (type === "from") {

            if (next - current < o.min_interval) {
```

```
            current = next - o.min_interval;
        }

    } else {

        if (current - next < o.min_interval) {
            current = next + o.min_interval;
        }

    }

    return this.convertToPercent(current);
},

checkMaxInterval: function (p_current, p_next, type) {
    var o = this.options,
        current,
        next;

    if (!o.max_interval) {
        return p_current;
    }

    current = this.convertToValue(p_current);
    next = this.convertToValue(p_next);

    if (type === "from") {

        if (next - current > o.max_interval) {
            current = next - o.max_interval;
        }

    } else {

        if (current - next > o.max_interval) {
            current = next + o.max_interval;
        }

    }

    return this.convertToPercent(current);
},

checkDiapason: function (p_num, min, max) {
    var num = this.convertToValue(p_num),
        o = this.options;

    if (typeof min !== "number") {
        min = o.min;
    }

    if (typeof max !== "number") {
```

```
        max = o.max;
    }

    if (num < min) {
        num = min;
    }

    if (num > max) {
        num = max;
    }

    return this.convertToPercent(num);
},

toFixed: function (num) {
    num = num.toFixed(20);
    return +num;
},

_prettify: function (num) {
    if (!this.options.prettify_enabled) {
        return num;
    }

    if (this.options.prettify && typeof this.options.prettify === "function") {
        return this.options.prettify(num);
    } else {
        return this.prettify(num);
    }
},

prettify: function (num) {
    var n = num.toString();
    return n.replace(/(\d{1,3}(?=(?:\d\d\d)+(?!\d)))/g, "$1" + this.options.prettify_separator);
},

checkEdges: function (left, width) {
    if (!this.options.force_edges) {
        return this.toFixed(left);
    }

    if (left < 0) {
        left = 0;
    } else if (left > 100 - width) {
        left = 100 - width;
    }

    return this.toFixed(left);
},

validate: function () {
    var o = this.options,
```

```
    r = this.result,
    v = o.values,
    vl = v.length,
    value,
    i;

if (typeof o.min === "string") o.min = +o.min;
if (typeof o.max === "string") o.max = +o.max;
if (typeof o.from === "string") o.from = +o.from;
if (typeof o.to === "string") o.to = +o.to;
if (typeof o.step === "string") o.step = +o.step;

if (typeof o.from_min === "string") o.from_min = +o.from_min;
if (typeof o.from_max === "string") o.from_max = +o.from_max;
if (typeof o.to_min === "string") o.to_min = +o.to_min;
if (typeof o.to_max === "string") o.to_max = +o.to_max;

if (typeof o.grid_num === "string") o.grid_num = +o.grid_num;

if (o.max < o.min) {
    o.max = o.min;
}

if (vl) {
    o.p_values = [];
    o.min = 0;
    o.max = vl - 1;
    o.step = 1;
    o.grid_num = o.max;
    o.grid_snap = true;

    for (i = 0; i < vl; i++) {
        value = +v[i];

        if (!isNaN(value)) {
            v[i] = value;
            value = this._prettify(value);
        } else {
            value = v[i];
        }

        o.p_values.push(value);
    }
}

if (typeof o.from !== "number" || isNaN(o.from)) {
    o.from = o.min;
}

if (typeof o.to !== "number" || isNaN(o.to)) {
    o.to = o.max;
}
```

```
if (o.type === "single") {

    if (o.from < o.min) o.from = o.min;
    if (o.from > o.max) o.from = o.max;

} else {

    if (o.from < o.min) o.from = o.min;
    if (o.from > o.max) o.from = o.max;

    if (o.to < o.min) o.to = o.min;
    if (o.to > o.max) o.to = o.max;

    if (this.update_check.from) {

        if (this.update_check.from !== o.from) {
            if (o.from > o.to) o.from = o.to;
        }
        if (this.update_check.to !== o.to) {
            if (o.to < o.from) o.to = o.from;
        }

    }

    if (o.from > o.to) o.from = o.to;
    if (o.to < o.from) o.to = o.from;

}

if (typeof o.step !== "number" || isNaN(o.step) || !o.step || o.step < 0) {
    o.step = 1;
}

if (typeof o.from_min === "number" && o.from < o.from_min) {
    o.from = o.from_min;
}

if (typeof o.from_max === "number" && o.from > o.from_max) {
    o.from = o.from_max;
}

if (typeof o.to_min === "number" && o.to < o.to_min) {
    o.to = o.to_min;
}

if (typeof o.to_max === "number" && o.from > o.to_max) {
    o.to = o.to_max;
}

if (r) {
    if (r.min !== o.min) {
```

```
                r.min = o.min;
            }

            if (r.max !== o.max) {
                r.max = o.max;
            }

            if (r.from < r.min || r.from > r.max) {
                r.from = o.from;
            }

            if (r.to < r.min || r.to > r.max) {
                r.to = o.to;
            }
        }

        if (typeof o.min_interval !== "number" || isNaN(o.min_interval) || !o.min_interval ||
o.min_interval < 0) {
            o.min_interval = 0;
        }

        if (typeof o.max_interval !== "number" || isNaN(o.max_interval) || !o.max_interval ||
o.max_interval < 0) {
            o.max_interval = 0;
        }

        if (o.min_interval && o.min_interval > o.max - o.min) {
            o.min_interval = o.max - o.min;
        }

        if (o.max_interval && o.max_interval > o.max - o.min) {
            o.max_interval = o.max - o.min;
        }
    },

    decorate: function (num, original) {
        var decorated = "",
            o = this.options;

        if (o.prefix) {
            decorated += o.prefix;
        }

        decorated += num;

        if (o.max_postfix) {
            if (o.values.length && num === o.p_values[o.max]) {
                decorated += o.max_postfix;
                if (o.postfix) {
                    decorated += " ";
                }
            } else if (original === o.max) {
```

```
            decorated += o.max_postfix;
            if (o.postfix) {
                decorated += " ";
            }
        }
    }

    if (o.postfix) {
        decorated += o.postfix;
    }

    return decorated;
},

updateFrom: function () {
    this.result.from = this.options.from;
    this.result.from_percent = this.convertToPercent(this.result.from);
    this.result.from_pretty = this._prettify(this.result.from);
    if (this.options.values) {
        this.result.from_value = this.options.values[this.result.from];
    }
},

updateTo: function () {
    this.result.to = this.options.to;
    this.result.to_percent = this.convertToPercent(this.result.to);
    this.result.to_pretty = this._prettify(this.result.to);
    if (this.options.values) {
        this.result.to_value = this.options.values[this.result.to];
    }
},

updateResult: function () {
    this.result.min = this.options.min;
    this.result.max = this.options.max;
    this.updateFrom();
    this.updateTo();
},

appendGrid: function () {
    if (!this.options.grid) {
        return;
    }

    var o = this.options,
        i, z,

        total = o.max - o.min,
        big_num = o.grid_num,
        big_p = 0,
        big_w = 0,
```

```
    small_max = 4,
    local_small_max,
    small_p,
    small_w = 0,

    result,
    html = '';



this.calcGridMargin();

if (o.grid_snap) {

    if (total > 50) {
        big_num = 50 / o.step;
        big_p = this.toFixed(o.step / 0.5);
    } else {
        big_num = total / o.step;
        big_p = this.toFixed(o.step / (total / 100));
    }

} else {
    big_p = this.toFixed(100 / big_num);
}

if (big_num > 4) {
    small_max = 3;
}
if (big_num > 7) {
    small_max = 2;
}
if (big_num > 14) {
    small_max = 1;
}
if (big_num > 28) {
    small_max = 0;
}

for (i = 0; i < big_num + 1; i++) {
    local_small_max = small_max;

    big_w = this.toFixed(big_p * i);

    if (big_w > 100) {
        big_w = 100;
    }
    this.coords.big[i] = big_w;

    small_p = (big_w - (big_p * (i - 1))) / (local_small_max + 1);

    for (z = 1; z <= local_small_max; z++) {
```

```
                if (big_w === 0) {
                    break;
                }

                small_w = this.toFixed(big_w - (small_p * z));

                html += '<span class="irs-grid-pol small" style="left: ' + small_w + '%"></span>';
            }

            html += '<span class="irs-grid-pol" style="left: ' + big_w + '%"></span>';

            result = this.convertToValue(big_w);
            if (o.values.length) {
                result = o.p_values[result];
            } else {
                result = this._prettify(result);
            }

            html += '<span class="irs-grid-text js-grid-text-' + i + '" style="left: ' + big_w + '%">' +
result + '</span>';
        }
        this.coords.big_num = Math.ceil(big_num + 1);



        this.$cache.cont.addClass("irs-with-grid");
        this.$cache.grid.html(html);
        this.cacheGridLabels();
    },

    cacheGridLabels: function () {
        var $label, i,
            num = this.coords.big_num;

        for (i = 0; i < num; i++) {
            $label = this.$cache.grid.find(".js-grid-text-" + i);
            this.$cache.grid_labels.push($label);
        }

        this.calcGridLabels();
    },

    calcGridLabels: function () {
        var i, label, start = [], finish = [],
            num = this.coords.big_num;

        for (i = 0; i < num; i++) {
            this.coords.big_w[i] = this.$cache.grid_labels[i].outerWidth(false);
            this.coords.big_p[i] = this.toFixed(this.coords.big_w[i] / this.coords.w_rs * 100);
            this.coords.big_x[i] = this.toFixed(this.coords.big_p[i] / 2);

            start[i] = this.toFixed(this.coords.big[i] - this.coords.big_x[i]);
```

```
            finish[i] = this.toFixed(start[i] + this.coords.big_p[i]);
        }

        if (this.options.force_edges) {
            if (start[0] < -this.coords.grid_gap) {
                start[0] = -this.coords.grid_gap;
                finish[0] = this.toFixed(start[0] + this.coords.big_p[0]);

                this.coords.big_x[0] = this.coords.grid_gap;
            }

            if (finish[num - 1] > 100 + this.coords.grid_gap) {
                finish[num - 1] = 100 + this.coords.grid_gap;
                start[num - 1] = this.toFixed(finish[num - 1] - this.coords.big_p[num - 1]);

                this.coords.big_x[num - 1] = this.toFixed(this.coords.big_p[num - 1] -
this.coords.grid_gap);
            }
        }

        this.calcGridCollision(2, start, finish);
        this.calcGridCollision(4, start, finish);

        for (i = 0; i < num; i++) {
            label = this.$cache.grid_labels[i][0];

            if (this.coords.big_x[i] !== Number.POSITIVE_INFINITY) {
                label.style.marginLeft = -this.coords.big_x[i] + "%";
            }
        }
    },

    // Collisions Calc Beta
    // TODO: Refactor then have plenty of time
    calcGridCollision: function (step, start, finish) {
        var i, next_i, label,
            num = this.coords.big_num;

        for (i = 0; i < num; i += step) {
            next_i = i + (step / 2);
            if (next_i >= num) {
                break;
            }

            label = this.$cache.grid_labels[next_i][0];

            if (finish[i] <= start[next_i]) {
                label.style.visibility = "visible";
            } else {
                label.style.visibility = "hidden";
            }
        }
```

```
},

calcGridMargin: function () {
    if (!this.options.grid_margin) {
        return;
    }

    this.coords.w_rs = this.$cache.rs.outerWidth(false);
    if (!this.coords.w_rs) {
        return;
    }

    if (this.options.type === "single") {
        this.coords.w_handle = this.$cache.s_single.outerWidth(false);
    } else {
        this.coords.w_handle = this.$cache.s_from.outerWidth(false);
    }
    this.coords.p_handle = this.toFixed(this.coords.w_handle  / this.coords.w_rs * 100);
    this.coords.grid_gap = this.toFixed((this.coords.p_handle / 2) - 0.1);

    this.$cache.grid[0].style.width = this.toFixed(100 - this.coords.p_handle) + "%";
    this.$cache.grid[0].style.left = this.coords.grid_gap + "%";
},

update: function (options) {
    if (!this.input) {
        return;
    }

    this.is_update = true;

    this.options.from = this.result.from;
    this.options.to = this.result.to;
    this.update_check.from = this.result.from;
    this.update_check.to = this.result.to;

    this.options = $.extend(this.options, options);
    this.validate();
    this.updateResult(options);

    this.toggleInput();
    this.remove();
    this.init(true);
},

reset: function () {
    if (!this.input) {
        return;
    }

    this.updateResult();
    this.update();
```

```
    },

    destroy: function () {
        if (!this.input) {
            return;
        }

        this.toggleInput();
        this.$cache.input.prop("readonly", false);
        $.data(this.input, "ionRangeSlider", null);

        this.remove();
        this.input = null;
        this.options = null;
    }
};

$.fn.ionRangeSlider = function (options) {
    return this.each(function() {
        if (!$.data(this, "ionRangeSlider")) {
            $.data(this, "ionRangeSlider", new IonRangeSlider(this, options, plugin_count++));
        }
    });
};


// requestAnimationFrame polyfill by Erik Möller. fixes from Paul Irish and Tino Zijdel

(function() {
    var lastTime = 0;
    var vendors = ['ms', 'moz', 'webkit', 'o'];
    for(var x = 0; x < vendors.length && !window.requestAnimationFrame; ++x) {
        window.requestAnimationFrame = window[vendors[x]+'RequestAnimationFrame'];
        window.cancelAnimationFrame = window[vendors[x]+'CancelAnimationFrame']
            || window[vendors[x]+'CancelRequestAnimationFrame'];
    }

    if (!window.requestAnimationFrame)
        window.requestAnimationFrame = function(callback, element) {
            var currTime = new Date().getTime();
            var timeToCall = Math.max(0, 16 - (currTime - lastTime));
            var id = window.setTimeout(function() { callback(currTime + timeToCall); },
              timeToCall);
            lastTime = currTime + timeToCall;
            return id;
        };

    if (!window.cancelAnimationFrame)
        window.cancelAnimationFrame = function(id) {
            clearTimeout(id);
        };
}());
```

```
}));
```

Додаток В. Перевірка на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка моніторингової системи для продажу меду та тестування його якості»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. Войтко В.В.

Оригінальність	**99%**
Схожість	1 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

 Виявлені у роботі запозичення не мають ознак плагіату, їх велика надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора.

 Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Тоха В.В

Керівник роботи _____ Войтко В.В

Додаток Г. Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА МОНІТОРИНГОВОЇ СИСТЕМИ ДЛЯ ПРОДАЖУ МЕДУ ТА ТЕСТУВАННЯ ЙОГО ЯКОСТІ

Рисунок Г.1 – Титульний слайд

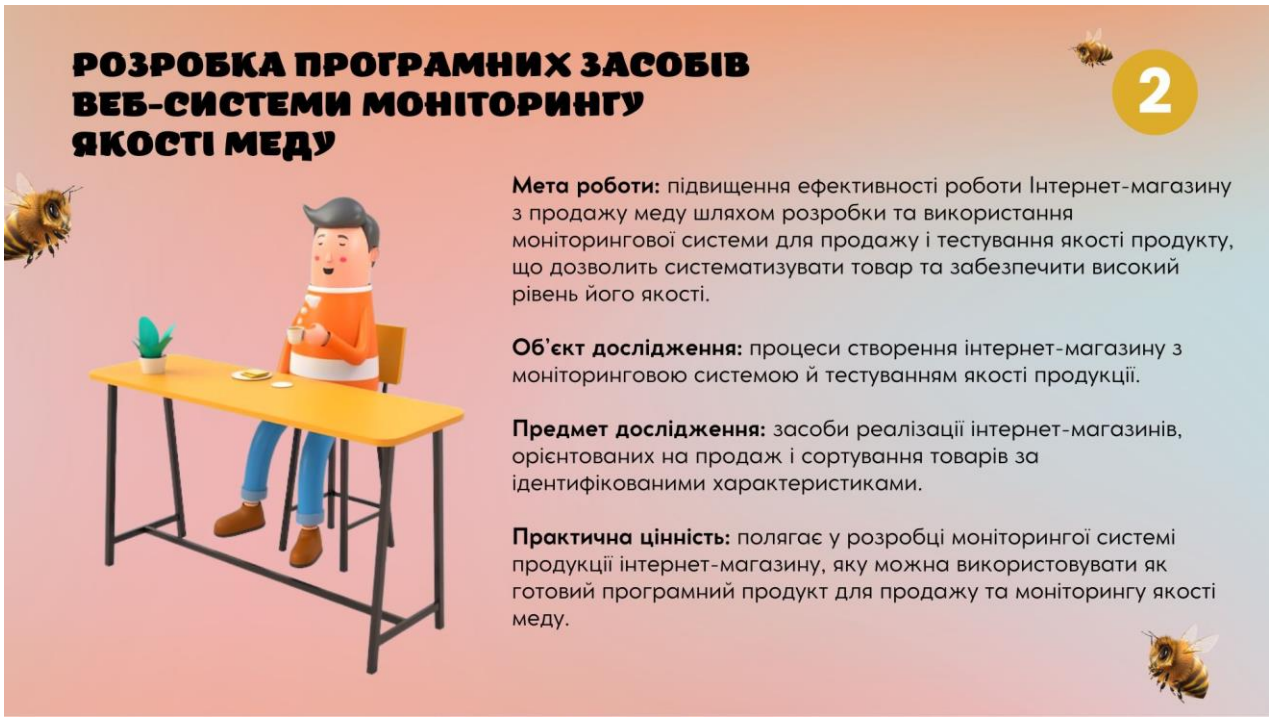

Рисунок Г.2 – Мета та об'єкти дослідження

Рисунок Г.3 – Наукова новизна

Рисунок Г.4 – Завдання бакалаврської дипломної роботи

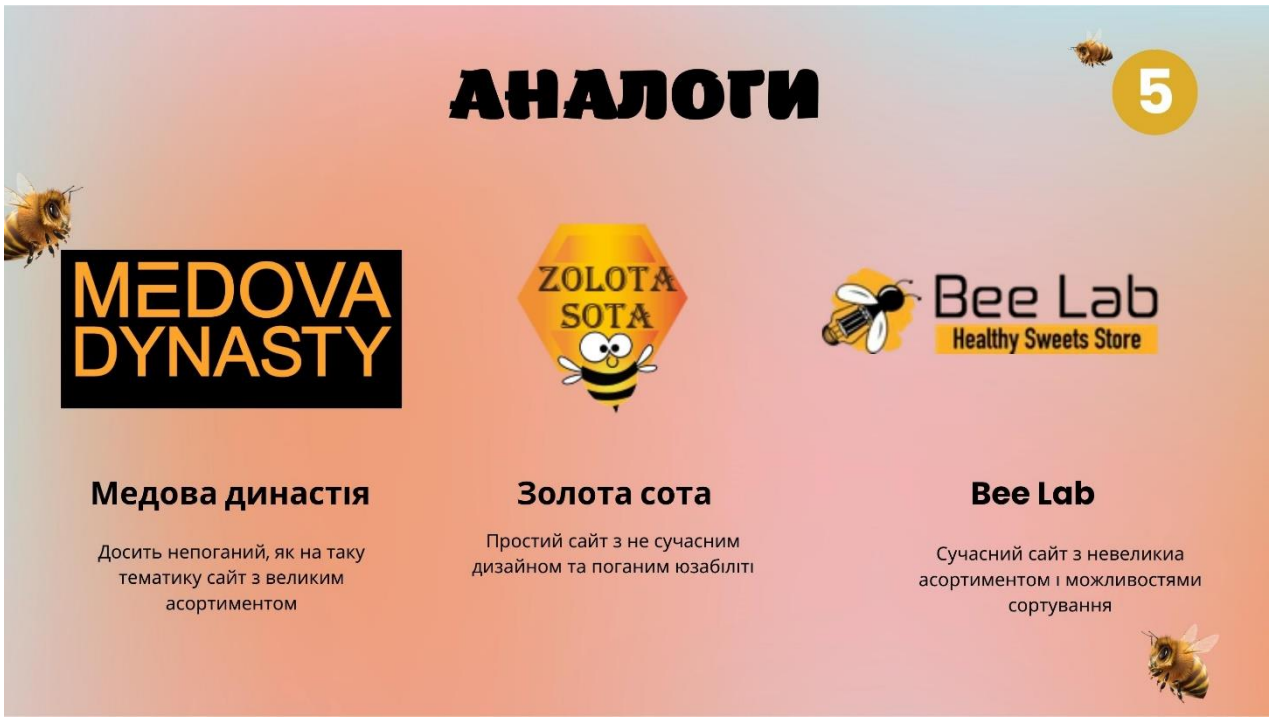

Рисунок Г.5 – Аналоги

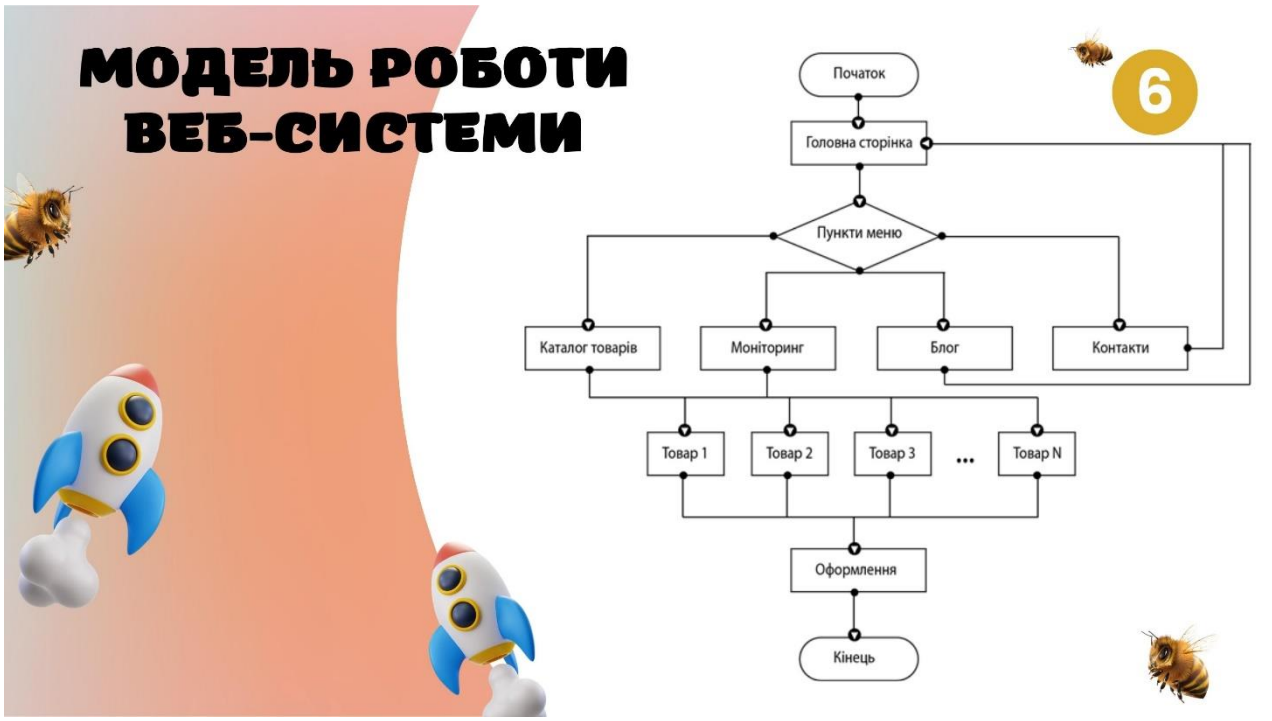

Рисунок Г.6 – Модель роботи веб-системи

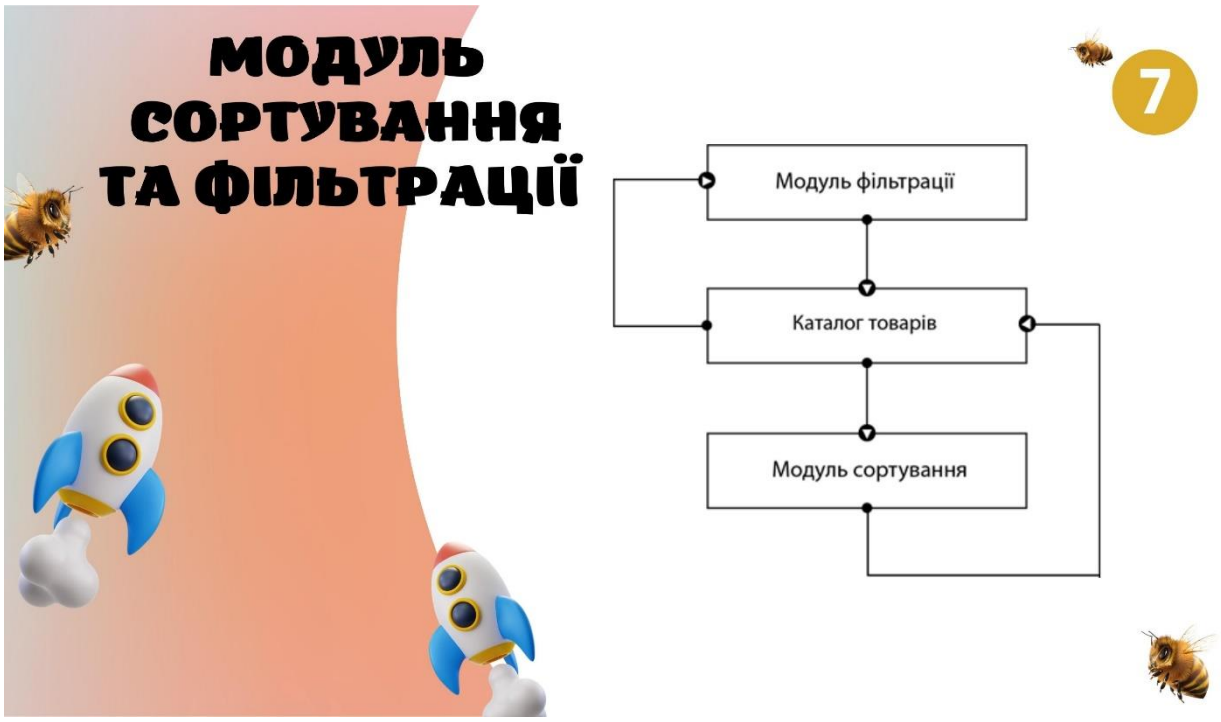

Рисунок Г.7 – Модуль сортування та фільтрації

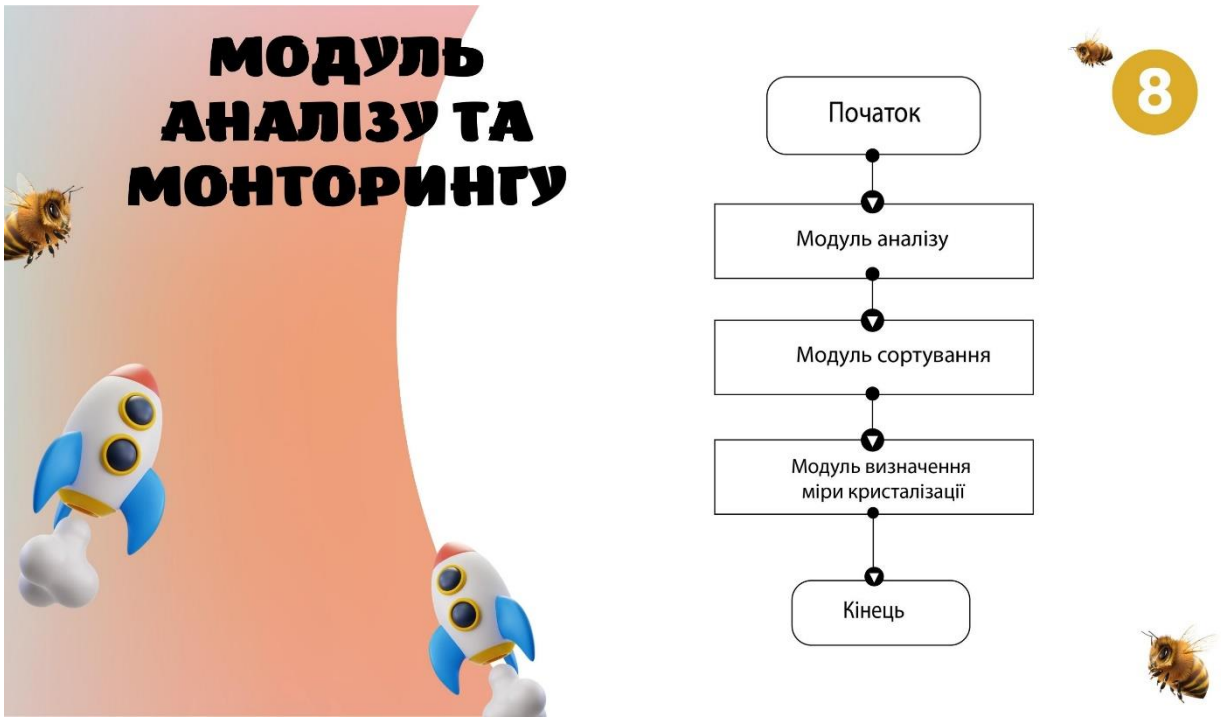

Рисунок Г.8 – Модуль аналізу та моніторингу

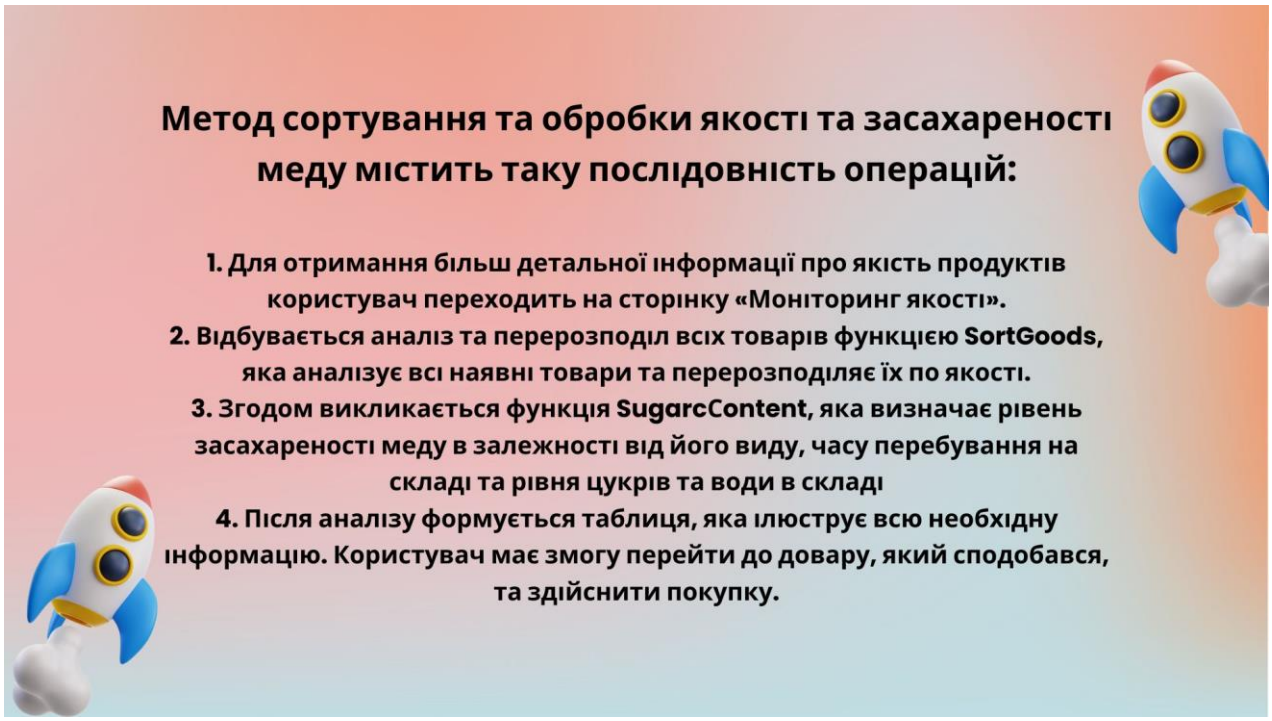

Рисунок Г.9 – Послідовність методу методу засахареності

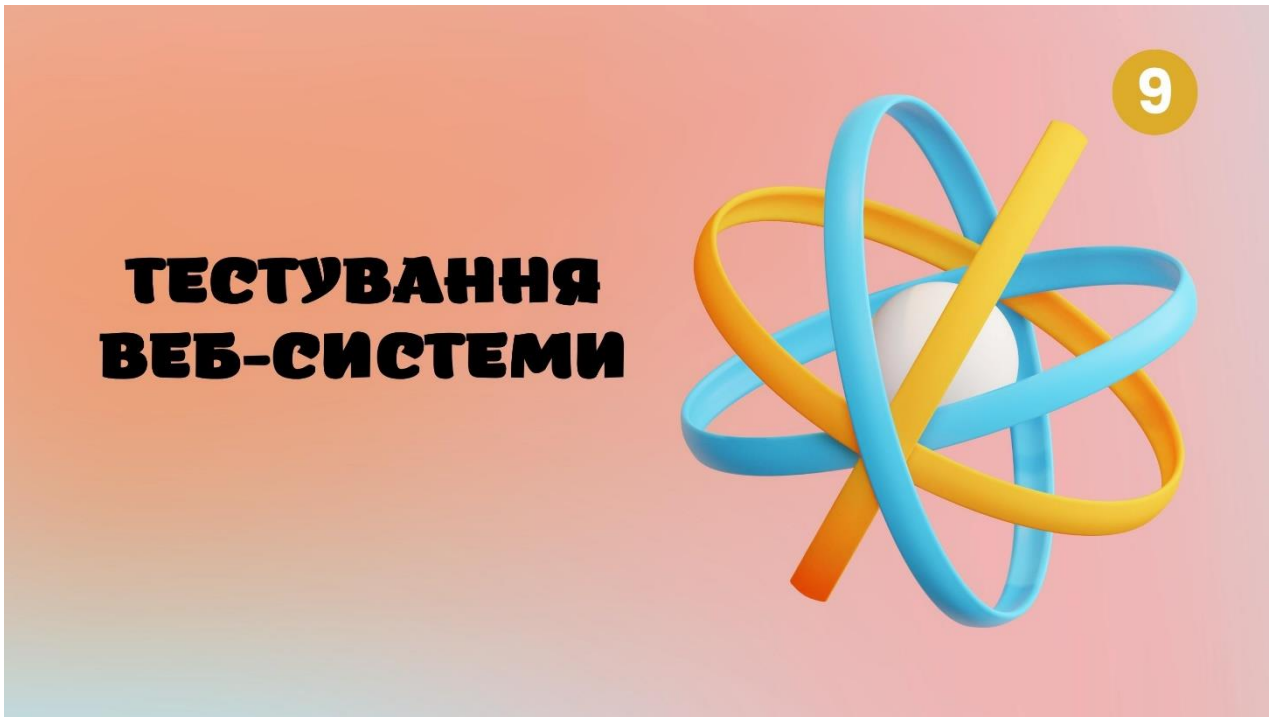

Рисунок Г.10 – Тестування

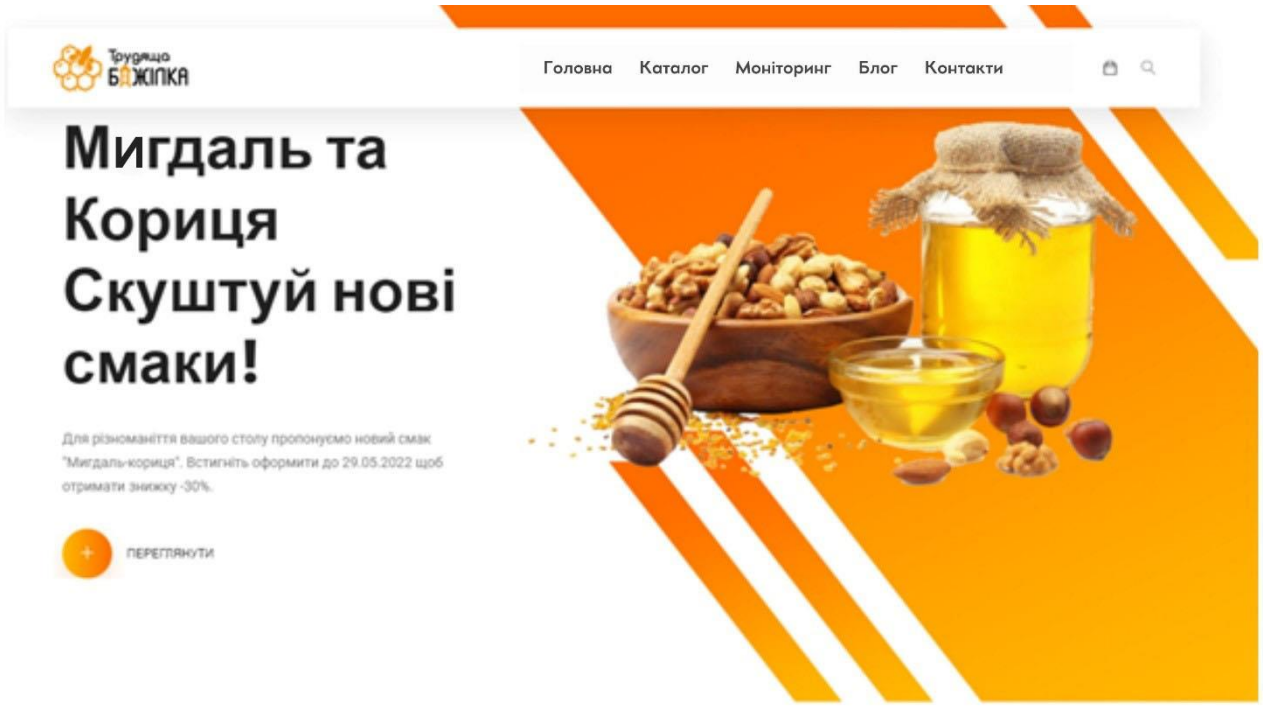

Рисунок Г.11 – Запуск та робота слайдера

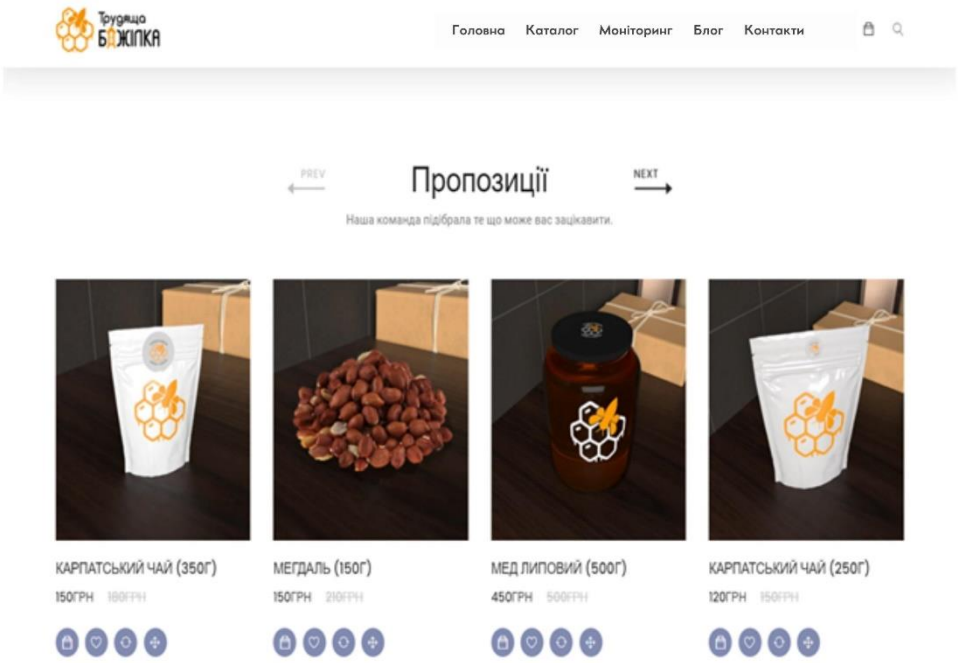

Рисунок Г.12 – Робота слайдера з блоками товарів

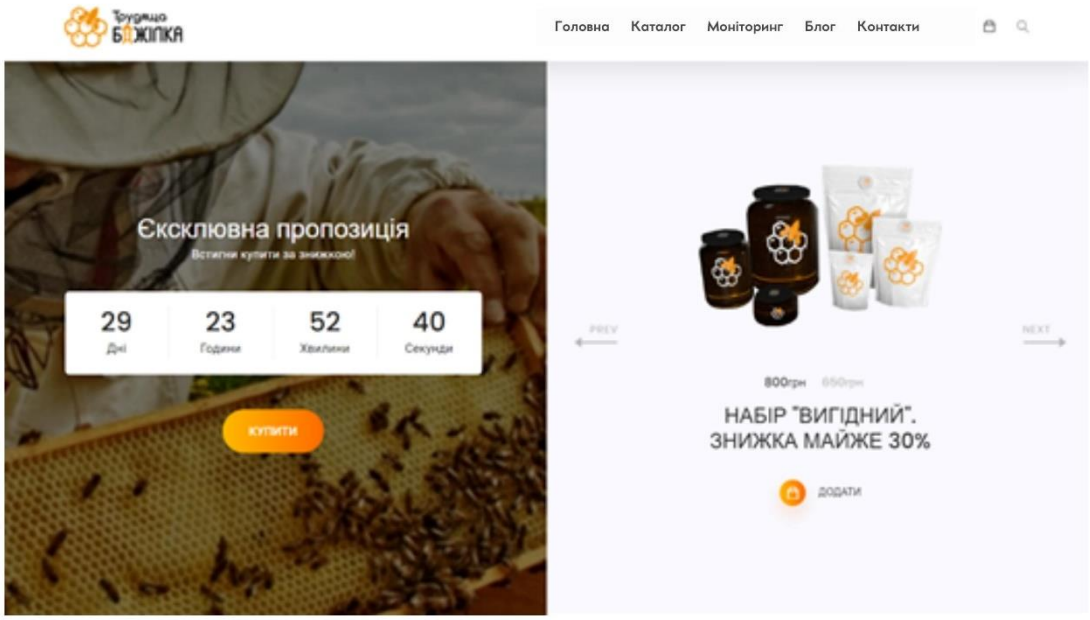

Рисунок Г.13 – Робота таймера та слайдера з наборами

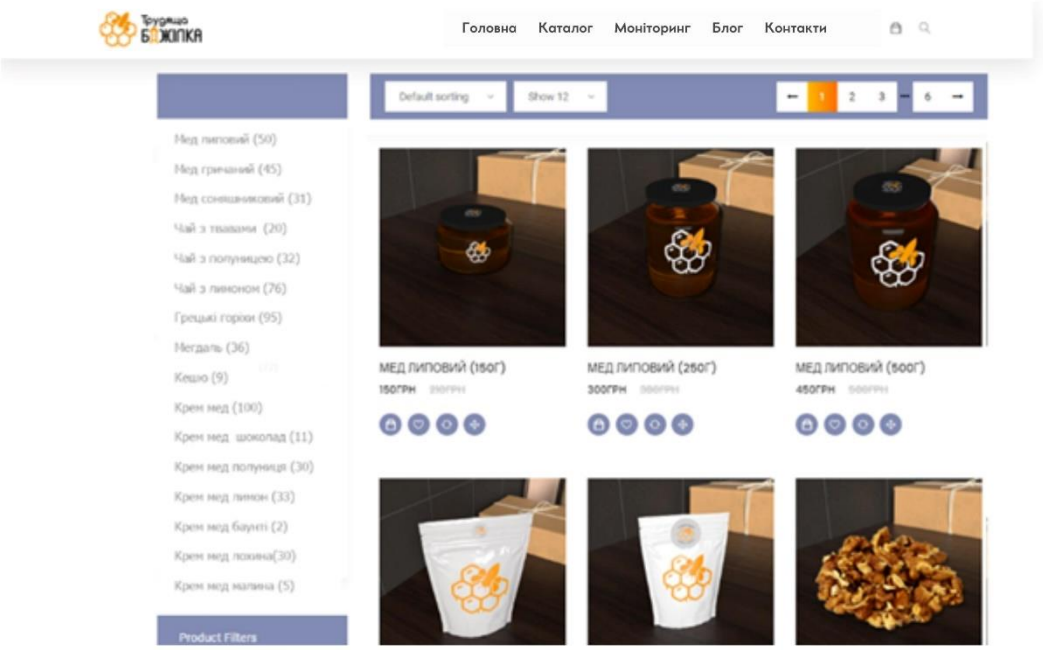

Рисунок Г.14 – Каталог товарів

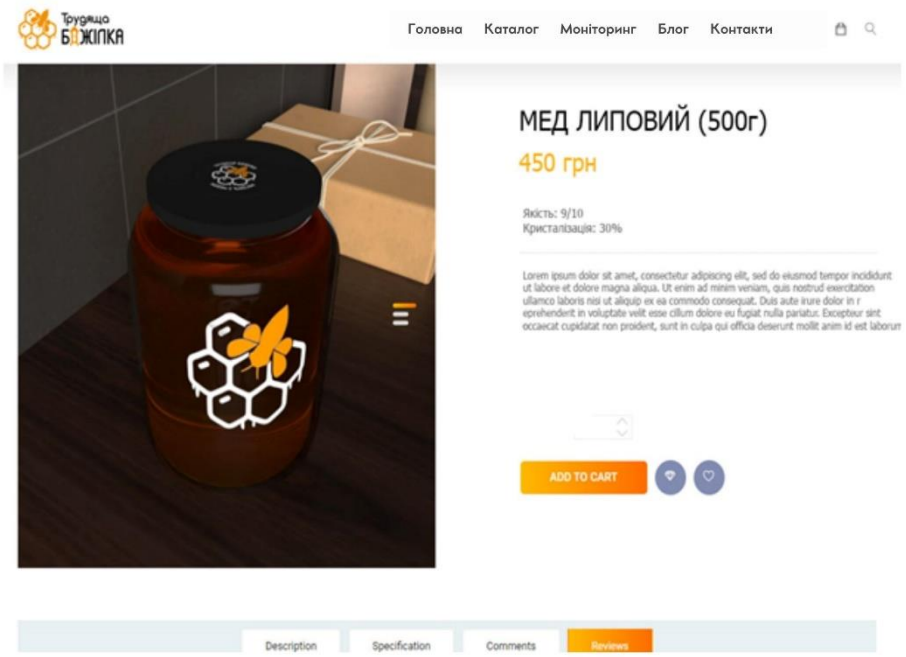

Рисунок Г.15 – Карточка товару

Рисунок Г.16 – Оформлення замовлення

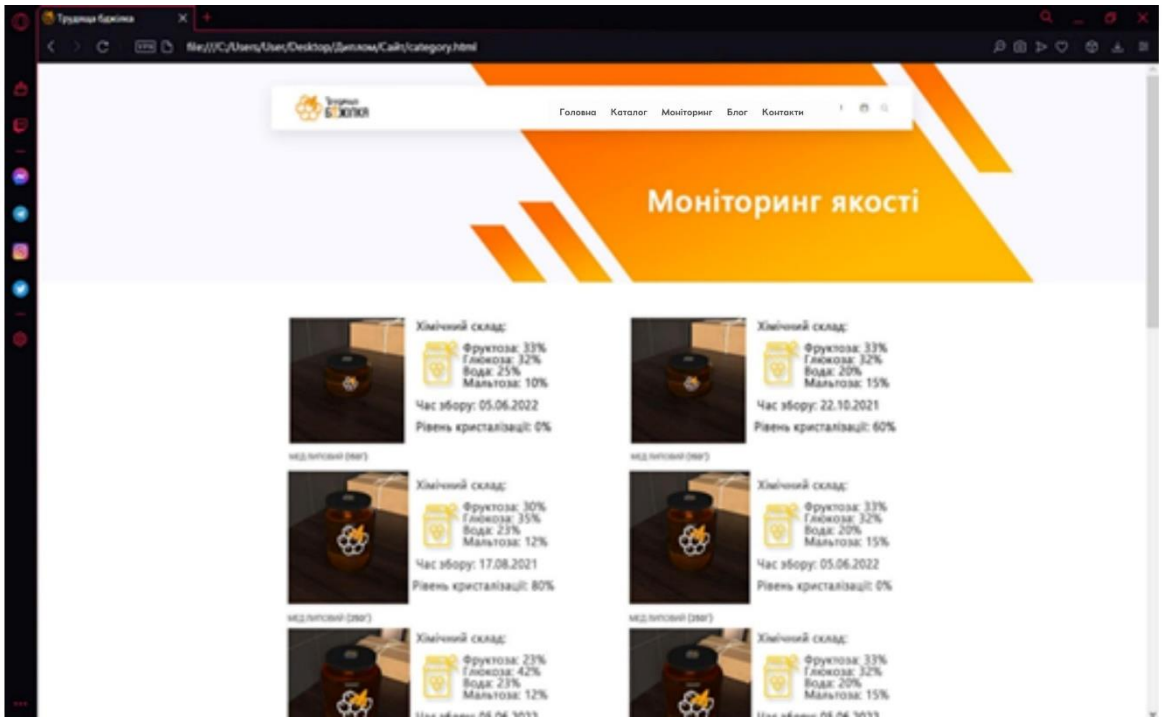

Рисунок Г.17 – Моніторинг якості

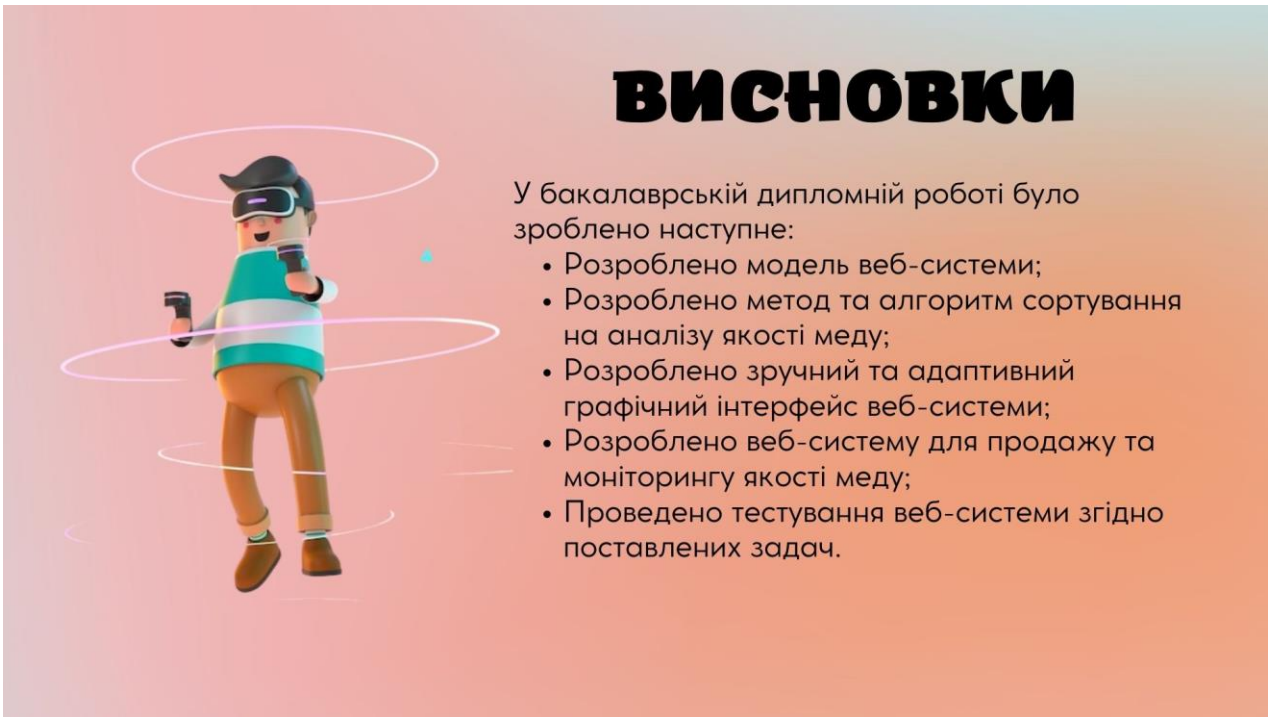

Рисунок Г.18 – Висновки

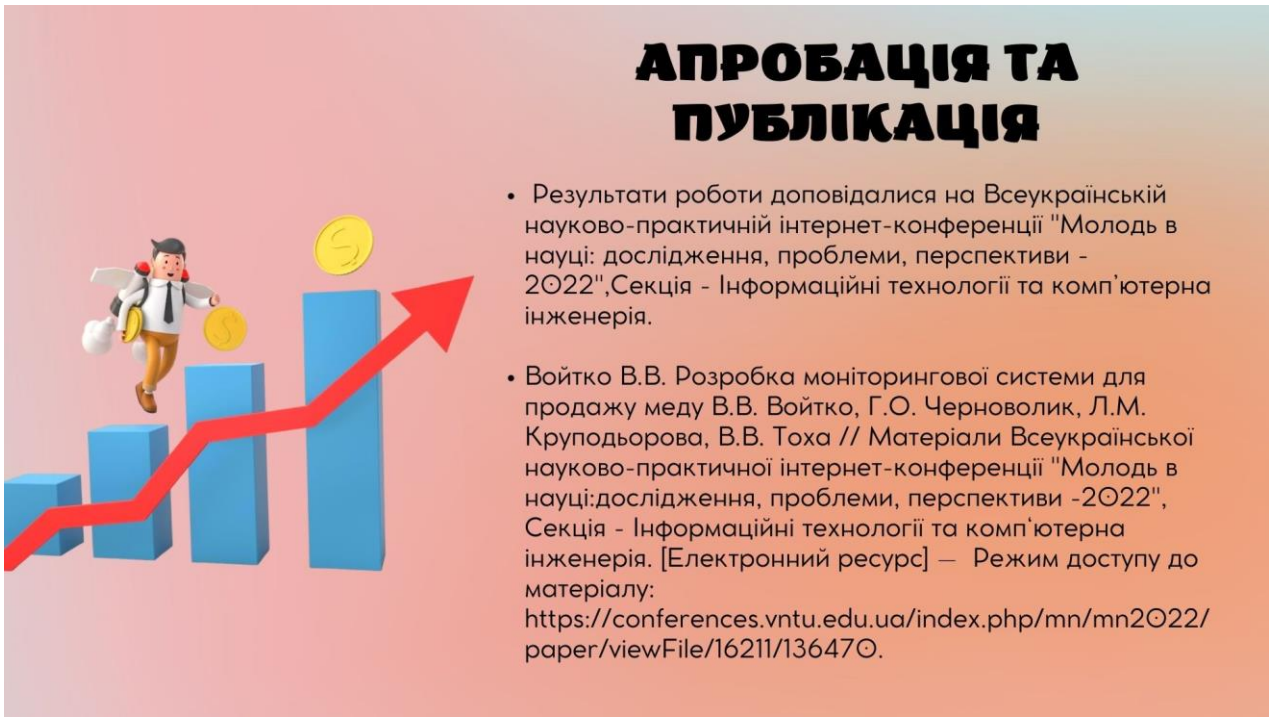

Рисунок Г.19 – Апробація та публікація