МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ ___Андрій ФОРСЮК__	_____ _Сергій ГРИБКОВ__
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«05» __червня_____ 2023р.	«05» ____червня_____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми __Комп'ютерні науки

на тему: Розроблення Web-додатку замовлення продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод»

Виконав: здобувач _4_ курсу, групи __КН-4-3____

_____Гриценко Іван Олексійович_____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник ____Харкянен Олена Валеріївна_____ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2023р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
Сергій ГРИБКОВ

" 04 " квітня 2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
Гриценка Івана Олексійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення Web-додатку для замовлення продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод»
керівник роботи Харкянен Олена Валеріївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від 03 квітня 2023 року № 204-кс
2. Строк подання здобувачем роботи 01.06.2023 р.
3. Вихідні дані до роботи
Інформація про ТМ «Яготинське для дітей» ТДВ «Яготинський Маслозавод», дані про поточний стан підприємства
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) системний аналіз діяльності підприємства, виявлення недоліків та визначення задач автоматизації, розробка технічного завдання, алгоритмізація та реалізація комплексу задач автоматизації, розробка інструкції користувача, висновки
5. Перелік графічного матеріалу
Організаційна структура підприємства, функціональні моделі, скріншоти програм-аналогів, скріншоти фрагментів коду, скріншоти розробленого веб-додатку

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доцент к.т.н. Харкянен О.В.		
2	доцент к.т.н. Харкянен О.В.		
3	доцент к.т.н. Харкянен О.В.		
4	доцент к.т.н. Харкянен О.В.		

7. Дата видачі завдання_____04 квітня 2023 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз діяльності підприємства	04.04.2023-22.04.2023	Виконано
2	Розробка технічного завдання	23.04.2023-31.04.2023	Виконано
3	Розробка веб-додатку	01.05.2023-22.05.2023	Виконано
4	Створення інструкції для користувачів	23.05.2023-24.05.2023	Виконано
5	Оформлення пояснювальної записки	25.05.2023-27.05.2023	Виконано
6	Створення презентації	28.05.2023-29.05.2023	Виконано

Здобувач _____ _____Гриценко І.О._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Харкянен О.В._____

АНОТАЦІЯ

Дана кваліфікаційна робота описує причини та процес створення web-додатку для замовлення продукції клієнтами у ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод».

Об’єктом дослідження є діяльність торгового відділу ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод».

Головною метою роботи була розробка Web-додатку для підприємства з використанням набутих за період навчання знань.

У кваліфікаційній роботі було здійснено аналіз об'єкта дослідження, вивчена діяльність торгового відділу підприємства та обґрунтована доцільність використання в його діяльності Web-додатку для замовлення продукції.

Функції, реалізовані у додатку забезпечують можливість замовникам самостійно формувати замовлення на продукцію підприємства, а працівники зможуть опрацьовувати інформацію про замовлену продукцію та організовувати її продаж замовникам.

У технічному завданні визначені вимоги щодо розробки веб-додатку. Робота містить опис процесу створення веб-додатку, а також надає інструкцію по користування додатком для замовників та працівників відділу підприємства.

Кваліфікаційна робота складається з 65 сторінок, 8 таблиць, 35 рисунків, 3 додатків та 27 літературних джерел.

КЛЮЧОВІ СЛОВА: WEB-ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, ЗАМОВЛЕННЯ, ЯГОТИНСЬКЕ ДЛЯ ДІТЕЙ, МАСЛОЗАВОД.

ABSTRACT

This work describes the reasons and the process of creating a web application for ordering products by customers in TM "Yagotynsk for children" TDV "Yagotynsk Maslozavod".

The object of the study is the activity of the trade department of TM "Yagotynsk for children" TDV "Yagotynskiy Maslozavod".

The main goal of the work was the development of a Web application for the enterprise using the knowledge acquired during the training period.

In the qualification work, an analysis of the research object was carried out, the activity of the trade department of the enterprise was studied and the feasibility of using a Web application for ordering products in its activity was substantiated.

The functions implemented in the application provide an opportunity for customers to independently form orders for the company's products, and employees will be able to process information about the ordered products and organize their sale to customers.

The terms of reference define the requirements for the development of a web application. The work contains a description of the process of creating a web application, and also provides instructions for using the application for customers and employees of the enterprise department.

The qualification work consists of 65 pages, 8 tables, 35 figures, 3 appendices and 27 literary sources.

KEY WORDS: WEB-APPLICATION, INFORMATION SYSTEM, DATABASE, ORDER, YAGOTYNSKY FOR CHILDREN, OIL FACTORY

ЗМІСТ

ВСТУП

В сучасному світі інформаційні технології відіграють все більш вагому роль у бізнес-середовищі. Швидкий розвиток технологій та зростання конкуренції на ринку ставлять перед підприємствами потребу вдосконалення продуктів та послуг, а також ефективного залучення та збереження клієнтів.

З урахуванням цих факторів, актуальним стає розробка веб-додатків та використання автоматизованих інформаційних систем для поліпшення роботи підприємств. Вони дозволяють забезпечити автоматизацію рутинних процесів, збільшити ефективність виробництва, покращити якість продуктів та послуг.

У даному контексті, створення веб-додатку для ТМ "Яготинське для дітей" ТДВ "Яготинський маслозавод" є важливим завданням. Цей веб-додаток має на меті створити зручний та ефективний інструмент для замовлення продукції компанії. Він надасть споживачам можливість переглядати асортимент продуктів, ознайомлюватись з їх характеристиками та цінами, а також робити замовлення в зручний спосіб.

Основною метою створення веб-додатку для ТМ "Яготинське для дітей" є поліпшення обслуговування клієнтів, забезпечення швидкості та зручності процесу замовлення продукції. Крім того, веб-додаток дозволить компанії збирати та аналізувати дані про вподобання споживачів, що сприятиме розумінню їхніх потреб та вимог.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ТМ «Яготинське для дітей» ТДВ «ЯГОТИНСЬКИЙ МАСЛОЗАВОД»

1.1. Загальна характеристика ТДВ «Яготинський маслозавод»

ТДВ "Яготинський маслозавод" є одним з провідних виробників молочної продукції в Україні, маслозавод" був заснований у 1929 році в місті Яготин, що знаходиться в Київській області України. З тих пір завод розширив свою діяльність та став одним з провідних виробників молочної продукції в Україні[1].

Протягом багатьох років ТДВ "Яготинський маслозавод" розвивався та впроваджував нові технології виробництва, що дозволило підвищити якість та асортимент продукції. Сьогодні завод є сучасним підприємством зі значним виробничим потенціалом, яке виробляє широкий асортимент молочної продукції, включаючи йогурти, кефіри, масло та інші продукти.

Однією з найвідоміших ліній продуктів ТДВ "Яготинський маслозавод" є бренд "Яготинське". Цей бренд включає в себе різноманітні молочні продукти, такі як йогурти, кефіри, масло та інші продукти. Бренд "Яготинське" є доволі відомим в Україні та доступний в багатьох магазинах та супермаркетах по всій країні.

У 2008 році ТДВ "Яготинський маслозавод" запустив на ринок України новий бренд "Яготинське для дітей", який спеціалізується на виробництві молочних продуктів для дітей різного віку. Бренд "Яготинське для дітей" включає в себе дитячі молочні суміші та йогурти, які містять необхідну кількість живильних речовин для правильного розвитку та здоров'я дітей.

Підприємство має значний досвід у виробництві молочної продукції та використовує сучасні технології для виробництва якісних продуктів. Завод прагне до сталого розвитку та підвищення якості своєї продукції.

На заводі встановлено сучасне обладнання закритого циклу з країн, таких як Швеція, Італія, Німеччина, Ізраїль, Болгарія, Тайвань та інші, яке гарантує відсутність контакту людських рук з продукцією під час її виробництва. За три роки існування ТМ "Яготинське для дітей" зміцнило свої лідерські позиції у категорії дитячого молочного харчування, завойовуючи 40% ринку в Україні.

1.2. Організаційна структура ТОВ «Яготинський маслозавод», роль і взаємодія підрозділів

ТДВ «Яготинський маслозавод» має велику та складну організаційну структуру, яка наведене на рисунок 1.1 [2].

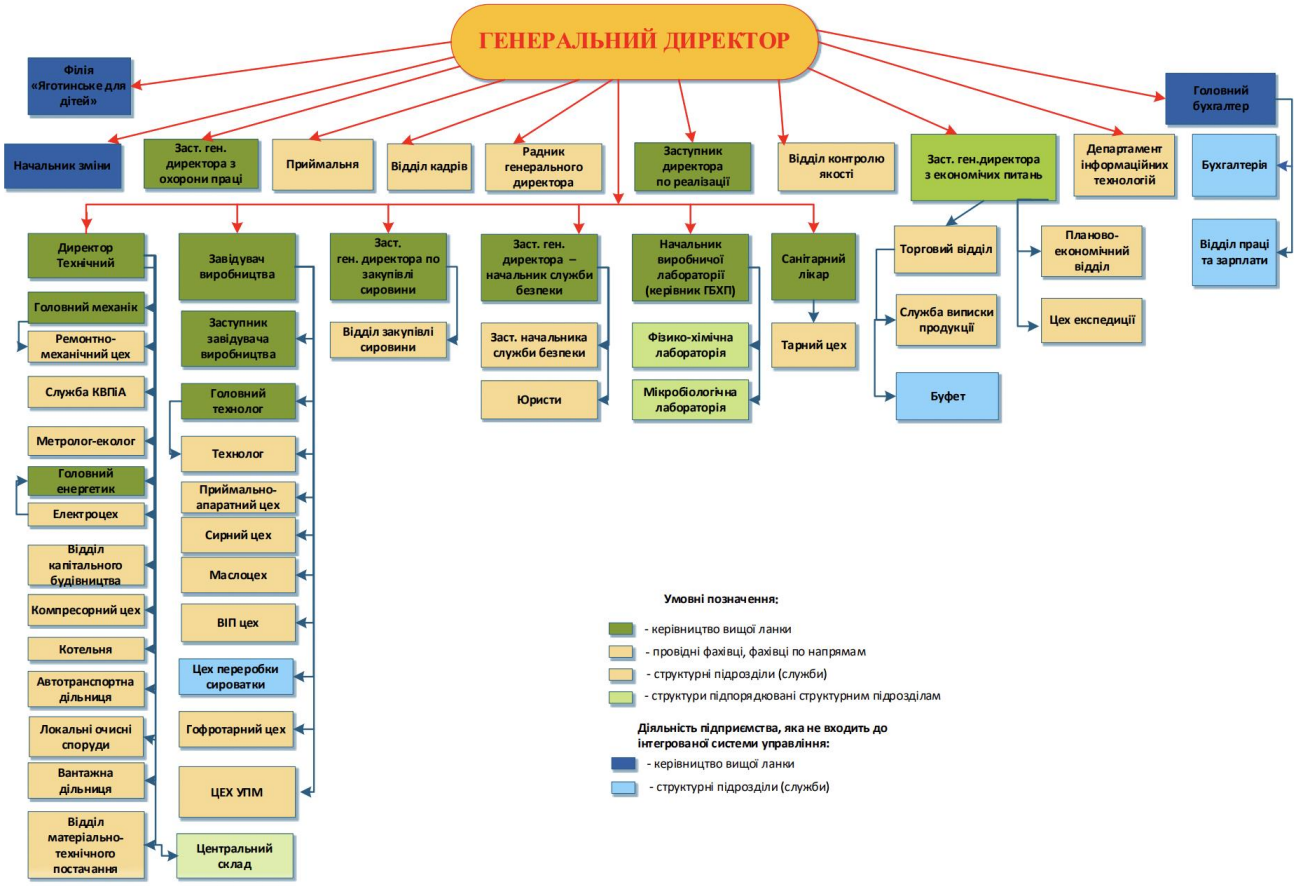


Рис.1.1. Організаційна структура ТОВ «Пирятинський сирзавод»

Основні елементи організаційної структури підприємства ТДВ «Яготинський маслозавод»:

- Генеральний директор: Вищий керівник підприємства, відповідає за загальне управління та прийняття стратегічних рішень.
- Заступник генерального директора з охорони праці: Відповідає за забезпечення безпеки та охорони праці на підприємстві, розробку та виконання відповідних політик та процедур.
- Відділ кадрів: Займається управлінням персоналом, включаючи найм, навчання, розвиток, утримання кадрів та управління персональними справами працівників.

- Радник генерального директора: Надає консультаційну підтримку генеральному директору з питань стратегії, управління та прийняття рішень.
- Відділ контролю якості: Відповідає за контроль та забезпечення високої якості продукції, включаючи проведення іспитів, аналіз та вдосконалення процесів виробництва.
- Заступник генерального директора з економічних питань: Відповідає за фінансове планування, бюджетування, фінансовий аналіз та управління економічними аспектами підприємства.
- Департамент інформаційних технологій: Займається розробкою, впровадженням та підтримкою інформаційних систем і технологій на підприємстві.
- Торговий відділ: Відповідає за реалізацію продукції, укладання контрактів, взаємодію з клієнтами та забезпечення розвитку ринків збуту.
- Служба виписки продукції: Здійснює процес виписки та оформлення продукції для подальшого відправлення клієнтам.
- Головний бухгалтер: Відповідає за фінансовий облік, звітність та податкові питання підприємства.
- Головний механік: Відповідає за технічний стан та ефективну роботу механічного обладнання на підприємстві.
- Головний електрик: Відповідає за технічний стан та ефективну роботу електричного обладнання на підприємстві.
- Головний технолог: Відповідає за розробку, впровадження та вдосконалення технологічних процесів виробництва для забезпечення високої якості продукції.

Ці підрозділи взаємодіють між собою для ефективного функціонування підприємства "Яготинський маслозавод". Кожен з них має свої специфічні обов'язки та відповідальності, спрямовані на досягнення спільних цілей.

Торговий відділ є одним з ключових підрозділів ТДВ "Яготинський маслозавод" і відіграє важливу роль у розвитку та успішному функціонуванні

підприємства. Його головна мета полягає в організації та забезпеченні ефективного збуту продукції, задоволенні потреб клієнтів та розвитку ринків збуту. Торговий відділ відіграє критичну роль у забезпеченні стійкого росту та прибутковості підприємства.

Функції торгового відділу:

- Реалізація продукції: Торговий відділ відповідає за здійснення процесу продажу продукції підприємства. Він встановлює контакти з потенційними клієнтами, проводить переговори, укладає угоди та контракти на постачання продукції. Відділ також відповідає за визначення оптимальних цін на продукцію та умови поставок.
- Розвиток ринків збуту: Торговий відділ активно вивчає ринки збуту, аналізує конкурентну ситуацію та споживчі потреби. Він визначає потенційні сегменти ринку, розробляє стратегії проникнення на нові ринки та розширення присутності на існуючих ринках. Таким чином, відділ сприяє розвитку та зміцненню позицій підприємства на ринку.
- Клієнтське обслуговування: Торговий відділ взаємодіє з клієнтами, задовольняє їх потреби та вимоги. Він надає консультації щодо продукції, допомагає у вирішенні технічних та комерційних питань, забезпечує своєчасну доставку продукції та реагує на запити та скарги клієнтів. Відділ прагне побудувати довгострокові та взаємовигідні відносини з клієнтами.
- Ведення документації: Торговий відділ відповідає за оформлення та зберігання необхідної документації, пов'язаної з угодами, контрактами, замовленнями та іншими документами, що стосуються процесу збуту продукції.

Також торговий відділ компанії впливає на інші підрозділи, взаємодія між відділами наведена в таблиці 1.1

Таблиця 1.1. Взаємодія підрозділів

№	Відділ	Одержання	Надання
1	Відділ кадрів	Інформація про нових працівників та їх умови праці	Інформація про вакансії та вимоги до кандидатів
2	Бухгалтерський	Фінансова звітність	Інформація про реалізовану продукцію та отримані кошти

№	Відділ	Одержання	Надання
3	Відділ контролю якості	Інформація про якість продукції, рекламації	Вимоги до якості продукції, результати контролю якості
4	Технологічний відділ	Вимоги до технологічних процесів та специфікації продукції	Технологічні вимоги до виробництва, оновлення технологій
5	Департамент інформаційних технологій	Запити на розробку та підтримку інформаційних систем	Постачання інформації про замовлення, стан запасів, статистичні дані

1.3. Аналіз нинішнього стану комп'ютеризації підприємства

Для зберігання інформації про зроблені замовлення на підприємстві, переважно, на сьогоднішній день, використовують Microsoft Excel.

Microsoft Excel використовується для збереження даних та документів у вигляді таблиць, у яких міститься інформація про замовлення, клієнтів. Завдяки формулам, які присутні у середовищі Microsoft Excel підприємство може отримувати різні звіти в залежності від отриманих даних.

На сьогоднішній день для замовлення продукції використовуються різні методи – від виклику на телефон, та замовлення необхідної партії продукції шляхом спілкування до листів на електронну пошту, що значно ускладнює роботу з замовниками та сповільнює роботу усього відділу та підприємства в цілому.

Відсутність інформаційної системи, яка зможе об'єднати в собі усі необхідні функції для відділу призводить до зменшення ефективності усього відділу.

1.4. Розроблення функціональної моделі та визначення задач автоматизації

1.4.1. Функціональна модель ТДВ «Яготинський маслозавод»

У середовищі AllFusion ERWin Process Modeler, що є потужним CASE-засобом, була створена функціональна модель для проведення більш докладного аналізу, виявлення переваг та недоліків існуючих бізнес-процесів і визначення завдань автоматизації [3].

На рисунку 1.2 представлена функціональна модель реалізації продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод», а на рисунку 1.3 зображено перший рівень декомпозиції моделі.

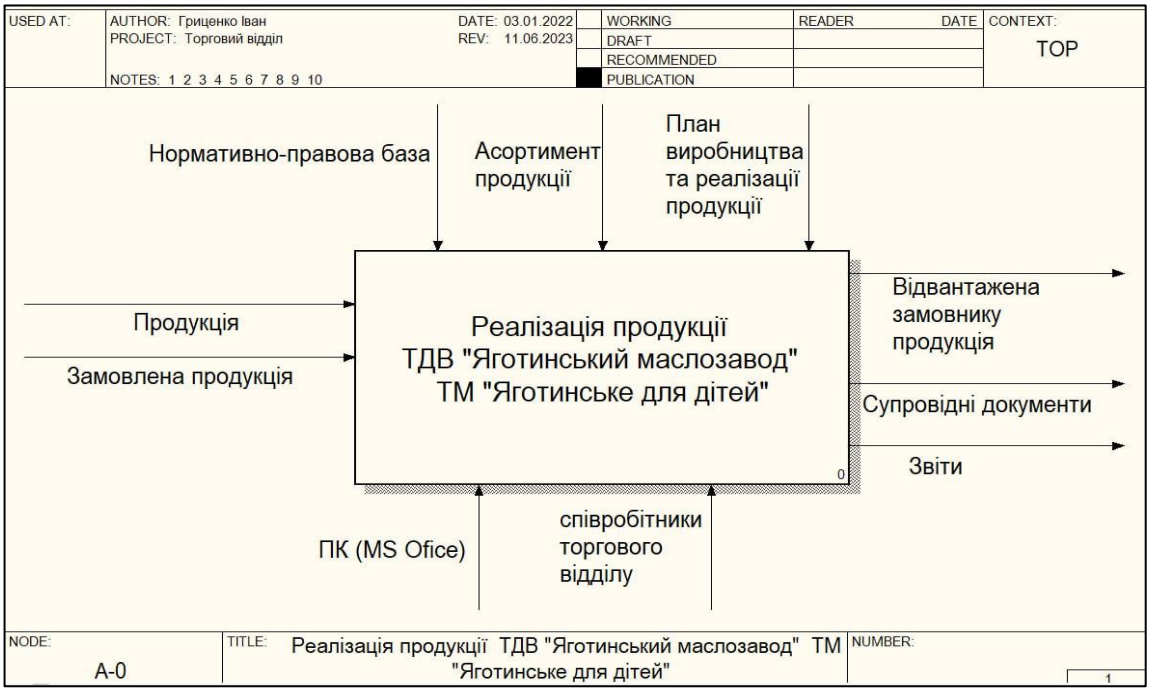


Рис. 1.2 Контекстна діаграма функціональної моделі торгового відділу.

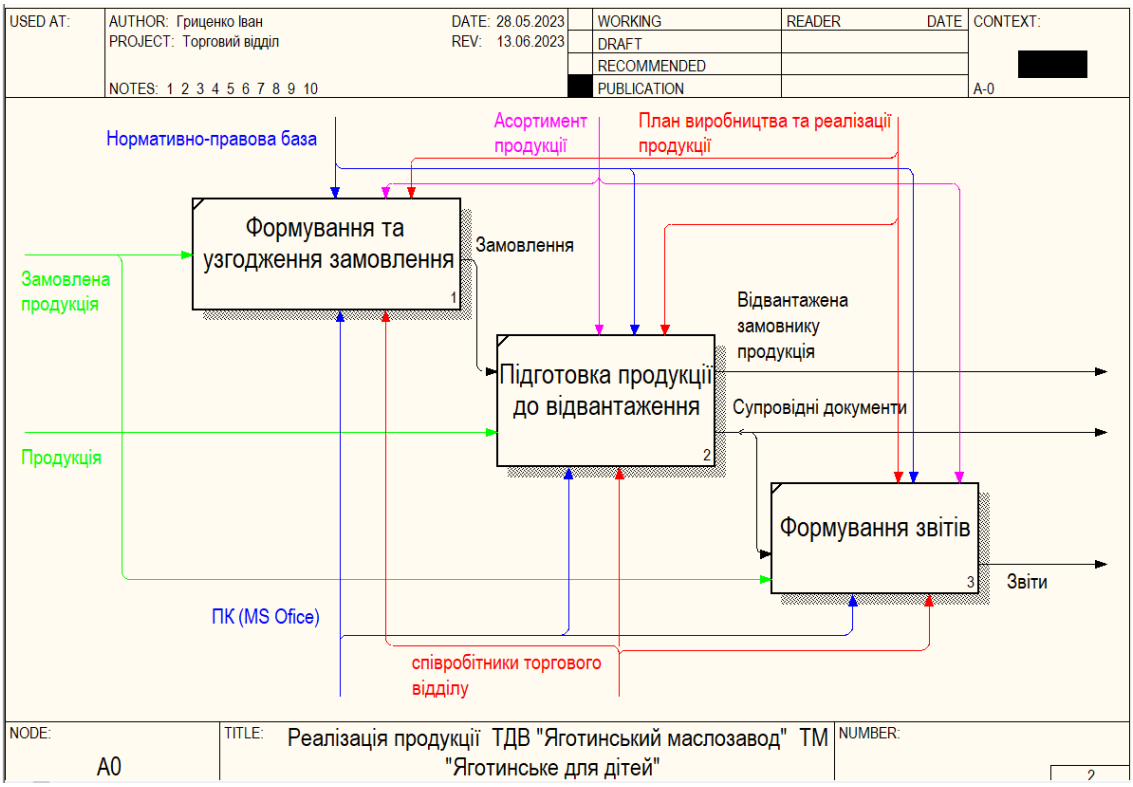


Рис. 1.3 Декомпозиції першого рівня функціональної моделі.

Наведена функціональна модель описує бізнес-процеси, які відбуваються при замовленні продукції клієнтом підприємства.

1.4.2. Виявлені проблеми

Із-за відсутності єдиної та зручної системи для замовлення продукції клієнтами підприємство вимушено витрачати більше часу та ресурсів на обробку замовлень. Для цієї здачі відділ використовує застарілі та неефективні методи, що в повній мірі відображається на ефективності та продуктивності усього відділу та підприємства.

Для вирішення цієї проблеми необхідно звернутись до сучасних та створити зручну систему для замовлення товару та збереження і відображення замовлень, статистики і існуючих товарів, що повинно покращити роботу відділу.

1.4.3. Задачі автоматизації

Первинною задачею стало створення зручного Web-додатку для замовлення продукції клієнтами та відображення усіх даних для працівників.

Виходячи з цього ми можемо виділити такі задачі для автоматизації:

- Створення замовлення клієнтом;
- Перегляд працівником замовлень клієнта;
- Можливість додавати нові продукти до системи;
- Можливість редагувати чи видаляти продукти працівником;
- Можливість переглядати статистику по замовленій продукції.

1.5. Огляд існуючих рішень

Існує декілька продуктів, для збереження різної інформації та взаємодії з нею, але програм для замовлення партій їстівної продукції у відкритому доступі мною не було знайдено, тому у цьому розділі ми будемо розглядати програмні засоби для взаємодії з інформацією, а саме таких програм налічується декілька: Microsoft Access, Microsoft Excel та Google Sheets (таблиці). Нижче наведено більш детальну інформацію про данні продукти:

Microsoft Access - це простий у використанні інструмент для створення бізнес-додатків, використовуючи шаблони або створюючи їх з нуля. Завдяки своїм багатим та інтуїтивно зрозумілим інструментам дизайну, Access може допомогти вам створити привабливі та високофункціональні додатки за мінімальний час[4].

Торговий відділ може використовувати Microsoft Access у своїй роботі для поліпшення ефективності та автоматизації різних процесів. Ось декілька способів, якими Access може бути корисним для торгового відділу: управління базою даних клієнтів - торговий відділ може створити базу даних клієнтів у Microsoft Access. Ця база даних може містити інформацію про клієнтів, контактні дані, історію замовлень та взаємодії. Вона дозволить зручно відстежувати зв'язок з клієнтами, планувати зустрічі та переговори, а також аналізувати дані для виявлення нових можливостей, керування продуктами та запасами - використання Access дозволяє створити базу даних з описом продукції, її характеристиками, цінами та станом запасів. Торговий відділ може відстежувати рух товарів, керувати запасами, отримувати сповіщення про необхідність поповнення запасів та генерувати звіти про продажі та виробництво. На рисунку 1.4 зображено інтерфейс програми Microsoft Access.

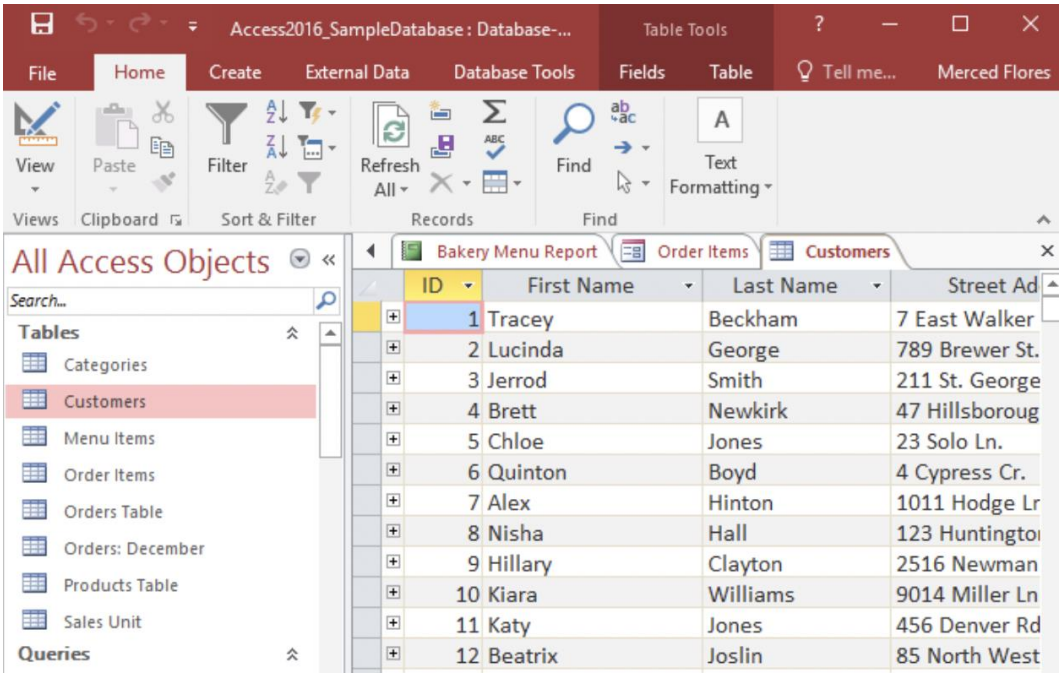


Рис.1.4 Інтерфейс Microsoft Access

Microsoft Excel є провідною програмою для роботи з електронними таблицями, потужним інструментом для візуалізації та аналізу даних. Він дозволяє піднести ваш аналітику на новий рівень. Excel використовується для зберігання, організації та маніпулювання даними. Він організовує числа та дані за допомогою формул та функцій. Аналіз в Excel використовується по всьому світу і використовується бізнесами всіх розмірів для проведення фінансового аналізу. Основне призначення цієї програми - збереження та аналіз даних [5].

Торговий відділ може використовувати Microsoft Excel у своїй роботі для різноманітних завдань та аналітики. Ось декілька способів, якими Excel може бути корисним для торгового відділу: Аналіз продажів: Excel може використовуватися для обробки даних про продажі, створення звітів та графіків, що дозволяють аналізувати тенденції продажів, ідентифікувати найбільш успішні товари або ринки, виявляти зростаючі або спадаючі продажі, а також здійснювати прогнозування майбутніх продажів, управління клієнтськими даними - Excel може бути використаний для створення таблиць або бази даних, де можна зберігати та організовувати інформацію про клієнтів, контакти, замовлення та історію взаємодії з клієнтами. За допомогою функцій фільтрації, сортування та підсумовування даних в Excel, торговий відділ може швидко відшукати необхідну інформацію про клієнтів.

На рисунку 1.5 зображено інтерфейс програми Microsoft Excel.

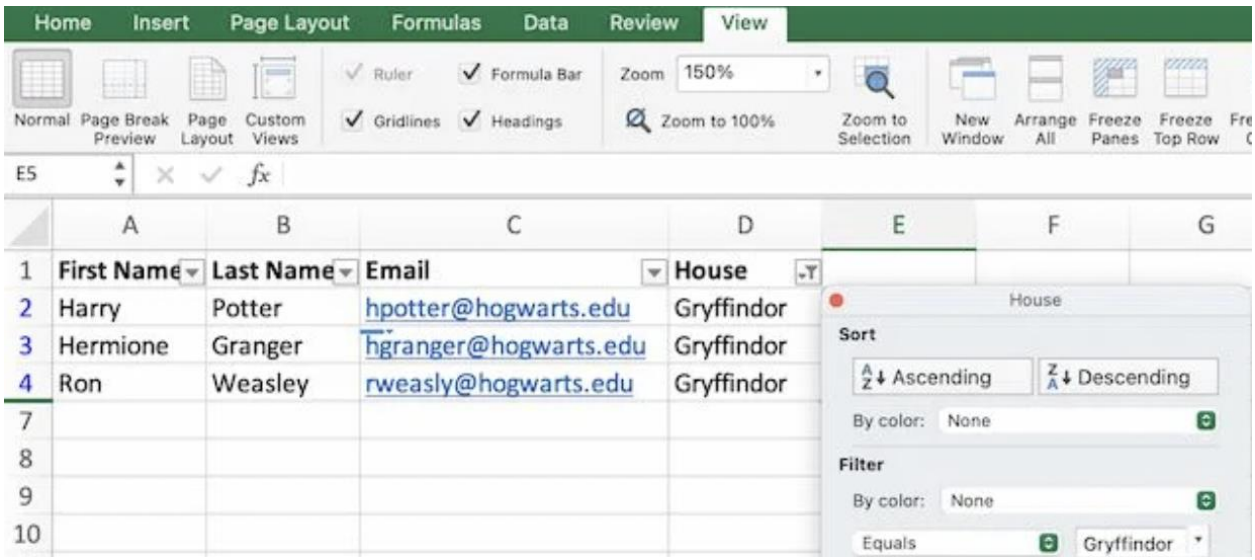


Рис.1.5 Інтерфейс Microsoft Excel

Google Sheets є безкоштовною веб-базованою програмою для роботи з електронними таблицями, яка надається Google в межах сервісу Google Drive. Додаток також доступний як десктопна програма на ChromeOS, а також як мобільний додаток на Android, Windows, iOS і BlackBerry. Google Sheets дозволяє користувачам редагувати, організовувати та аналізувати різні типи інформації. Він дозволяє співпрацю, і кілька користувачів можуть редагувати та форматувати файли в реальному часі, а всі зміни, зроблені в електронній таблиці, можуть бути відстежені за допомогою історії змін [6].

Торговий відділ може використовувати Google Sheets у своїй роботі для спільної роботи над даними, зберігання та обробки інформації онлайн. Ось декілька способів, якими Google Sheets може бути корисним для торгового відділу: спільна робота над таблицями: Google Sheets дозволяє спільно працювати над таблицями з різних місць і одночасно з іншими співробітниками. Торговий відділ може створювати спільні таблиці для зберігання даних про продажі, клієнтів, замовлення та іншу інформацію, що легко доступна та оновлюється в режимі реального часу, відстеження продажів та виробничих показників - За допомогою Google Sheets, торговий відділ може створювати таблиці та графіки, що відстежують продажі, прибутковість, запаси та інші виробничі показники. Це дозволяє торговому відділу аналізувати дані, відстежувати тенденції та приймати управлінські рішення на основі наявної інформації. На рисунку 1.6 наведено інтерфейс Google sheets.

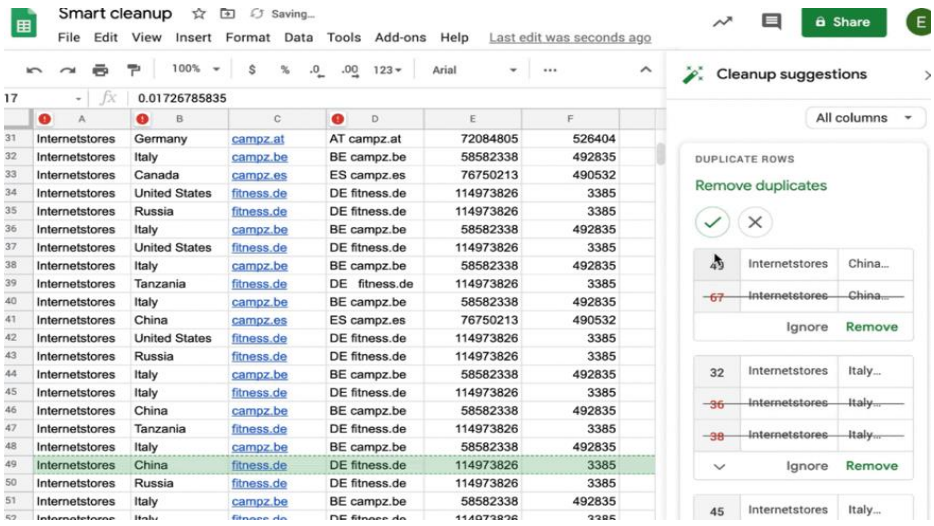


Рис. 1.6 інтерфейс Google sheets.

У таблиці 1.2 приведено порівняльну характеристику розглянутих додатків:

Таблиця 1.2. Порівняльна характеристика аналогів

Особливість	Microsoft Access	Google Sheets	Microsoft Excel
Функціональні можливості	Велика кількість можливостей для управління базами даних та автоматизації процесів	Обмежений функціонал порівняно з Access та Excel, але можливість спільної роботи та доступу до даних онлайн	Широкий спектр можливостей для обробки даних та аналітики
Керування базою даних	Оптимізовано для управління великими базами даних та створення складних структур	Обмежена можливість керування базою даних, простіше використання для менших обсягів даних	Не підтримує створення структурованих баз даних, але має функціонал для обробки даних
Спільна робота та доступ до даних	Можливість обміну даними та спільної роботи з іншими користувачами, але обмежена доступність з-за потреби встановлення Access на пристрої користувачів	Легка спільна робота над таблицями та доступ до даних онлайн, не потрібна встановлення окремого програмного забезпечення	Обмежена можливість спільної роботи, потреба в інсталяції Excel на пристроях користувачів
Аналітика та графіки	Обмежена можливість створення складних аналітичних звітів та графіків, більше спрямована на управління базами даних	Добре підходить для простих аналітичних звітів та графіків, простіше використання для розрахунків	Розширений набір інструментів для створення аналітичних звітів, графіків та розрахунків
Вартість	Включений у пакет Microsoft Office, вимагає ліцензії для використання	Безкоштовна версія з обмеженими можливостями, розширений функціонал є платним	Включений у пакет Microsoft Office, вимагає ліцензії для використання

Використання Microsoft Excel є досить ефективним, він забезпечує потужні функції аналізу даних, гнучкість та доступність для багатьох користувачів. Використання Excel дозволяє торговому відділу ефективно виконувати завдання зі збору, обробки та аналізу даних, що допомагає в прийнятті управлінських рішень та досягненні успіху в ринковій діяльності, але для оперативного формування і прийняття замовлень на продукцію наведених у таблиці 1.2 засобів недостатньо.

1.6. Обґрунтування доцільності проектування й розроблення Web-додатку для замовлення продукції

Використовуючи веб-додаток підприємства матимуть змогу в декілька кліків замовити партію необхідної для реалізації продукції. В свою чергу вся інформація про ці замовлення буде швидко поступати до менеджерів ТДВ «Яготинський маслозавод» ТМ «Яготинське для дітей». На основі замовлень можна буде формувати статистичну звітність по кожному замовнику та реалізованому товару.

Переваги веб-додатку:

- цілодобовий он-лайн доступ до сайту клієнтів та співробітників підприємства;
- захист інформації клієнта та компанії завдяки різним інтернет протоколам (таким як https та інші);
- збір та відображення статистики продажів продукції підприємства;
- зручний та зрозумілий інтерфейс.

В цілому цей веб-додаток мінімізує та спрощує замовлення необхідної продукції підприємства та збирає і зручно відображає інформацію про замовлення, що зробить роботу персоналу простішою та ефективнішою.

1.7. Концептуальна модель системи

Після того, як на основі наведеної у розділі 1.4.1 функціональної моделі, було виявлено та описано вузькі місця у процесі реалізації продукції, а також знайдено рішення у вигляді web-додатку для її замовлення ми можемо розробити функціональну модель стану TO-BE для торгового відділу ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод».

На рисунку 1.7 зображено оновлену функціональну модель, а на рисунку 1.8 зображено перший рівень декомпозиції оновленої моделі.

Доданий у схему ресурс у вигляді web-додатку спрощує процес замовлення продукції, а також призначений для формування звітів з продажів.

Рис. 1.7 Функціональна модель, стану TO-BE

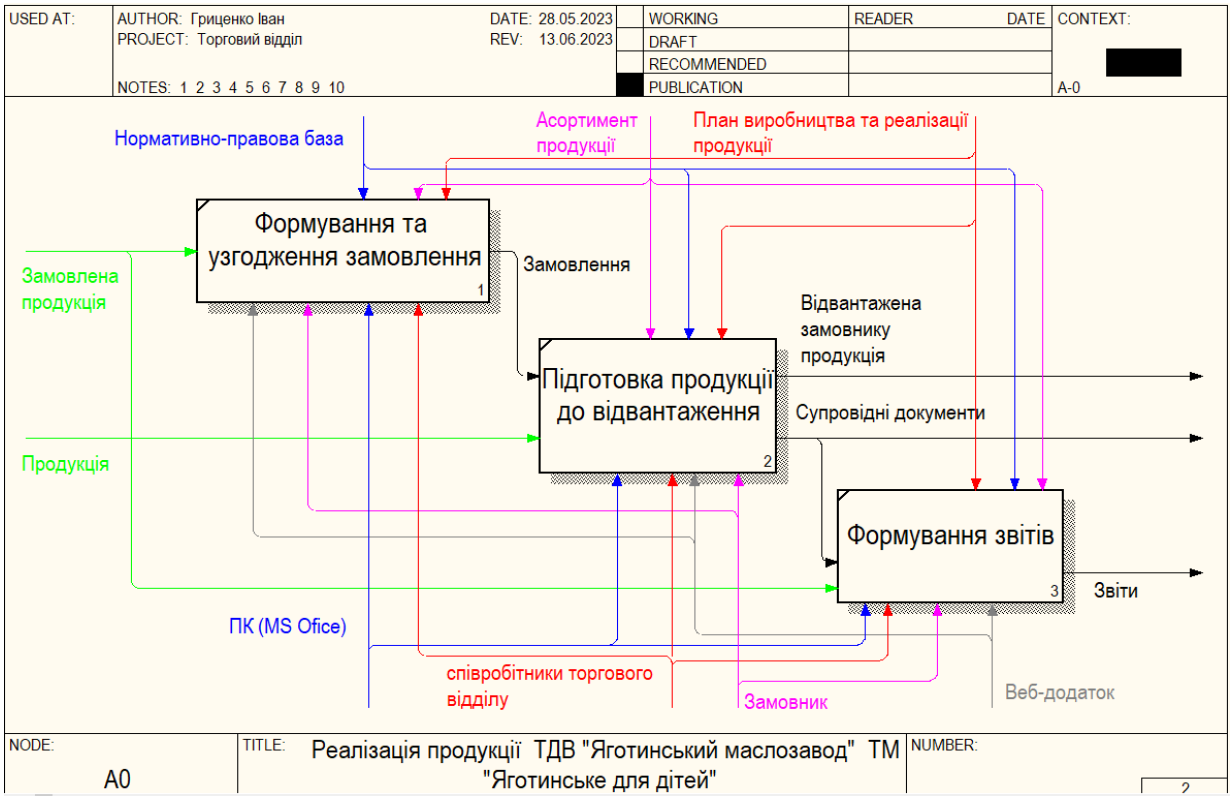


Рис. 1.8 Декомпозиція першого рівня функціональної моделі, стану ТО-ВЕ

1.8. Розрахунок економічного ефекту від впровадження системи

Методика розрахунків для визначення економічного ефекту від впровадження системи важлива, оскільки вона є основою техніко-економічного обґрунтування розробки автоматизованої системи і описана у джерелі [7].

Рівень новизни для розроблюваних задач оцінюється як "В" з використанням типових проектних рішень, що мають схожі рішення. Група складності алгоритму визначена як 2, що включає алгоритми обліку, звітності, статистики та пошуку.

Спочатку проведемо розрахунок витрат на оплату праці.

Визначення інформації, що використовується, наведено у таблиці 1.3.

Таблиця 1.3. Визначення виду інформації

Вид інформації	Позначення	К-сть наборів даних
Змінна	ЗІ	5
Нормативно-довідкова	НДІ	5
Банк (база) даних	БД	1
Обробка в режимі реального часу	РЧ	Так
Забезпечення телекомунікаційної обробки даних і управління віддаленими об’єктами	ТОУ	Ні

У таблиці 1.4. наведені витрати на розробку проекту.

Таблиця 1.4. Витрати на розробку проекту

Вид системи	Стадія розробки системи	
	Ескізний проект	Технічне завдання
Управління якістю продукції, управління технологічними процесами, управління стандартизацією, управління технічною підготовкою виробництва	В	В
	T_1=67	T_2=31

Зараз необхідно встановити тривалість етапів "технічний проект", "робочий проект" і "впровадження". Вхідними даними є:

- кількість форм вхідної інформації 5;
- кількість форм вихідної інформації 5;
- базове значення витрат часу для стадії «Технічний проект» - $T_{Б3}$ = 153;
- базове значення витрат часу для стадії «Робочий проект» - $T_{Б4}$ = 217;
- базове значення витрат часу для стадії «Впровадження» – $T_{Б5}$ = 80;

Витрати на розробку «технічного проекту» $T_з$ визначаються за (1.1).

$$T_з = T_{Б3} * k_п * k_о \quad (1.1)$$

де, $T_{Б3}$ – базове значення витрат часу для стадії «Технічний проект», яке визначається залежно від виду розроблюваної системи, $k_о$ – коефіцієнт новизни, $k_п$ – коефіцієнт трудомісткості робіт, що визначається за (1.2).

$$k_П = \frac{k_1 * m + k_2 * n + k_3 * p}{m + n + p} \quad (1.2)$$

де k_1, k_2, k_3 – поправочні коефіцієнти, m – кількість видів змінної інформації, n – кількість видів нормативно-довідкової інформації , p – кількість видів баз даних.

У нашому проекті, ступінь новизни становить 1.26, так як у нас присутній вид обробки в реальному часі.

Поправочні коефіцієнти будуть становити для k_1 – 1.0, k_2 – 0.72 та k_3 – 2.08.

$$k_П = \frac{1 * 5 + 0.72 * 5 + 2.08 * 1}{5 + 5 + 1} = 0{,}970$$

$$T_з = 153 * 0{,}970 * 1{,}26 = 186{,}99$$

Витрати часу на стадіях «робочий проект» T_4 і «впровадження» (T_5) визначається за формулою (1.3).

$$T_4 = T_{Б4} * k_п * k_о * k_с \quad (1.3)$$

де, $T_{Б4}$ – базове значення витрат часу для стадії «Технічний проект», яке визначається залежно від виду розроблюваної системи, $k_о$ – коефіцієнт новизни, $k_п$ – коефіцієнт трудомісткості робіт, а $k_с$ – коефіцієнт складності контролю вхідної та вихідної інформації визначається з таблиці.

Тому поправочні коефіцієнти будуть становити для k_1 – 1.1, k_2 – 0.58 та k_3 – 0,48, коефіцієнт новизни 1.32, а коефіцієнт складності контролю вхідної та вихідної інформації для нашого проекту буде становити 1.08.

$$k_П = \frac{1.1 * 5 + 0.58 * 5 + 0{,}48 * 1}{5 + 5 + 1} = 0{,}807$$

$$T_4 = 217 * 0{,}862 * 1{,}32 * 1{,}08 = 266{,}66$$

Для розрахунку витрат часу на стадії «впровадження» (1.4) маємо такі коефіцієнти: k_1 – 1.2, k_2 – 0.65 та k_3 – 0.54, коефіцієнт новизни 1.21, коефіцієнт складності контролю вхідної та вихідної інформації для нашого проекту буде становити 1.08.

$$T_5 = T_{Б5} * k_п * k_о * k_с \quad (1.4)$$

$$T_5 = 80 * 0{,}807 * 1{,}21 * 1{,}08 = 88{,}23$$

Отже, загальні витрати складають:

$$67 + 31 + 186{,}99 + 310{,}62 + 84{,}36 = 679{,}97$$

Тепер визначимо чисельність виконавців Ч за формулою 1.5

$$Ч = \frac{T_Σ}{Ф} \quad (1.5)$$

де Ф – кількість робочих днів на виконання проекту

Тому, враховуючи, що на виконання кваліфікаційної роботи дається 360 годин із 7-годинним робочим днем, тому на розробку виділено:

$$Ф = 360 / 7 = 51.42 \text{ дні}$$

Якщо брати час в місяцях, то визначаємо кількість місяців із розрахунку 22 робочих дні.

$$М = Ф / 22 = 51{,}42 / 7 = 2{,}34 \text{ місяці}$$

Отже на виконання такого проекту потрібна чисельність виконавців:

$$Ч = \frac{679,97}{51,42} = 13,22$$

Оплата праці виконавців підраховується за формулою 1.6.

$$V_1' = Ч * М * ЗП_{ПР} \quad (1.6)$$

де $ЗП_{ПР}$ – місячна оплата праці програміста, Ч – чисельність виконавців, М – кількість місяців роботи.

Приймемо розмір заробітної плати програміста 30000 грн, тоді загальна сума заробітних плат програмістів складає:

$$V_1' = 13 * 2.34 * 30000 = 928044 = \text{ грн}$$

Наступним пунктом витрат є витрати пов'язані з розробкою програми.

Дійсний річний фонд часу ПК у годинах дорівнює числу робочих годин у році для оператора, за винятком часу на технічне обслуговування і ремонт ПК (в середньому 5 год/міс + 6 робочих днів/рік)

$$Т_{ПК} = 2000 - (6 * 8 + 5 * 12) = 1892 \text{ год}$$

Оскільки під час виконання кваліфікаційної роботи здобувач в середньому витрачає 300 годин машинного часу, то величина фонду часу ПК:

$$Т_{ПК} = 1892 * (300 / 2000) = 283,8 \text{ год}$$

За формулою 1.6 розрахуємо поточні витрати на експлуатацію

$$V_1'' = З_{ОП} + З_{АМ} + З_{ел} + З_Р \quad (1.6)$$

де $З_{оп}$ – заробітна плата обслуговуючого персоналу, $З_р$ – витрати на поточний ремонт і технічне обслуговування комп'ютера, $З_{мат}$ – непрямі витрати пов'язані з експлуатацією комп'ютера, $З_{ам}$ – амортизаційні відрахування, що обраховується за формулою 1.7.

$$З_{АМ} = \frac{Ц_{пк}}{Н_а} \quad (1.7)$$

де $Ц_{пк}$ – балансова вартість комп'ютера, $Н_а$ – норма амортизаційних відрахувань, яка для комп'ютера 5.

Балансова вартість ПК обраховується за формулою

1. $$Ц_{пк} = Ц_р * (1 + k_{ун}) \quad (1.8)$$

де $Ц_р$ – ринкова вартість ПК, орієнтовно складає 40000 грн, $k_{ун}$ – коефіцієнт, що враховує витрати на установку комп'ютера, складає 0,12. Тому балансова вартість комп'ютера складає:

$$Ц_{пк} = 40000 * (1 + 0{,}12) = 44800$$

Амортизаційні відрахування будуть становити

$$З_{АМ} = \frac{44800}{5} = 8960 \text{ грн}$$

Витрати на електроенергію споживану комп'ютером, обчислюються за формулою 1.9.

$$З_{ел} = Р_{пк} * Т_{пк} * Ц_{ел} * А \qquad (1.9)$$

де $Р_{пк}$ – потужність ПК, $Т_{пк}$ – фонд корисного робочого часу, $Ц_{ел}$ – вартість 1 кВт електроенергії для підприємств, А – коефіцієнт інтенсивного використання ПК, складає 0,9.

Враховуючи потужність всіх комп'ютерів 0,4 кВт, ціна на електроенергію 1,86 грн/кВт, витрати на електроенергію складають:

$$З_{ел} = 0{,}4 * 283{,}8 * 1{,}86 * 0{,}9 = 190 \text{ грн}$$

Витрати на поточний ремонт складають і технічне обслуговування комп'ютера визначаються як 6% від балансової вартості комп'ютера.

$$З_р = 44800 * 0{,}06 = 2688 \text{ грн}$$

Непрямі витрати, пов'язані з експлуатацією комп'ютера визначаються як 5% від балансової вартості комп'ютера.

$$З_{мат} = 44800 * 0{,}05 = 2240 \text{ грн}$$

Заробітна плата обслуговуючого персоналу в середньому становить 10000 грн, то поточні витрати на експлуатацію будуть становити

$$V_1^{"} = 10000 + 8960 + 190 + 2688 + 2240 = 24078 \text{ грн}$$

А загальні витрати

$$V_1 = V_1^{`} + V_1^{"} = 928044 + 24078 = 952122 \text{ грн}$$

Наступним кроком є розрахунок витрат на підготовку приміщення та навчання персоналу.

Витрати на підготовку приміщення відсутні – $V_3 = 0$, так як приміщення вже присутнє.

Витрати на навчання персоналу – можна припустити, що навчання буде тривати один місяць, тому можна сказати $V_4 = 4500$ грн.

Витрати на придбання та встановлення комп'ютера дорівнюють балансовій вартості комп'ютера, тому $V_2 = Ц_{пк} = 44800$ грн.

Загальна вартість розробки і впровадження системи вираховуються за формулою 1.10

$$V_\Sigma = V_1 + V_2 + V_3 + V_4 \quad (1.10)$$

$$V_\Sigma = 952122 + 44800 + 0 + 4500 = 1001422 \text{ грн}$$

Оскільки норма амортизаційних втрат для комп'ютерних систем $Н_А = 5$, то для обрахування річного економічного ефекту слід брати до розгляду величину:

$$V_p = \frac{V_\Sigma}{Н_А} = \frac{1001422}{5} = 200284{,}4 \text{грн}$$

Термін окупності визначається за формулою 1.11

$$T_{ок} = \frac{1}{К_{ЕФ}} \quad (1.11)$$

де $К_{еф}$ економічної ефективності визначається за формулою 1.12

$$К_{ЕФ} = \frac{П_р}{V_p} \quad (1.12)$$

де $П_р$ – річний прибуток від впровадження системи, який можна орієнтовно оцінити в 90000 грн, тому економічний ефект від впровадження складатиме

$$К_{ЕФ} = \frac{90000}{200284{,}4} = 0{,}44$$

Відповідно термін окупності системи складатиме:

$$T_{ок} = \frac{1}{0{,}44} = 2{,}27 \text{ роки}$$

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ

Дослідивши діяльність підприємства ми можемо скласти технічне завдання [8]:

2.1. Загальні положення

2.1.1. Найменування: «Web-додаток замовлення продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод».

2.1.3 Відповідно до ДСТУ, вимоги до оформлення результатів робіт зі створення системи повинні дотримуватися на всіх етапах розробки. Порядок передачі та оформлення результатів повинен визначатися відповідно до змісту проекту та календарного плану виконання робіт.

2.2. Призначення та цілі створення системи

Призначення веб-додатку – забезпечення зручного функціоналу для замовлення продукція Яготинського маслозаводу та збереження інформації про замовлення і відображення необхідної статистики по замовленням. Таким чином додаток дозволяє зберігати, аналізувати та переглядати великі обсяги даних про замовлену продукцію, що дозволить значно підвищити ефективність виробництва та націлитись на найбільш популярні позиції.

Цілі:

- Створення зручного інтерфейсу для користувачів, як для клієнтів так і для парцівників;
- Аналіз обсягу даних, що дозволить зрозуміти, який товар має більший попит;
- Підвищення ефективності роботи різних відділів, які пов’язані з виробництвом та доставкою продукції на місця призначення;
- Забезпечення постійного доступу до додатку та, відповідно, до бази даних;
- Забезпечення безпеки збережених даних;
- Доступ до статистичних даних, які будуть створюватись відповідно до замовленої продукції.

2.3. Характеристика об’єкта автоматизації

«Яготинський маслозавод», а саме торговий відділ є об’єктом дослідження. Даний відділ відповідає за доставку товару підприємства по всій Україні.

Розробка веб-додатку проводиться для зручного замовлення продукції клієнтами та аналізу отриманих даних відповідними працівниками.

Веб-додаток для замовлення продукції є предметом автоматизації, що дозволить вести зручний облік замовленої продукції для працівників та зручний інтерфейс замовлення товару для клієнтів.

Користувачами веб-додатку є:

- Клієнти, які зможуть замовити продукції у зручному додатку.
- Працівники відділу, які аналізують отриману інформацію і укладають договори з замовниками.

2.4. Вимоги до системи

Веб-додаток повинен складатися з двох частин – для замовників та для працівників.

Працівники повинні мати можливість переглядати дані про замовлену продукцію, та аналізувати отримані данні.

Замовники повинні мати змогу швидко зробити замовлення продукцію для свого підприємства.

Web-додаток повинен:

- Мати зручний і зрозумілий інтерфейс, як для клієнтів так і для працівників;
- Бути захищеним та надійним від різних атак на Web-додаток ;
- Мати змогу працювати в різних браузерах та на різних пристроях;
- Забезпечувати швидку роботу з інтерфейсом;
- Повинен бути адаптивним і підтримувати різні види екранів та моніторів;
- Мати змогу розширення в майбутньому.

2.4.1. Функціональні вимоги до веб-додатку.

Відповідно до зазначених вимог до системи, веб-додаток має складатися з двох частин – для працівників, та для замовників.

Перелік функцій, які повинні виконувати веб-додаток показаний в таблиці 2.1

Таблиця 2.1. Перелік функцій, вхідної та вихідної інформації

№ п/п	Найменування функції	Вхідна інформація	Вихідна інформація
1	Перегляд інформації про продукцію	Таблиця «продукти»	Головна сторінка, сторінка «Продукти»
2	Редагування інформації про продукцію	Таблиця «продукти»	Сторінка адмін-панелі «Продукти»
3	Додавання інформації про продукцію	Таблиця «продукти»	Сторінка адмін-панелі «додати продукти»
4	Видалення інформації про продукцію	Таблиця «продукти»	Сторінка адмін-панелі «Продукти»
5	Перегляд інформації про замовлення продукції	Таблиця «замовлення»	Сторінка адмін-панелі «Замовлення»
6	Відображення статистики	Таблиця «замовлення»	Сторінка адмін-панелі «Звітність»
7	Замовлення продукції	Таблиця «замовлення»	Головна сторінка «продукти»
8	Авторизація	Таблиця «юзери»	Сторінка авторизації «Юзери»

2.4.2. Вимоги до гарантійної підтримки

Гарантійний термін має бути не менше одного року починаючи з моменту прийому-передачі веб-додатку. У разі виявлення помилок або дефектів протягом гарантійного терміну, постачальник повинен забезпечити безкоштовний ремонт

веб-додатку протягом десяти робочих днів після отримання повідомлення про помилку. Крім того, угода з постачальником веб-додатку повинна чітко визначати вимоги до гарантійної підтримки.

Постачальник веб-додатку повинен забезпечувати гарантійну підтримку, що включає:

- Має бути забезпечено доступ до оновлень та патчів, щоб усунути виявлені уразливості.
- Має бути забезпечена підтримка під час вирішення інцидентів та проблем, пов'язаних з функціоналом веб-додатку.
- Необхідно надавати консультації щодо роботи веб-додатку та вирішення технічних проблем.
- Необхідно забезпечити можливість проведення безоплатного ремонту програмного забезпечення веб-додатку у разі виявлення помилок та дефектів протягом гарантійного терміну.

2.4.3. Вимоги до лінгвістичного забезпечення системи

Веб-додаток повинен бути на Українській мові:

- Перевірка правопису та граматики введеного користувачем тексту на різних мовах.
- Підтримка різних мовних форматів даних, таких як дата, час та числа.

2.4.4. Вимоги до апаратного забезпечення

Беручи до уваги, що ця система є веб-додатком, для її використання достатньо мати пристрій, який може запускати і відображати вміст у сучасному браузері, такому як Google Chrome. У таблиці 2.2. наведені рекомендовані системні вимоги.

Таблиця 2.2. Системні вимоги

Операційна система	Рекомендовані системні вимоги
MacOS X	Macbook Pro 2011, Macbook Air 2012 або більш нові версії; macOS X 10.9 або вище
Android	Версія 4.2 або вище
iOS	Версія 12 або вище
Windows	процесор Core i5 2-го покоління (от 2 ГГц), Core i5 3-го чи 4-го покоління або аналогічні; Windows 7 або вище
Linux	Робота залежить від дистрибутивів, драйверів та середовища

2.5. Склад і зміст робіт по створенню системи

В таблиці 2.3. наведені стадії створення системи і терміни виконання робіт

Таблиця 2.3. Найменування робіт при створенні системи

№ п/п	Найменування робіт	Строки виконання робіт
1	Передпроектне дослідження об’єкта автоматизації	01.03.2023
2	Технічне завдання	01.04.2023
3	Технічних проект	08.04.2023
4	Оформлення документації	29.04.2023

2.6. Порядок контролю та приймання системи

Система вводиться в ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод». При введенні в дію, система повинна пройти приймальні випробування відповідно до ДСТУ 3974-2000 [9].

- З залученням замовника, розробники проводять спільні випробування системи з метою перевірки її функціональності та готовності до експлуатації. Розробник складає план випробувань, який підлягає затвердженню замовником.
- Система надається для дослідної експлуатації відповідно до технічного завдання та інструкцій для користувача. Після проведення дослідної

експлуатації формується список необхідних вдосконалень та рекомендовані терміни їх виконання.

2.7. Вимоги до складу та змісту робіт із підготовки до введення системи в дію

Для того, щоб ввести систему в дію, замовник виконує ряд підготовчих робіт, які включають:

- Укомплектування необхідних технічних засобів;
- Проведення дослідної експлуатації та введення системи в дію.

2.8. Вимоги до документації

- У планах передбачається створення комплексної документації, яка буде включати технічне завдання, технічний проект та інструкцію з використання системи.
- Документація повинна розроблятися згідно вимог Державних стандартів серії 19 «Єдина система програмної документації» [10] та серії 24 «Єдина система стандартів автоматизованих систем управління» [11].

2.9. Джерела розробки

Для створення даного технічного завдання досягнуті наступні джерела інформації:

- ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання [12];
- ДСТУ 3973–2000 Система розроблення та поставлення продукції на виробництво [13].

РОЗДІЛ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ

3.1. Інформаційне забезпечення системи

Додаток було розроблено за допомогою NoSQL бази даних Firebase.

Для розробки додатку були створені наступні сутності у базі даних:

- Замовлення – містить інформацію про замовлення які зробили клієнти;
- Продукти – містить інформацію про продукти, які реалізує компанія;
- Користувач адмін-панелі – містить інформацію про користувачів, які мають доступ до адмін.-панелі;
- Замовлені продукти – підсутність сутності «Замовлення», яка містить інформацію про замовленні продукти певним користувачем.

Сутності, безпосередньо, описані і використані в програмному коді, наведеному у додатку В.

3.2. Алгоритмізація та реалізація комплексу задач автоматизації

При розробленні веб-додатку було обрано сучасні та зручні інструменти, які допомагають при розробці, такі як бібліотеки та фреймворки, які будуть описані нижче.

Серверна частина проекту побудована на основі Firebase, потужної платформи для розробки веб-додатків. Firebase надає розробникам надійні і масштабовані інструменти, що сприяють ефективній роботі з базою даних та серверною логікою. Згідно з документацією Firebase, вона має широкий спектр функцій, включаючи аутентифікацію користувачів, зберігання та синхронізацію даних в режимі реального часу, зберігання файлів та багато іншого.

Основним інструментом для роботи з серверною частиною став Firebase Cloud Firestore - це гнучка, масштабована база даних для розробки мобільних, веб-додатків та серверів від Firebase та Google Cloud. Як і база даних Firebase Realtime, вона синхронізує дані між клієнтськими програмами за допомогою прослуховувачів у реальному часі і пропонує автономну підтримку для мобільних пристроїв та Інтернету, щоб ви могли створювати чуйні програми, які працюють незалежно від затримки в мережі або підключення до Інтернету. Cloud Firestore

також пропонує безшовну інтеграцію з іншими продуктами Firebase та Google Cloud, включаючи хмарні функції [14].

Ключові можливості Cloud Firestore:

- Гнучкість – Модель даних Cloud Firestore підтримує гнучкі ієрархічні структури даних. Зберігайте свої дані у документах, організованих у колекції. Документи можуть містити складні вкладені об'єкти на додаток до вкладених колекцій[15].
- Виразні запити - У Cloud Firestore ви можете використовувати запити для отримання окремих, певних документів або для отримання всіх документів у колекції, які відповідають параметрам запиту. Запити можуть включати кілька пов'язаних фільтрів та поєднувати фільтрацію та сортування. Вони також індексуються за замовчанням, тому продуктивність запитів пропорційна розміру набору результатів, а не набору даних;
- Оновлення у реальному часі - Як і база даних у реальному часі, Cloud Firestore використовує синхронізацію даних для оновлення даних на будь-якому підключеному пристрої. Однак він також призначений для ефективного виконання простих одноразових запитів на вибірку;
- Офлайн-підтримка - Cloud Firestore кешує дані, які активно використовує вашу програму, тому програма може записувати, читати, прослуховувати та запитувати дані, навіть якщо пристрій знаходиться в автономному режимі. Коли пристрій повертається до мережі, Cloud Firestore синхронізує всі локальні зміни з Cloud Firestore;
- Створено для масштабування - Cloud Firestore пропонує вам найкраще із потужної інфраструктури Google Cloud: автоматичну реплікацію даних у кількох регіонах, надійні гарантії узгодженості, атомарні пакетні операції та підтримку реальних транзакцій. Ми розробили Cloud Firestore для обробки найскладніших робочих навантажень баз даних із найбільших у світі додатків;

Для під’єднання Firebase до проекту потрібно виконати декілька кроків:

Першим кроком потрібно створити проект на сайті Firebase, як наведено на рис 3.1.

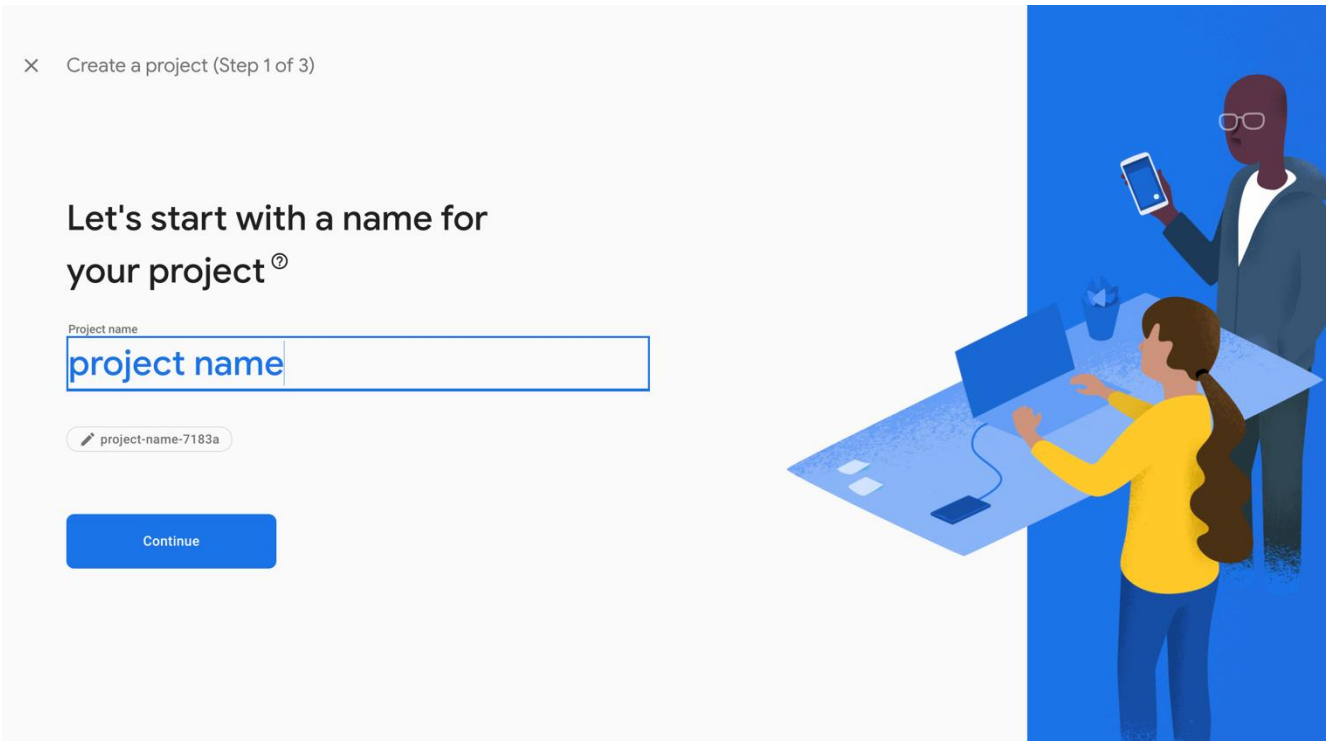


Рис.3.1. Перший етап під’єднання Firebase до проекту

Далі ми повинні встановити пакети залежностей проекту прописавши команду «npm install firebase» в консолі. Потім необхідно скопіювати параметри доступу до Firebase, які були згенеровані сервісом при створенні проекту.

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCWs61UwGIk-EP-icHt91bW96RGliTlAr8",
  authDomain: "project-name-7183a.firebaseapp.com",
  projectId: "project-name-7183a",
  storageBucket: "project-name-7183a.appspot.com",
  messagingSenderId: "434329267302",
  appId: "1:434329267302:web:7b884503eebe7e18e211a7"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Рис.3.2. Приклад згенерованого файлу ініціалізації проекту для розробки

Ми повинні створити файл в проекті і додати туди наведений нижче код.

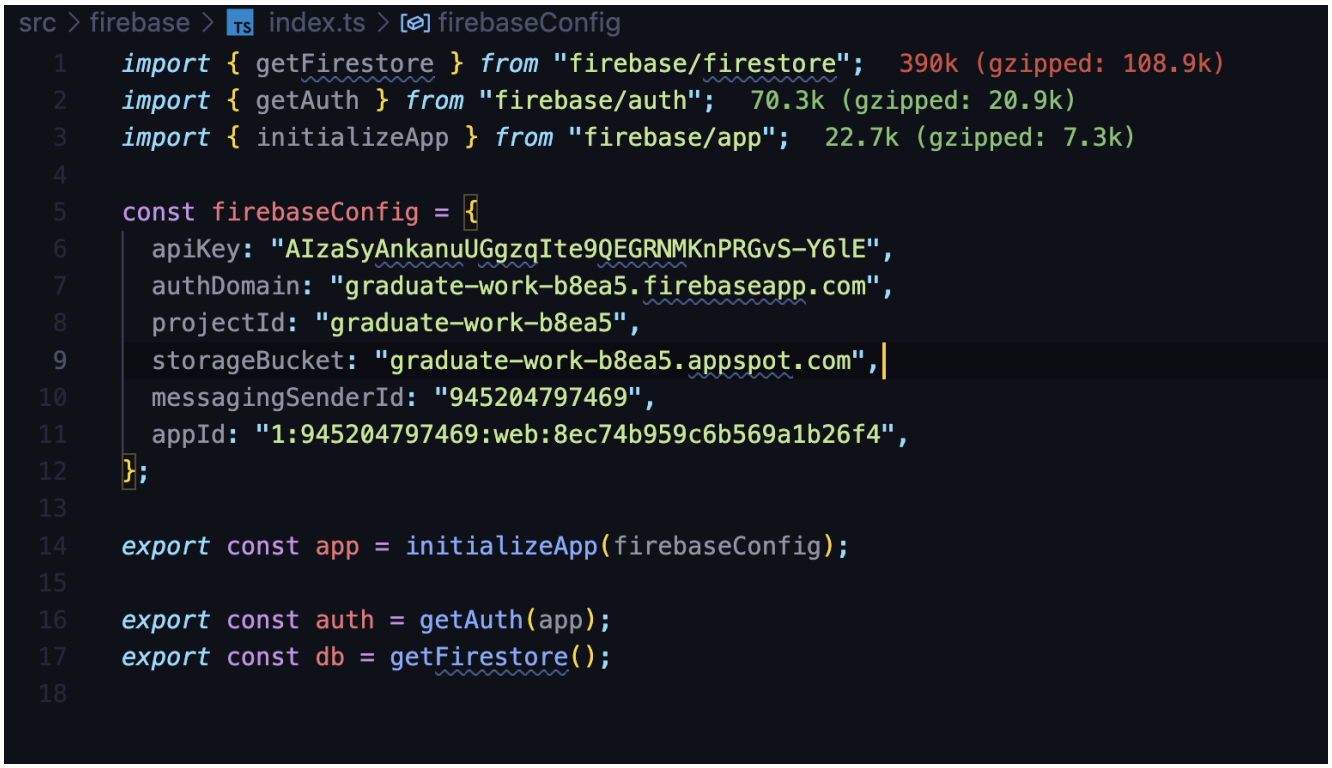


Рис.3.3. Інтеграція Firebase до проекту

Після підключення та налаштування конфігу ми повинні експортувати з файлу декілька змінних, таких як “auth” – змінна для, за допомогою якої на проекті буде проводитись авторизація, згідно з документацією [16], змінна “app” – безпосередньо ця змінна включає в себе данні про ініціалізацію проекту, ну і змінна “db” – для взаємодії з нашими даними, які зберігаються в Firebase cloud firestore.

Після того, як зв’язок з серверною частиною налаштований, необхідно починати писати логіку для взаємодії з сервером. Першим кроком необхідно створити у проекті сховище для збереження інформації. Для початку необхідно обрати метод збереження даних у проекті, переглянемо наступні варіанти:

- Redux і Redux-toolkit - Redux — це контейнер передбачуваного стану для програм JavaScript. Це допомагає вам писати програми, які ведуть себе узгоджено, запускаються в різних середовищах (клієнт, сервер і рідне) і які легко тестувати. Крім того, він забезпечує чудовий досвід для розробників, наприклад редагування коду в реальному часі [17]. Redux-toolkit - це офіційна бібліотека для написання логіки на Redux. Вона охоплює ядро Redux і містить пакети та функції, які, на нашу розробників, необхідні для створення програми з використанням

Redux. Redux Toolkit використовує передові практики, спрощує більшість завдань Redux, запобігає поширеним помилкам і полегшує написання програм Redux [17].

- MobX - це бібліотека, яка робить керування станом простим і масштабованим завдяки прозорому застосуванню функціонального реактивного програмування.

Розглянемо детальніше MobX [18]:

1. Прямий – дозволяє писати мінімалістичний код без шаблонів, який відобразить наміри. Намагаєтеся оновити поле запису? Просто використовуйте звичайне призначення JavaScript — система реагування виявить усі ваші зміни та розповсюдить їх туди, де вони використовуються. Під час оновлення даних в асинхронному процесі не потрібні спеціальні інструменти.

2. Оптимальна візуалізація без зусиль - Усі зміни та використання ваших даних відстежуються під час виконання, створюючи дерево залежностей, яке фіксує всі зв’язки між станом і результатом. Це гарантує, що обчислення, які залежать від вашого стану, наприклад компоненти React, виконуються лише тоді, коли це вкрай необхідно. Немає необхідності вручну оптимізувати компоненти за допомогою схильних до помилок і неоптимальних методів, таких як запам’ятовування та селектори.

- useContext - це хук React, який дозволяє передавати дані через компоненти безпосередньо без необхідності передавати їх через вкладені пропси. Ви можете використовувати useContext, щоб отримати доступ до глобального стану, який був розміщений у провайдері, та оновлювати його через відповідний контекст.

Для того щоб обрати кращий стейт менеджер необхідно зробити порівняльну таблицю табл 3.1.

Таблиця 3.1. Порівняльна таблиця методів збереження інформації

	useContext	**Redux**	**MobX**

Складність	Простий у використанні, але потребує більше ручного коду для організації стану	Потребує більше налаштування та додаткових пакетів	Легкий у використанні та не потребує багато додаткового коду
Реактивність	Неявна реактивність, компоненти оновлюються при зміні стану контексту	Неявна реактивність, компоненти оновлюються при оновленні стору	Явна реактивність, компоненти оновлюються при зміні спостережуваних значень
Управління станом	Використовує контекст для передачі та оновлення стану	Централізоване збереження стану в Redux-сторі	Децентралізоване збереження стану в MobX-сторах
Додаткові пакети	Не потребує додаткових пакетів	Для розширеної функціональності потрібно встановити додаткові пакети, такі як redux-thunk або redux-saga	Не потребує додаткових пакетів для базового використання
Розмір коду	Зазвичай має менший розмір коду, оскільки не потребує додаткових бібліотек	Може мати більший розмір коду через налаштування Redux та додаткові пакети	Має помірний розмір коду, включаючи декларативні змінювачі та спостережувані значення
Спрощена архітектура	Так, використовується пряме передавання стану через контекст	Так, за допомогою централізованого збереження стану та дій	Так, за допомогою декларативних змінювачів та спостережуваних значень
Екосистема	Вбудований у React, немає додаткових вимог або залежностей	Велика екосистема з багатьма розширеннями	Велика екосистема з підтримкою документації та інструментами

Після уважного аналізу таблиці-порівняння, було обрано useContext, як метод для управління станом проекту із-за деяких переваг, по-перше, useContext

відповідає потребам у простоті та швидкості розробки. Його використання не потребує додаткових пакетів або налаштувань, оскільки він вже входить до складу React, по-друге, useContext пропонує пряме передавання стану через контекст, що спрощує архітектуру додатку.

Для використання useContext нам необхідно зробити contextProvider в який ми маємо обгорнути наш додаток, для отримання інформації у будь якій частині проекту.

Рис.3.4. Приклад провайдеру хука useContext для збереження інформації про замовлення

Рис.3.5. Обертання кореневого файлу App у провайдери різних контекстів

Далі, для того щоб отримати данні з firebase cloud firestore необхідно створити функцію яка буде при запуску сторінки «підписуватись» на обрану

колекцію, ця «підписка» відбувається завдяки функції з cloud firestore під назвою «onSnapshot», в яку необхідно першим параметром передати також функцію «collection» в яку ми передаємо конфігурацію бази даних, та назву колекції інформацію з якої ми хочемо отримати, а другим параметром «onSnapshot» є коллбек функція, в середні якої ми пишемо свою логіку для збереження даних, вже безпосередньо, у самому додатку. Рис 3.6 наведена функція, яка буде отримувати данні з обраної колекції в реальному часі.

```
39  const fetchOrders = () => {
40    const unsub = onSnapshot(collection(db, "orders"), (doc) => {
41      // eslint-disable-next-line
42      const orders = doc.docs.map((data: any) => data.data()) as Order[];
43      const orderedProductsByDate = orders.sort(
44        (a, b) => b?.orderTime?.seconds - a?.orderTime?.seconds
45      );
46      setOrders(orderedProductsByDate);
47      return () => {
48        unsub();
49      };
50    });
51  };
52  useEffect(() => {
53    fetchOrders();
54  }, []);
```

Рис 3.6 Функція яка буде брати данні з обраної колекції в реальному часі

На наступному етапі необхідно зробити функцію яка буде додавати значення до сутностей у базі даних, наприклад, додавати нове замовлення до колекції «замовлення». Для цього необхідно створити функцію яка буде здійснювати підготовку даних для відправки і відправляти їх у базу даних за допомогою функції під «setDoc», в яку необхідно передати відповідні параметри та підготовлені раніше дані.

```
const handleAddOrder = async (order: Omit<Order, "orderTime">) => {
  const currentTime = await Timestamp.now();

  try {
    await setDoc(doc(db, "orders", generateRandomId()), {
      ...order,
      orderTime: currentTime,
    });
  } catch (error) {
    console.error("Error: ", error);
  }
};
```

Рис 3.7 Функція, яка додає продукт до колекції «orders»

Також необхідно зробити функції для зміни значень сутностей або їх видалення. Логіка функції по зміні значень елементів колекції полягає у тому що, спочатку необхідно вказати шлях до колекції, яку ми хочемо змінити, передавши до функції «doc» необхідні параметри, а далі потрібно викликати функцію «updateDoc» в яку передаємо вказані раніше параметри і значення яке необхідно замінити.

Для видалення продукту з колекції необхідно викликати функцію «deleteDoc» в яку ми передаємо необхідні параметри та id елементу який видаляється. Функція яка змінює значення необхідного елементу сутності, наведена на рис 3.8., а функція яка видаляє елемент з колекції за його id - на рис 3.9.

```
const handleEditProduct = async (product: Product) => {
  const updateProduct = doc(db, "products", product.fid);
  // eslint-disable-next-line
  const { fid, ...pushedProduct } = product;
  await updateDoc(updateProduct, { ...pushedProduct });
};
```

Рис 3.8. Функція, яка змінює значення необхідного елементу сутності

```
const handleRemoveProduct = async (fid: string) => {
  const docRef = doc(db, "products", fid);

  deleteDoc(docRef).catch((error) => {
    console.error("Помилка при видаленні продукту:", error);
  });
};
```

Рис 3.9. Функція, яка видаляє елемент з колекції за його id

В цілому логіка роботи Firebase cloud firestore укладається в можливості створення сутностей чи колекцій з якими можна взаємодіяти в реальному часі, видаляти чи редагувати інформацію за певними параметрами чи просто відстежувати всі їх дані.

Після того як ми додамо різні дані до Firebase cloud firestore він матиме такий вигляд, як зображено на рис. 3.10.

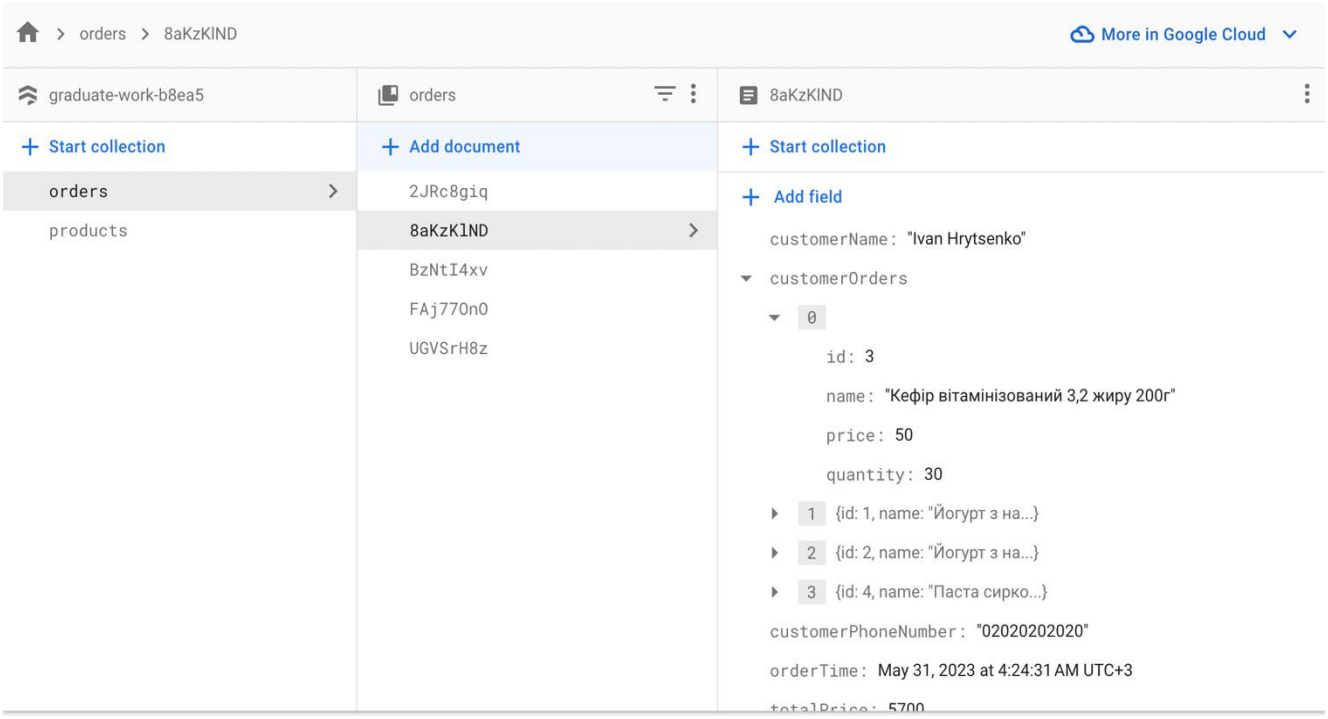


Рис. 3.10. Вигляд колекція cloud firestore

Функції Firebase не закінчуються на cloud firestore, також тут присутній модуль для реєстрації і авторизації користувачів, за допомогою якого буде створено користувача, який матиме доступ до адмін.-панелі проекту.

Firebase аутинтифікація - Більшості програм необхідно знати особу користувача. Знання особистості користувача дозволяє програмі безпечно зберігати дані користувача у хмарі і забезпечувати однаковий персоналізований досвід на всіх пристроях користувача.

Firebase Authentication надає серверні служби, прості у використанні SDK і готові бібліотеки інтерфейсу користувача для автентифікації користувачів у вашому додатку. Він підтримує автентифікацію з використанням паролів, телефонних номерів, популярних постачальників федеративних посвідчень, таких як Google, Facebook та Twitter тощо.

Для початку роботи з Firebase Authentication на сайті необхідно натиснути на вкладку Authentication і натиснути кнопку «почати». Далі нам буде запропоновано вибрати метод, за яким будуть реєструватися чи авторизовуватись наші користувачі, тут перелічені такі способи:

- За допомогою емейлу та паролю;
- За допомогою номеру телефону;
- Анонімна авторизація;
- За допомогою сервісів google, facebook, github, twitter, apple та інші.

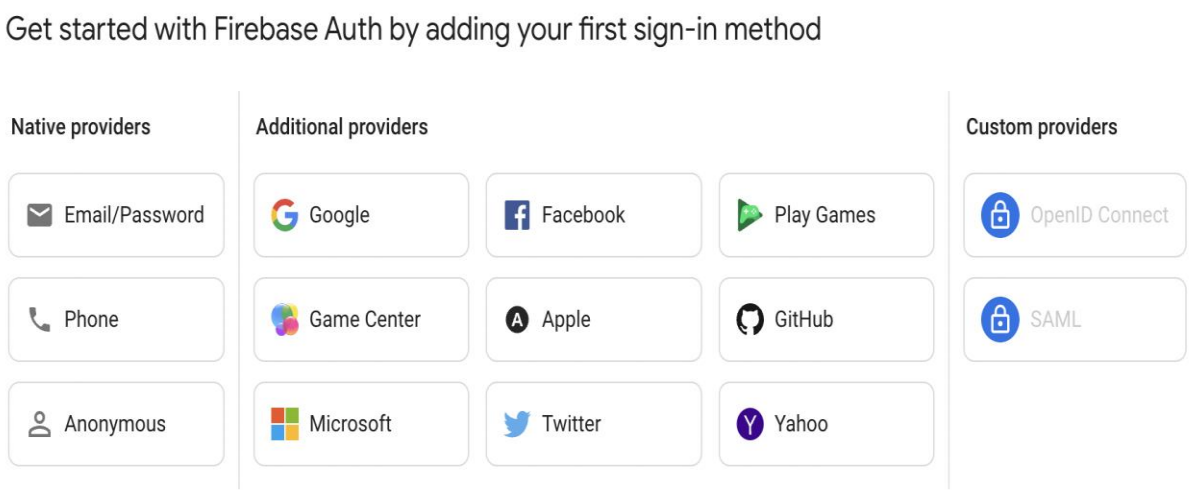


Рис. 3.11. Методи авторизації і реєстрація які можна додати до свого додатку.

В додатку буде забезпечена не тільки можливість авторизуватись для доступу в адмін-панель, що захистить данні від несанкціонованого доступу, а аккаунти, безпосередньо, будуть створюватись у панелі керування Firebase authentication (див. рис 3.12).

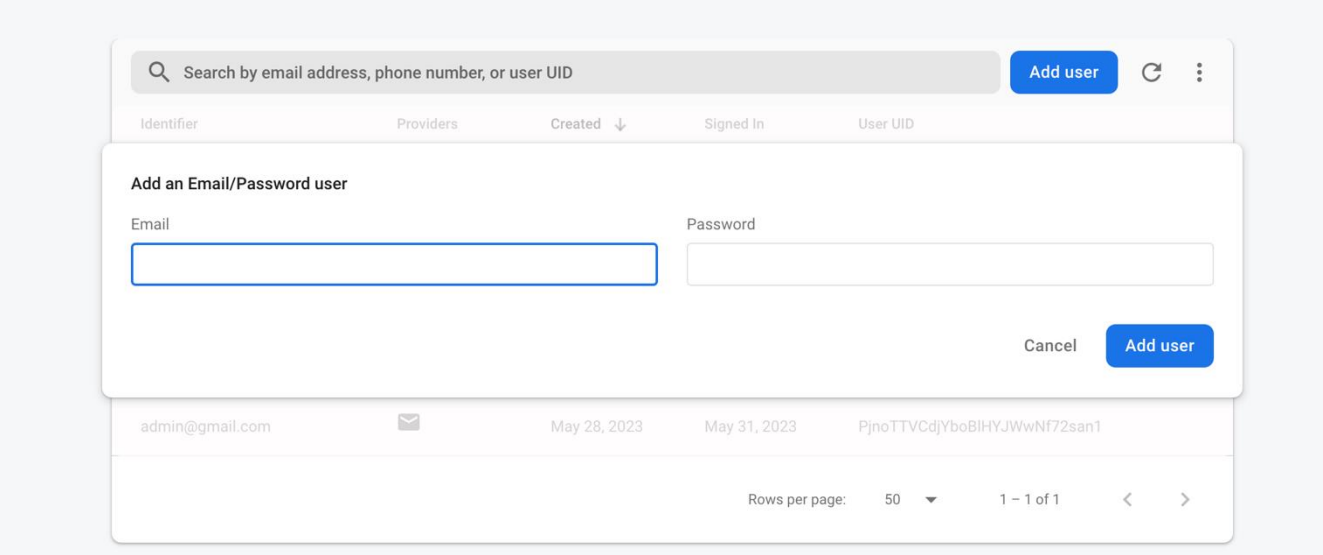


Рис. 3.12. Можливість зареєструвати нового користувача через панель керування Firebase authentication

Авторизація у додатку відбувається за допомогою функції «signInWithEmailAndPassword», в яку необхідно передати параметри доступу, введений е-мейл та пароль, як зображено на рис. 3.13.

```
const handleLogin = async (userLoginData: UserLoginFields) => {
  try {
    await signInWithEmailAndPassword(
      auth,
      userLoginData.email,
      userLoginData.password
    );
    navigate("/");
    // eslint-disable-next-line
  } catch (error: any) {
    error.message && setError(error.message);
    console.log(error.message);

    setTimeout(() => {
      setError(null);
    }, 3000);
  }
};
```

Рис. 3.13. Функція авторизації у додатку

Закінчивши роботу з серверною частиною необхідно побудувати інтерфейс для користувачів, для цієї задачі мною було використано бібліотеку JavaScript для розробки користувацьких інтерфейсів – React. Ця бібліотека забезпечує декларативний підхід до створення компонентів, який спрощує управління станом та маніпулювання DOM [19].

Також для полегшення роботи зі створення та стилізації деяких компонентів була обрана бібліотека Material UI. Material UI – це бібліотека компонентів React з відкритим кодом, яка реалізує Material Design від Google. Він містить повну колекцію готових компонентів, які готові до використання у виробництві одразу після вилучення, інтерфейс Material UI чудовий за дизайном і має набір параметрів налаштування, які спрощують реалізацію власної системи дизайну на основі наших компонентів [20].

Особливу увагу треба було приділити формам та роботі з ними, для цього мною були обрані такі бібліотеки, як Formik та Yup. Formik — це невелика бібліотека, яка допоможе вам із 3 найбільш неприємними частинами при роботі з формами:

- Отримання значень у стан форми та вихід із нього.
- Перевірка та повідомлення про помилки.
- Обробка подання форми.

Розташувавши все вищезазначене в одному місці, а застосування Formik дозволяє спростити тестування і рефакторинг [21].

Yup - це конструктор схем для аналізу та перевірки значень під час виконання [22].

Для зручної навігації по сторінкам була обрана бібліотека React-router-dom, яка дозволяє декларативно визначати шляхи та зв'язки між різними компонентами у додатку. Завдяки цьому, можна організувати навігацію між різними сторінками або розділами нашого інтерфейсу. Використання React Router DOM спрощує управління станом навігації і дозволяє змінювати вміст сторінок без необхідності перезавантажувати всю сторінку. Це забезпечує більш плавну та швидку взаємодію з користувачем [23].

Після установки усіх необхідних бібліотек до проекту, був створений файл «theme.ts» у якому описано головні, другорядні та допоміжні кольори проекту (див. рис. 3.14).

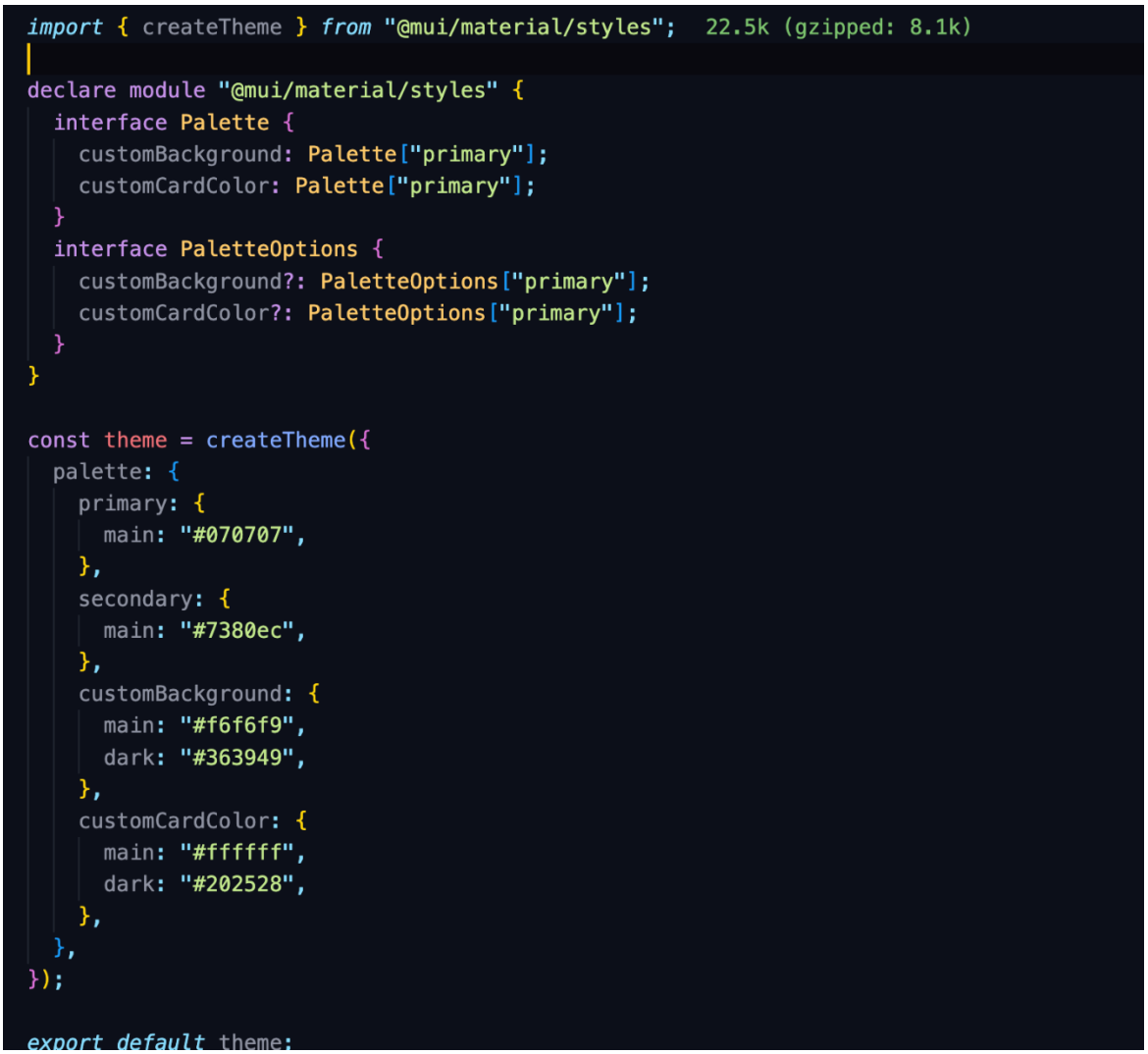


Рис. 3.14 Налаштування файлу «theme.ts»

Закінчивши з первинними налаштуваннями проекту необхідно визначити сторінки, які будуть створені у проекті. Основні сторінки для функціонування проекту:

- Головна – сторінка на якій клієнти можуть дізнатися інформацію про підприємство та зробити замовлення продукції;
- Сторінка авторизації – на ній працівник може авторизуватись і отримати доступ до панелі керування;

- Сторінка з усіма продуктами – це сторінка панелі керування, на якій працівник бачить всі товари та може їх, за необхідністю, відредагувати;
- Сторінка з таблицею замовлень – це сторінка на якій зберігається інформація про всі замовлення;
- Сторінка зі статистикою – на цій сторінці працівник може переглянути статистику по продажу товарів за певний проміжок часу;
- Сторінка на з формою – на цій сторінці буде форма, за допомогою якої можна додати новий продукт до бази даних.

Принцип побудови сторінок простий – спершу необхідно розробити дизайн, потім скласти «каркас» сторінки за допомогою мови html, наступним кроком необхідно додати стилів до проекту щоб він відповідав створеному раніше дизайну, після чого необхідно додавати функції до необхідних елементів, а останнім кроком йде дробіння об’ємних сторінок на більш маленькі та зручні частинки – компоненти.

Після того як створені сторінки (на рис. 3.15. представлено фрагмент коду сторінки з формою для додавання продуктів), потрібно налаштувати роутинг або навігацію по цим сторінкам, для цього створюємо 2 файли: «list.tsx» та «index.tsx», в першому файлі буде зберігатись 2 змінних: приватні посилання та публічні посилання, приватні будуть показані лише авторизованим користувачем, а публічні будуть доступні для не авторизованих користувачів (див. рис. 3.16). У файлі «index.tsx» буде описано логіку відображення приватних і публічних маршрутів, фрагмент коду наведено на рисунку 3.17.

```
return (
  <Box>
    <Typography variant="h3">Додати новий продукт:</Typography>
    <form
      onSubmit={formik.handleSubmit}
      style={{
        width: "60%",
        display: "flex",
        flexDirection: "column",
        gap: "15px",
      }}
    >
      <Box>
        <Typography>Назва продукту</Typography>
        <TextField
          fullWidth
          placeholder="name"
          type="text"
          onChange={formik.handleChange}
          id="name"
          helperText={formik.errors.name && formik.errors.name}
          value={formik.values.name}
        />
      </Box>
```

Рис. 3.15 Фрагмент коду сторінки з формою для додавання продуктів

```
export const PUBLIC_ROUTES: Route[] = [
  {
    path: "/",
    component: <OrderPage />,
  },
  {
    path: "/login",
    component: <Login />,
  },
];

export const PRIVATE_ROUTES: Route[] = [
  {
    path: "/table",

    component: (
      <AdminPanelLayout>
        <DataGridOuter />
      </AdminPanelLayout>
    ),
  },

  {
    path: "/reports",

    component: (
      <AdminPanelLayout>
        <Reports />
      </AdminPanelLayout>
    ),
  },
```

Рис. 3.16 Фрагмент коду файлу «list.tsx» з приватними та публічними шляхами

```tsx
export const Root: FC = () => {
  const { isUserLogined, isDataFetched } = useAuthContext();

  return (
    <>
      {isDataFetched ? (
        <>
          {isUserLogined ? (
            <Routes>
              {PRIVATE_ROUTES.map(({ path, component }) => (
                <Route path={path} element={component} key={`Route-${path}`} />
              ))}
              <Route path="*" element={<PageNotFound />} />
            </Routes>
          ) : (
            <Routes>
              {PUBLIC_ROUTES.map(({ path, component }) => (
                <Route path={path} element={component} key={`Route-${path}`} />
              ))}
              <Route path="*" element={<PageNotFound />} />
            </Routes>
          )}
        </>
      ) : (
        <LinearProgress />
      )}
    </>
  );
};
```

Рис. 3.17 Фрагмент коду файлу «index.tsx» з логікою відображення шляхів

Сторінки на яких є форми для заповнення повинні мати валідацію даних, для того щоб запобігати помилковому введенню даних та для формування дружнього інтерфейсу користувача. Саме для таких випадків була обрана бібліотека Yup, яка надає можливість створювати схеми для валідації форм, схему наведено на рисунку 3.18, а на рисунку 3.19 наведено результат роботи цієї схеми.

```tsx
import * as Yup from "yup";  40.7k (gzipped: 12.6k)

export const validationSchema = Yup.object({
  description: Yup.string()
    .min(5, "Опис зандто короткий")
    .max(100, "Опис зандто довгий")
    .required("Обов'язкове поле"),
  name: Yup.string()
    .min(5, "Назва зандто коротка")
    .required("Обов'язкове поле"),
  price: Yup.number().required("Обов'язкове поле"),
});
```

Рис. 3.18 фрагмент коду валідації сторінки авторизації

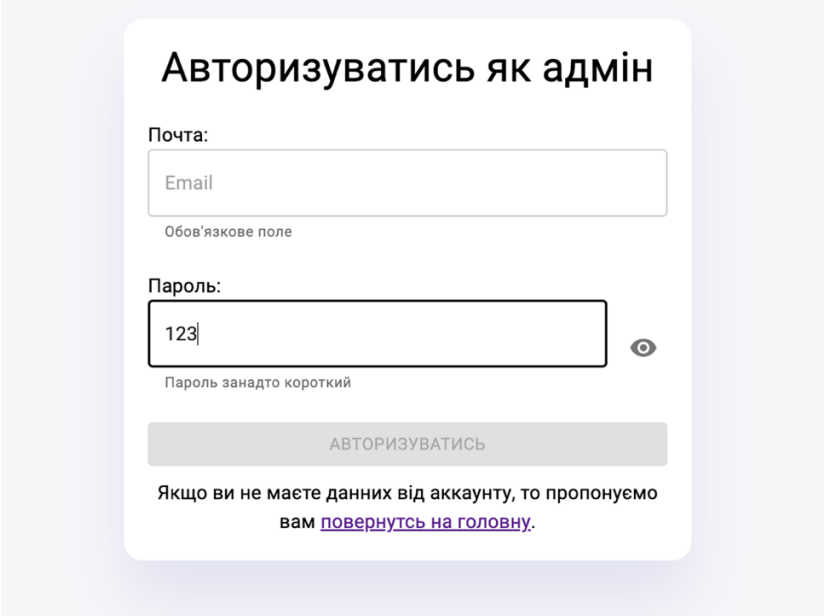


Рис. 3.19 Результат роботи валідації

3.3. Інструкція користувача

Цей додаток можуть використовувати як працівники підприємства, для моніторингу статистики продажів, додавання нових товарів, редагування або видалення товарів, так і підприємці, для замовлення товару для реалізації.

Після того як користувачі потрапляють на головну сторінку сайту, вони можуть ознайомитись з коротким описом підприємства, також на сторінці наведений слайдер із зображеннями та форма для замовлення продукції, що зображені на рис. 3.20.

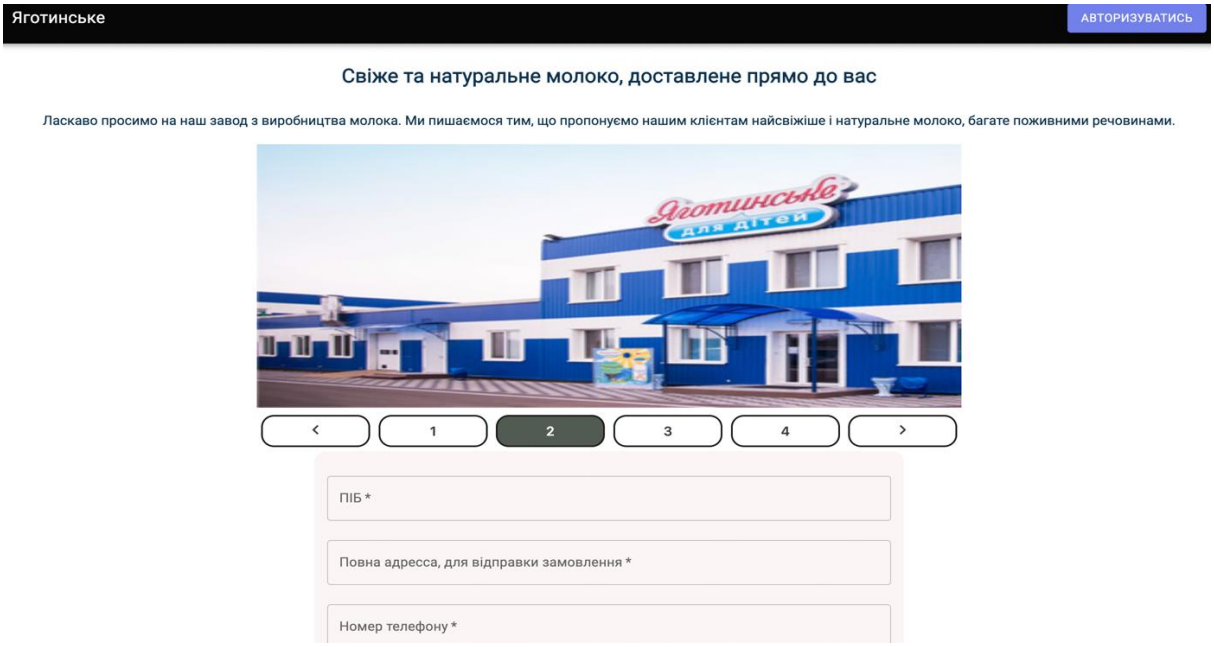


Рис. 3.20 Головна сторінка

На цьому етапі замовник може зробити замовлення товару, через заповнення необхідної форми, де він має вказати: своє ім’я, адресу на яку відправляти замовлення, свій номер телефону та вибрати товари у необхідній кількості.

Працівники підприємства мають логін та пароль, за допомогою якого вони можуть авторизуватись у системі, для цього їм потрібно на головній сторінці натиснути кнопку «*Авторизуватись*» та ввести необхідні дані у форму. Сторінка авторизації зображена на рис 3.21.

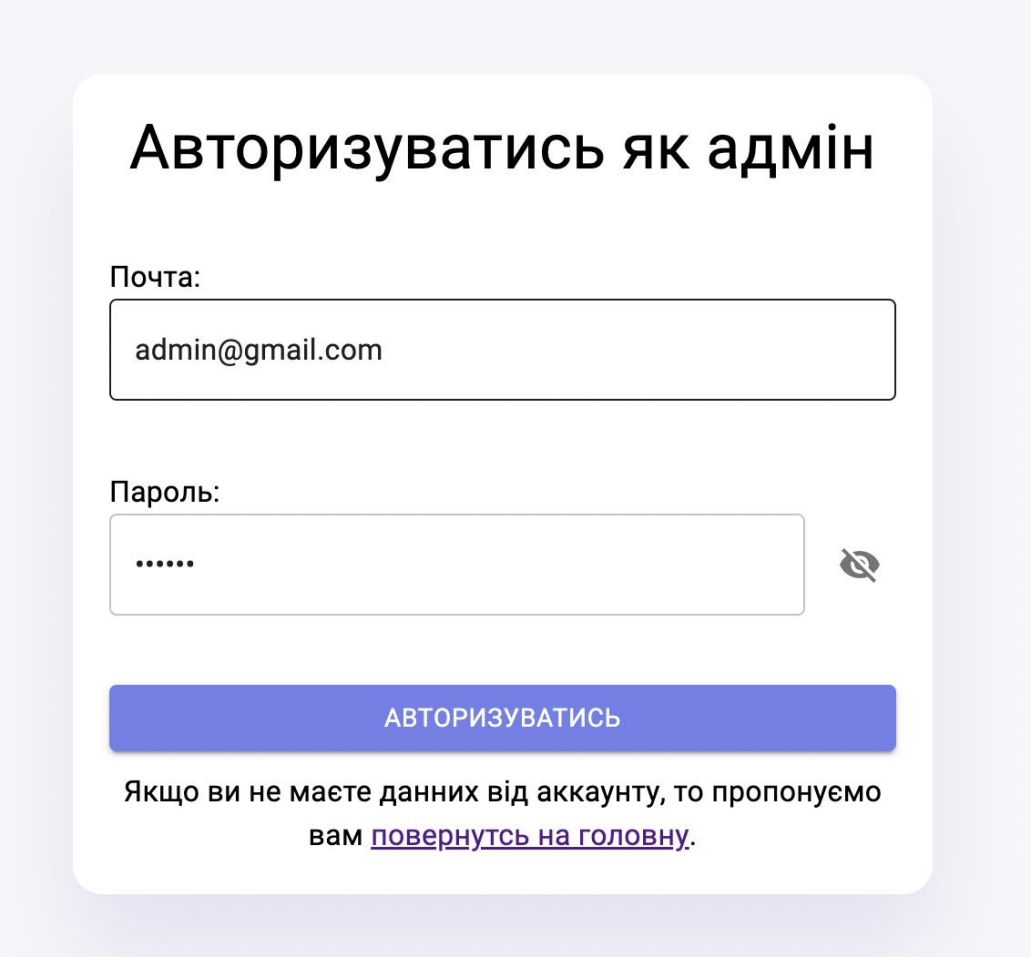


Рис 3.21 Сторінка авторизації

Авторизувавшись працівник потрапляє до панелі управління замовленнями. Перша сторінка, яка зустріне користувача – це сторінка з усім асортиментом продукції підприємства. Сторінка наведена на рис 3.22.

Біля кожного продукту є синій маркер, натиснувши на який користувач зможе відредагувати інформацію про продукт, а натиснувши на червону урну працівник зможе видалити продукт.

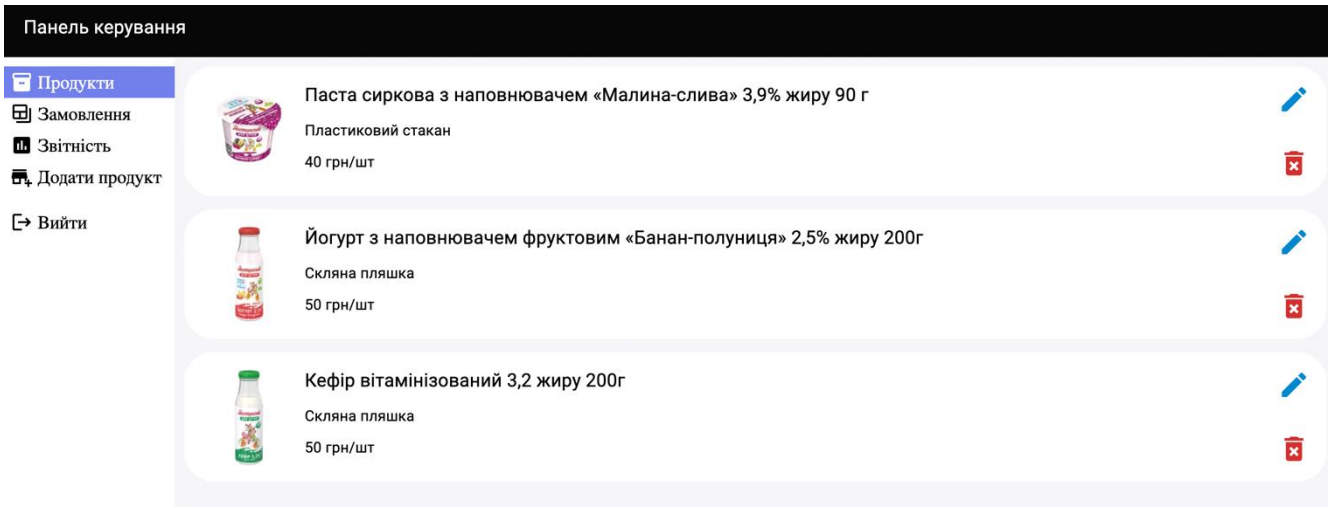


Рис 3.22 Головна сторінка панелі керування, на якій зображені продукти

Обравши наступний пункт у панелі навігації, а саме «замовлення» працівники опиняться на сторінці, де відображені усі замовлення компанії, починаючи з останнього замовлення. В цій таблиці вони можуть фільтрувати данні, приховувати чи відображати необхідні стовбці, відфільтрувати таблицю чи роздрукувати її, або зберегти у форматі «CSV», ця сторінка наведена на рисунку 3.23. Також натиснувши на кнопку «подробиці» у стовпці «відкрити подробиці замовника», працівник побачить вкладену таблицю у якій наведено всі товари та деталі замовлення, дану таблицю зображено на рисунку 3.24.

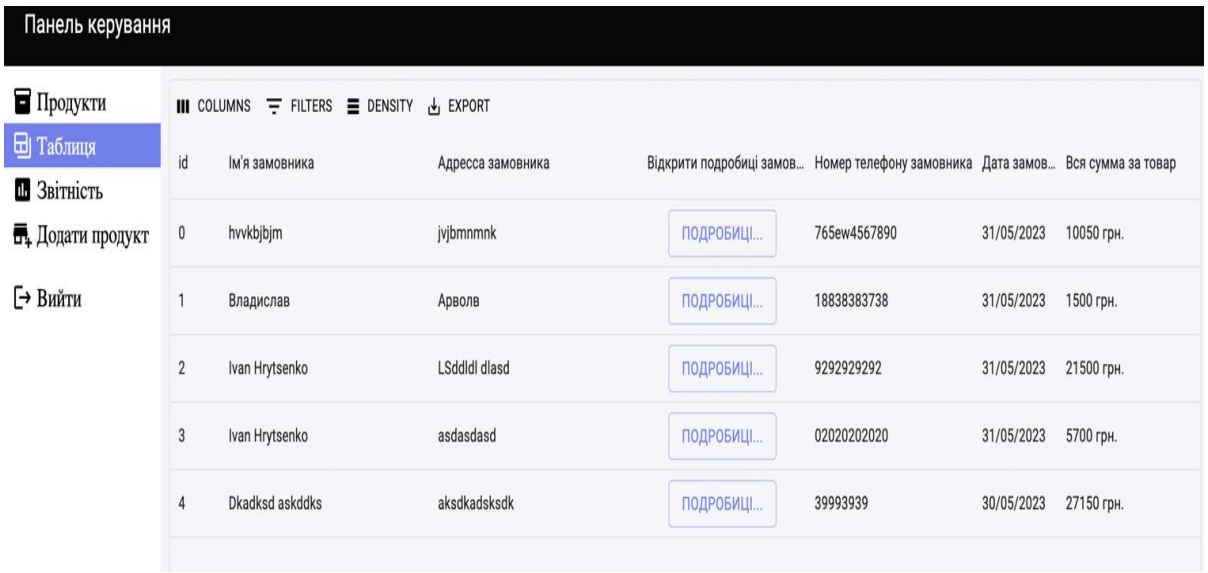


Рис.3.23. Сторінка з таблицею всіх замовленню.

id	Назва продукту	Кількість (шт.)	Ціна продукту (за шт.)	Ціна за всю кількість товару
0	Паста сиркова з наповн...	100 шт.	40 грн.	4000 грн.
1	Кефір вітамінізований 3,...	120 шт.	50 грн.	6000 грн.
2	Йогурт з наповнювачем ...	90 шт.	50 грн.	4500 грн.
3	Йогурт з наповнювачем ...	140 шт.	50 грн.	7000 грн.

Rows per page: 100 1–4 of 4

Рис.3.24. Таблиця з подробицями замовлення.

Відкривши сторінку – «звітність» працівник побачить форму де він має заповнити дві дати: дату кінця та дату початку вибірки та назву товару, після чого отримає таку статистику:

- кількість замовлених товарів за визначений проміжок днів та їх відображення у вигляді діаграми;
- загальну суму, отриману за замовлені товари та матиме можливість розрахувати відсоток виконання планових надходжень;
- обсяги продажу та отриманих прибутків по вибраному товару за обраний період

Сторінка без заповнення часового проміжку зображена на рисунку 3.25, а сторінка з вказаним проміжком дат та статистикою зображена на рисунку 3.26

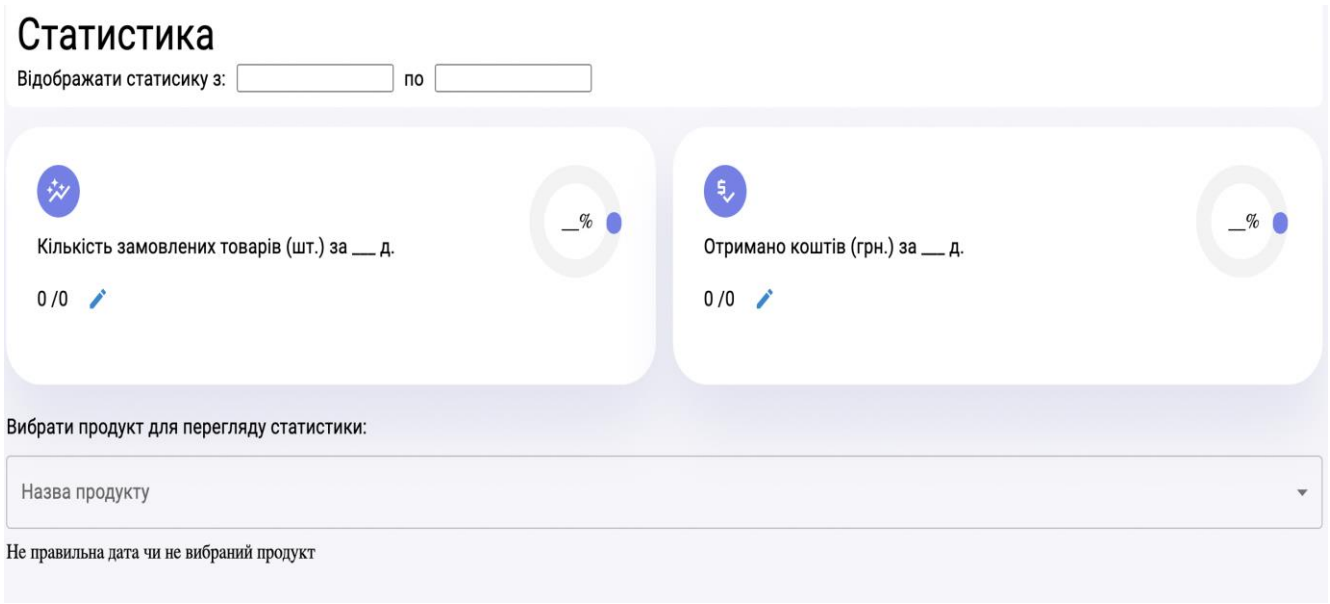


Рис.3.25. Сторінка звітності без заповнення часового проміжку

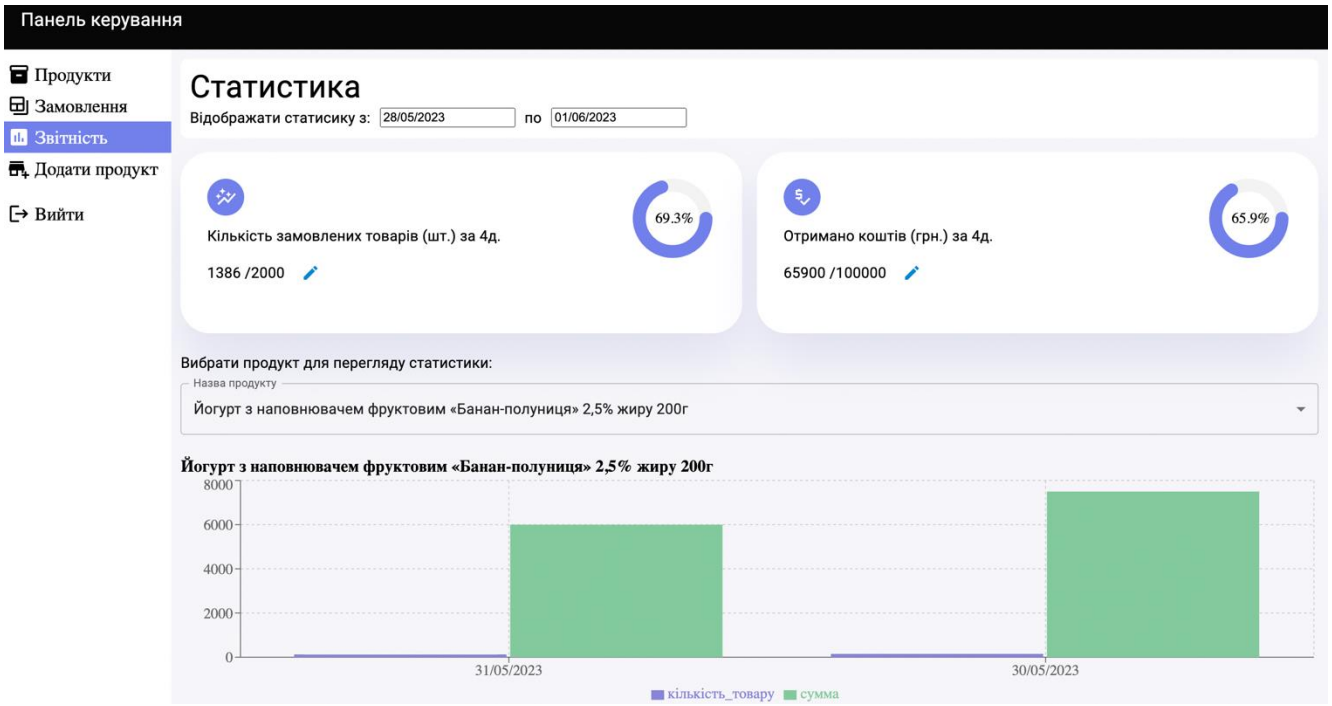


Рис.3.26. Сторінка звітності з вказаним проміжком дат та статистикою.

У вкладці «додати продукт» наявна форма для додавання товарів до бази даних (див. рис. 3.27).

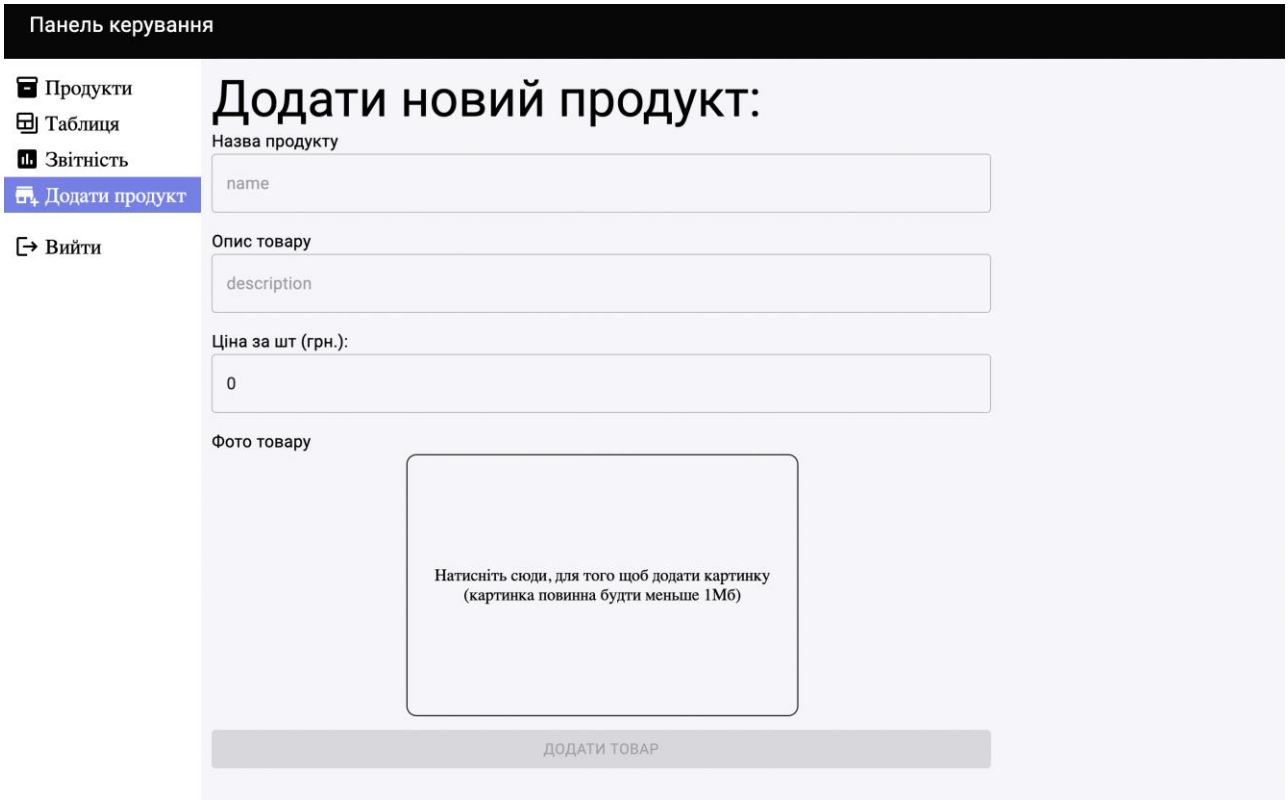


Рис.3.27. Сторінка для введення продукції

Останній елемент на панелі навігації – це «вийти», він призначений для виходу з аккаунту на головну сторінку сайту.

3.4. Технічне та системне забезпечення розробки

3.4.1. Обґрунтування вибору технічних засобів

- JavaScript є однією з найбільш популярних мов програмування, особливо в контексті веб-розробки. Ця мова має широку спільноту розробників, розгалужену екосистему та безліч інструментів для розробки. Використання JavaScript дозволяє створювати динамічні та інтерактивні веб-додатки, які задовольняють сучасні вимоги ринку[24].
- React - це одна з найпопулярніших та передових бібліотек для розробки інтерфейсів, особливо для веб-додатків. Вона має велику спільноту розробників, що постійно активна, широку екосистему та потужні інструменти розробки. Використання React дозволяє швидко створювати гнучкі, швидкодіючі та повторно використовувані компоненти, сприяючи швидкій розробці та покращенню користувацького досвіду [19].
- Firebase - це потужна та інноваційна платформа для розробки збереження та обробки даних в хмарі. Вона має активну та зростаючу спільноту розробників, багатий набір функціональностей та зручні інструменти розробки. Використання Firebase дозволяє легко зберігати та синхронізувати дані, реалізовувати аутентифікацію користувачів та забезпечувати високу швидкість та стабільність додатків.
- Material UI: Material UI - це прекрасна бібліотека для розробки інтерфейсів, яка пропонує елегантні та сучасні компоненти. Вона має активну спільноту розробників, яка постійно вносить внесок у розвиток цієї бібліотеки. Material UI забезпечує швидку розробку та можливість створювати стильні та привабливі інтерфейси, використовуючи концепцію Material Design від Google[20].
- Visual Studio Code є популярним та потужним редактором коду, який має широкий вибір розширень та плагінів. Цей редактор надає зручне та

ефективне середовище розробки, що включає багатофункціональний редактор, підсвічування синтаксису, автодоповнення, можливість налагодження та інші корисні функції [25].

3.4.2. Розробка і обґрунтування стратегії адміністрування системи

Адміністрування системи та розробка стратегії адміністрування є необхідною частиною процесу розробки додатку для замовлення продукції Яготинського маслозаводу.

Основні пункти, які були враховані:

- Авторизація – дозволяє зайти працівнику у систему за допомогою власного аккаунту, який підприємство надає йому;
- Збереження файлів – веб додаток надає можливість зберігати інформацію у форматі «CVS» або за допомогою друку.

3.4.3. Заходи захисту від несанкціонованого доступу до системи

Веб-додаток для замовлення продукції Яготинського маслозаводу має такі заходи захисту безпеки від несанкціонованого доступу:

- Захист сторінки протоколом https;
- Авторизація працівника, перед тим як потрапити до панелі керування;
- Можливість завантажити або роздрукувати данні
- Приховані данні для підключення до бази даних

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

Охорона праці є невід'ємною частиною будь-якої діяльності, де присутня робота людей, включаючи інформаційну сферу. Розробка та впровадження веб-додатку для моніторингу вподобання споживачів маслозаводу з використанням комп’ютерної техніки, що означає необхідність приділяти особливу увагу питанням охорони праці працівників, зайнятих цими процесами.

Охорона праці в контексті розробки та використання веб-додатка означає забезпечення безпеки та здоров'я працівників на робочому місці. Для досягнення цієї мети необхідно вживати широкий спектр заходів, спрямованих на запобігання ризикам та негативним наслідкам, пов’язаним із роботою з комп’ютерною технікою.

Основною метою охорони праці є забезпечення здорових та безпечних умов праці працівників на підприємстві, які має створити керівник відповідно до чинного законодавства України. Необхідною умовою для працівників на підприємстві щодо забезпечення безпеки всіх працівників є навчання правилам безпеки життєдіяльності на виробництві та обов’язковий контроль їх виконання. Безпосередній контроль умов виконання правил з охорони праці працівниками покладається на начальника відділу охорони праці Яготинського маслозавода.

Система законодавства України про охорону праці включає в себе набір взаємопов'язаних нормативно-правових актів, які регулюють відносини у сфері виконання державної політики щодо правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини під час праці. Ця система включає такі закони, як Закон України «Про охорону праці», Кодекс законів про працю України, Закон України «Про загальнообов’язкове державне соціальне страхування від нещасних випадків на виробництві та професійних захворюваннях, які спричинили втрату працездатності», а також інші нормативно-правові акти, що приймаються на підставі цих законів.

В статті 43 Конституції України зазначено:

Кожен має право на працю, що включає можливість заробляти собі на життя працею, яку він вільно обирає або на яку вільно погоджується.

Держава створює умови для повного здійснення громадянами права на працю, гарантує рівні можливості у виборі професії та роду трудової діяльності, реалізовує програми професійно-технічного навчання, підготовки і перепідготовки кадрів відповідно до суспільних потреб. Використання примусової праці забороняється. Не вважається примусовою працею військова або альтернативна (невійськова) служба, а також робота чи служба, яка виконується особою за вироком чи іншим рішенням суду або відповідно до законів про воєнний і про надзвичайний стан. Кожен має право на належні, безпечні і здорові умови праці, на заробітну плату, не нижчу від визначеної законом. Використання праці жінок і неповнолітніх на небезпечних для їхнього здоров'я роботах забороняється. Громадянам гарантується захист від незаконного звільнення. Право на своєчасне одержання винагороди за працю захищається законом [26].

Закон України «Про охорону праці» визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров’я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в України [27].

Державна політика в галузі охорони праці базується на принципах:

- пріоритету життя і здоров'я працівників, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці;
- підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;
- комплексного розв'язання завдань охорони праці на основі загальнодержавної, галузевих, регіональних програм з цього питання та з урахуванням інших напрямів економічної і соціальної політики, досягнень в галузі науки і техніки та охорони довкілля;

- соціального захисту працівників, повного відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;
- встановлення єдиних вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності;
- адаптації трудових процесів до можливостей працівника з урахуванням його здоров'я та психологічного стану;
- забезпечення координації діяльності органів державної влади, установ, організацій, об'єднань громадян, що розв'язують проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництва і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці на місцевому та державному рівнях;
- використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародного співробітництва.

Управління охороною праці є надзвичайно важливою складовою діяльністю будь-якого підприємства. Його основна мета забезпечує у забезпеченні безпеки, здоров'я та добробуту працівників, а також у запобіганні можливих негативних наслідків, які можуть виникнути в результаті проведення роботи.

Охорона праці включає широкий спектр заходів, які повинні бути вжиті на підприємствах. Один з окремих аспектів - це регулярні перевірки технічного стану обладнання та інвентарю. Це дозволяє виявити дефекти, несправності або зношені деталі, які можуть стати загрозою для безпеки працівників. Результати перевірки дозволили зробити ремонт або заміну несправних елементів, забезпечуючи безпечні умови праці.

Проведення навчання з питань безпеки та здоров'я праці є ще одним аспектом охорони праці на підприємствах. Це може включати навчання працівників правилам безпеки, правильному використанню обладнання та інструментів,

процедурам реагування на надзвичайній ситуації та іншим аспектам, які допомагають зменшити ризики для здоров’я та безпеки.

Розробка та виконання планів екстрених ситуацій також входять до складу управління охороною праці. Ці плани розпочати процедуру евакуації у разі виникнення пожежі, аварії чи інших небезпечних ситуацій. Вони включають інструкції щодо поведінки працівників, виклику більшості служб та вживання заходів безпеки для мінімізації ефекту пошкоджень та травм.

Системні медичні огляди працівників на виявленні виявлені захворювання або стани, які можуть бути спричинені робочими умовами. Це включає перевірку фізичного здоров'я, діагностику можливих професійних захворювань та рекомендації щодо запобігання подальшого пошкодження здоров'я.

Крім вищезгаданих заходів, управління охороною праці також включає:

- встановлення та забезпечення дотримання вимог нормативно-правових актів з охорони праці. Роботодавець повинен ознайомити працівників з правилами та інструкціями з безпеки праці та забезпечити їх виконання;
- організація і проведення аналізу ризиків на робочих місцях. Це включає ідентифікацію наявних небезпек, оцінку ризиків, розробку та впровадження заходів з мінімізації ризиків. Такий аналіз дозволяє варіанти альтернативи усунення небезпечних умов праці та забезпечити безпеку працівників;
- забезпечення належного обладнання та оснащення робочих місць. Роботодавець повинен забезпечити працівників необхідними засобами індивідуального захисту, а також правильно налаштованим та безпечним обладнанням. Це може включати захисні екрани, вентиляційні системи, сигналізацію небезпеки та інші засоби, що сприяють зниженню ризиків для здоров'я та безпеки працівників.

ВИСНОВКИ

В кваліфікаційній роботі розроблено web-додаток для замовлення продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод».

В процесі виконання цієї роботи було закріплено всі знання та навички, які були отримані під час навчання, а саме з програмування, системного аналізу та проектування.

Під час проведення системного аналізу було проведено загальний огляд підприємства, оцінено рівень комп'ютеризації та виявлені проблемні аспекти. Запропоновано рішення, яке полягає у створенні веб-додатку. Проведено порівняння цього рішення з існуючими продуктами, обговорено доцільність його розробки та розраховано очікуваний економічний ефект від впровадження системи. Особлива увага була приділена складанню технічного завдання, включаючи вимоги до функціональності додатку, терміни розробки, передачі та гарантійної підтримки.

На наступному етапі складено технічне завдання в якому зазначено всі вимоги до додатку, термінів розробки, вимог до гарантійної підтримки та інших деталей.

Згідно вимог технічного завдання розроблено web-додаток для замовлення продукції підприємства ТДВ «Яготинський маслозавод» та збереження інформації про замовлені товари для подальшої їх передачі у відповідні відділи підприємства. При розробці додатку були задіяні різноманітні бібліотеки, такі як: React, Firebase, Formik, Yup, Material UI та інші.

Після завершення розробки було створено інструкцію для користувачів додатку, де описано основні функції та способи взаємодії з ним.

Отже, у кваліфікаційній роботі здійснена розробка web-додатку для оперативного замовлення продукції ТМ «Яготинське для дітей» ТДВ «Яготинський маслозавод», який підвищить ефективність роботи з замовниками продукції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний веб-сайт «Молочного Альянсу». [Електронний ресурс]. URL: https://milkalliance.com.ua/en/company/enterprises/yagotinskij-maslozavod/ (дата звернення 20.04.2023)

2. Звіт про управління ТДВ «Яготинський маслозавод» за 2020. [Електронний ресурс]. URL: https://milkalliance.com.ua/tools/cms/site/download.php?url=/uploads/site_factory_docs/file/0008/34.pdf&name=zvit-pro-upravlinnya-za-2020-rik (дата звернення 20.04.2023)

3. М'якшило, О. М. CASE-технології у проектуванні інформаційних систем [Електронний ресурс] [Текст] : навч. посіб. / О. М. М'якшило, Л. Г. Загоровська. — Київ : НУХТ, 2017. — 190 с. — каф. інформаційних систем.

4. Огляд продукту Microsoft access. [Електронний ресурс]. URL: https://www.microsoft.com/en-us/microsoft-365/access (дата звернення 20.04.2023)

5. Огляд продукту Microsoft excel. [Електронний ресурс]. URL: https://www.microsoft.com/en-in/microsoft-365/excel (дата звернення 20.04.2023)

6. Огляд продукту google sheets. [Електронний ресурс]. URL: https://corporatefinanceinstitute.com/resources/excel/google-sheets/ (дата звернення 20.04.2023)

7. Управління ІТ проектами [Електронний ресурс]: методичні рекомендації до самостійної роботи для здобувачів освітнього ступеня «Бакалавр» спеціальності 122 «Комп'ютерні науки» освітньо-професійних програм «Комп'ютерні науки» та «Інформаційні системи та штучний інтелект» денної та заочної форм навч. / уклад. С. В. Грибков, О. Л. Сєдих – К.: НУХТ, 2022 – 27 с.

8. Методичні рекомендації до виконання випускної кваліфікаційної роботи на здобуття освітнього ступеня «Бакалавр» спеціальності 122 «Комп'ютерні науки» освітньо–професійної програми «Комп'ютерні науки» денної та заочної форм навчання [Електрон. ресурс] / уклад. О.М. М'якшило, М.П. Костіков. – К. : НУХТ, 2022 – 34 с.

9. ДСТУ 3974-2000. Система розроблення та поставлення продукції на виробництво.

10. ДСТУ 3321-2003. Системи конструкторської документації. Терміни та визначення основних понять.

11. ДСТУ 1.0:2003. СТУ 1.0:2003. Національна стандартизація. Основні положення

12. ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;

13. ДСТУ 3973–2000 Система розроблення та поставлення продукції на виробництво.

14. Огляд документації бібліотеки Firebase. [Електронний ресурс]. URL: https://firebase.google.com/ (дата звернення 20.04.2023)

15. Огляд документації бібліотеки Cloud Firestore. [Електронний ресурс]. URL: https://firebase.google.com/docs/firestore?hl=en (дата звернення 20.04.2023)

16. Огляд документації бібліотеки Firebase Auth. [Електронний ресурс]. URL: https://firebase.google.com/docs/auth?hl=en (дата звернення 20.04.2023)

17. Огляд документації бібліотеки Redux. [Електронний ресурс]. URL: https://redux.js.org/introduction/getting-started (дата звернення 20.04.2023)

18. Огляд документації бібліотеки MobX. [Електронний ресурс]. URL: https://mobx.js.org/README.html (дата звернення 20.04.2023)

19. Огляд бібліотеки React. [Електронний ресурс]. URL: https://react.dev/ (дата звернення 20.04.2023)

20. Огляд бібліотеки Material UI. [Електронний ресурс]. URL: https://mui.com/material-ui/getting-started/overview/ (дата звернення 20.04.2023)

21. Огляд документації бібліотеки Formik. [Електронний ресурс]. URL: https://formik.org/docs/overview (дата звернення 20.04.2023)

22. Огляд документації бібліотеки Yup. [Електронний ресурс]. URL: https://www.npmjs.com/package/yup (дата звернення 20.04.2023)

23. Огляд документації бібліотеки React-router-dom. [Електронний ресурс]. URL: https://reactrouter.com/en/main/start/tutorial (дата звернення 20.04.2023)

24. Огляд мови програмування JavaScript. [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 20.04.2023)

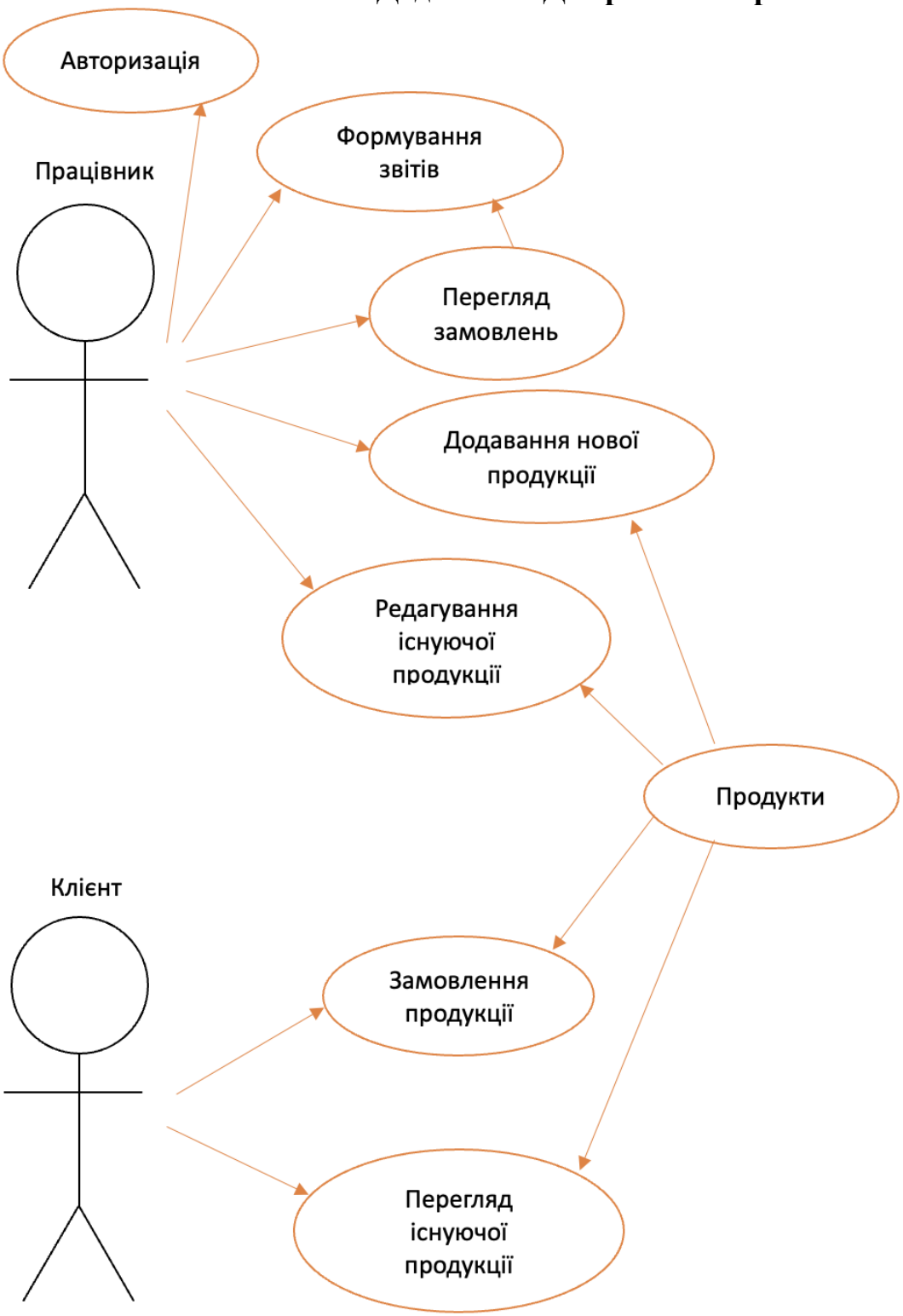
25. Огляд програми для розробки програмного забезпечння VS code [Електронний ресурс]. URL: https://code.visualstudio.com/doc (дата звернення 20.04.2023)

26. Конституція України - Розділ 2 Конституції України, стаття 43.

27. Закон України «Про охорону праці», документ 2694-XII

ДОДАТКИ

Додаток А. Діаграма використання системи



Додаток Б. Інтерфейс веб-додатку

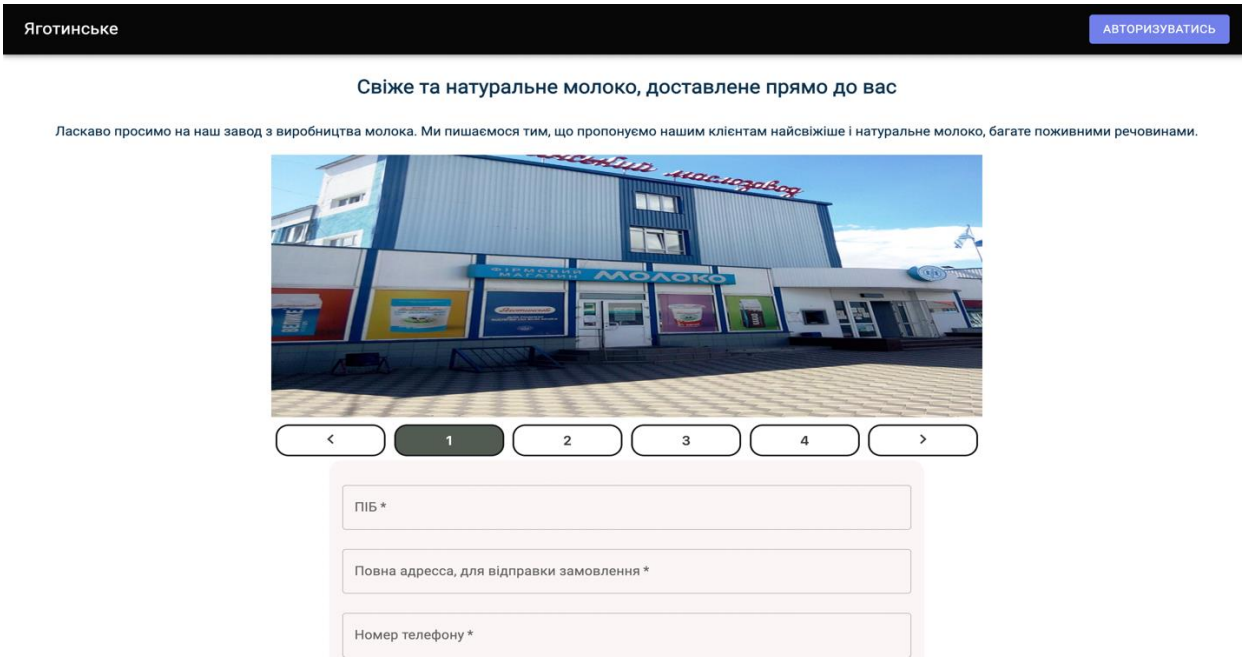


Рис.Б.1. Головна сторінка веб-додатку

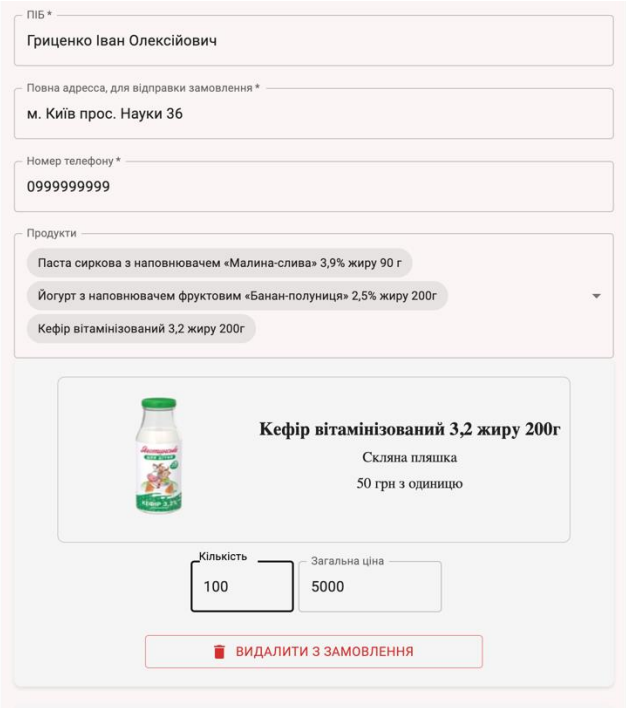


Рис.Б.2. Головна сторінка веб-додатку з заповненою формою замовлення

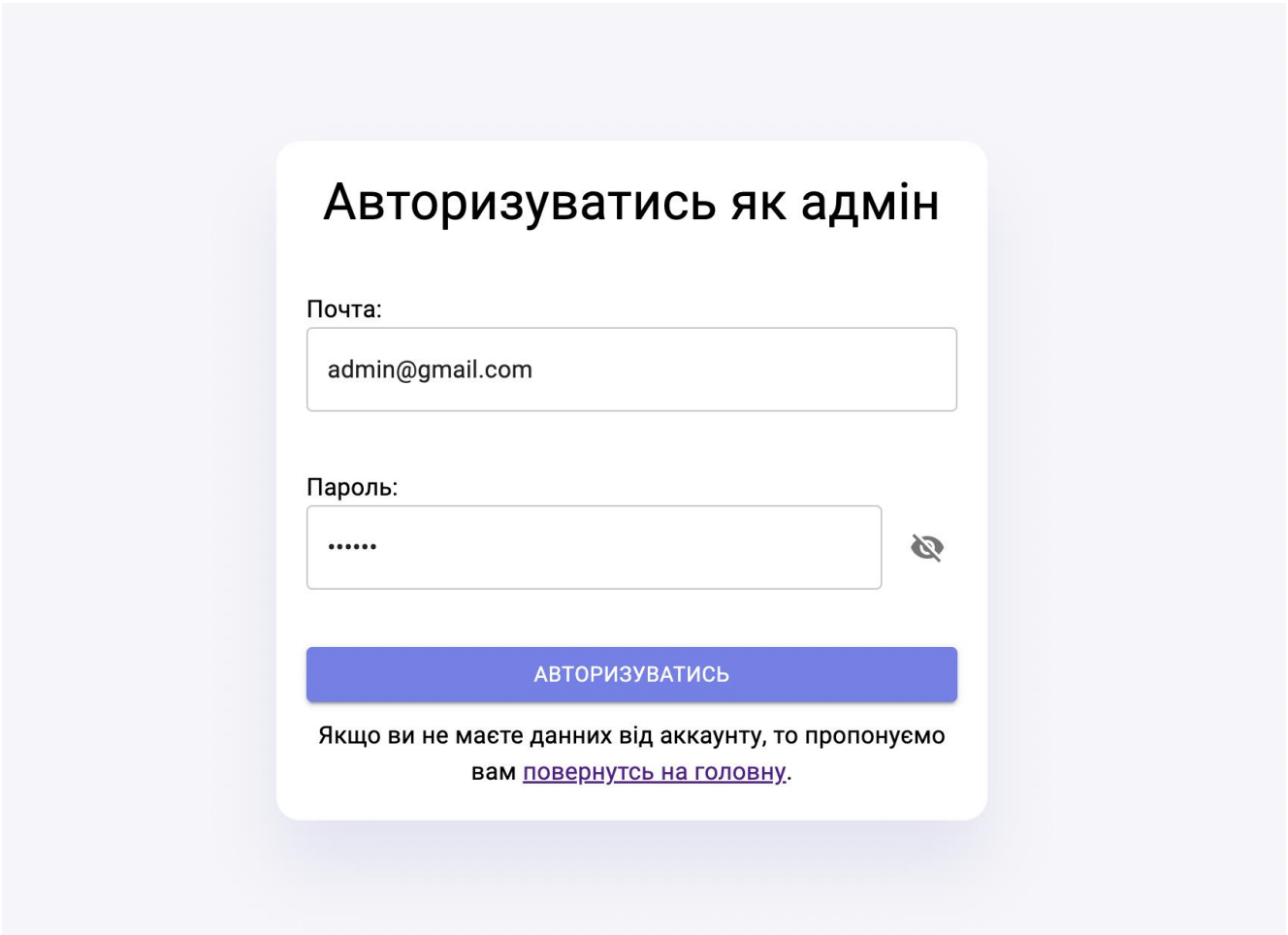


Рис.Б.3. Сторінка авторизації

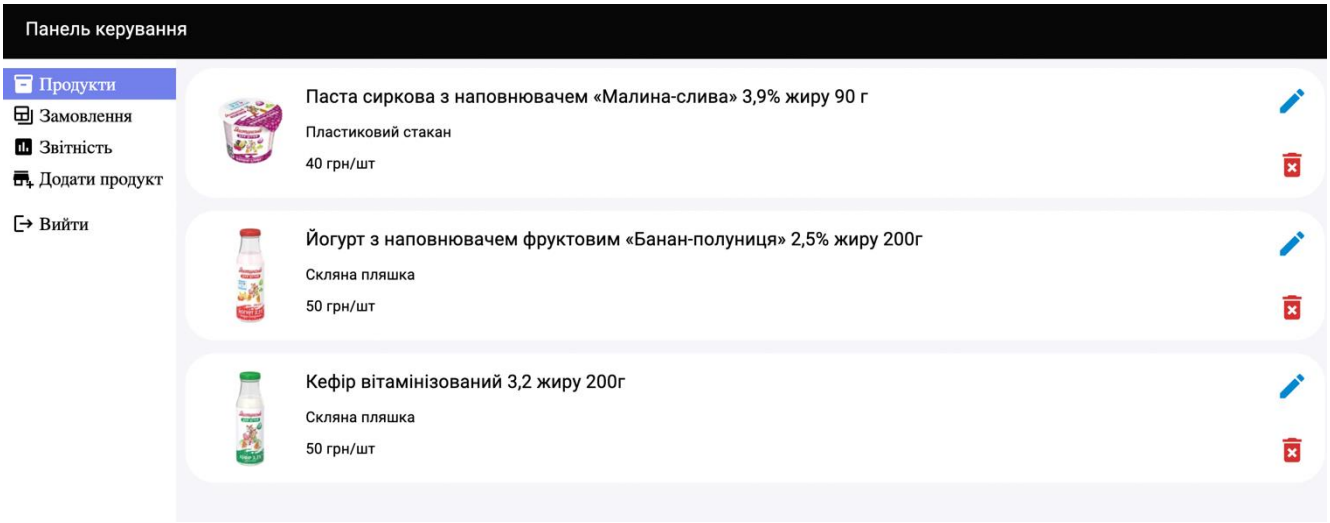


Рис.Б.4. Сторінка адмін-панелі з продуктами підприємства

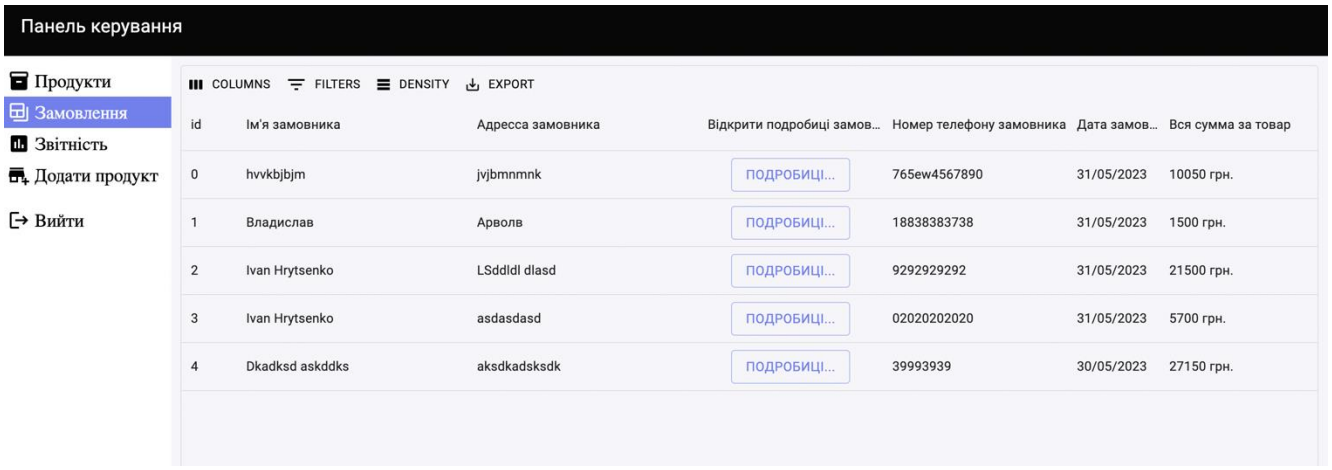


Рис.Б.5. Сторінка адмін-панелі з переліком замовників

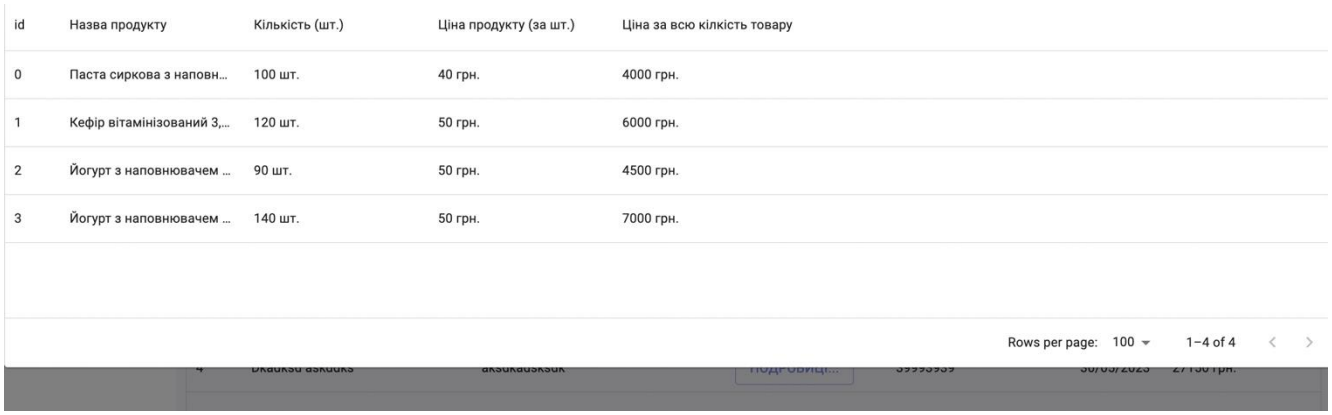


Рис.Б.6. Сторінка адмін-панелі з табличкою замовлених продуктів певним користувачем

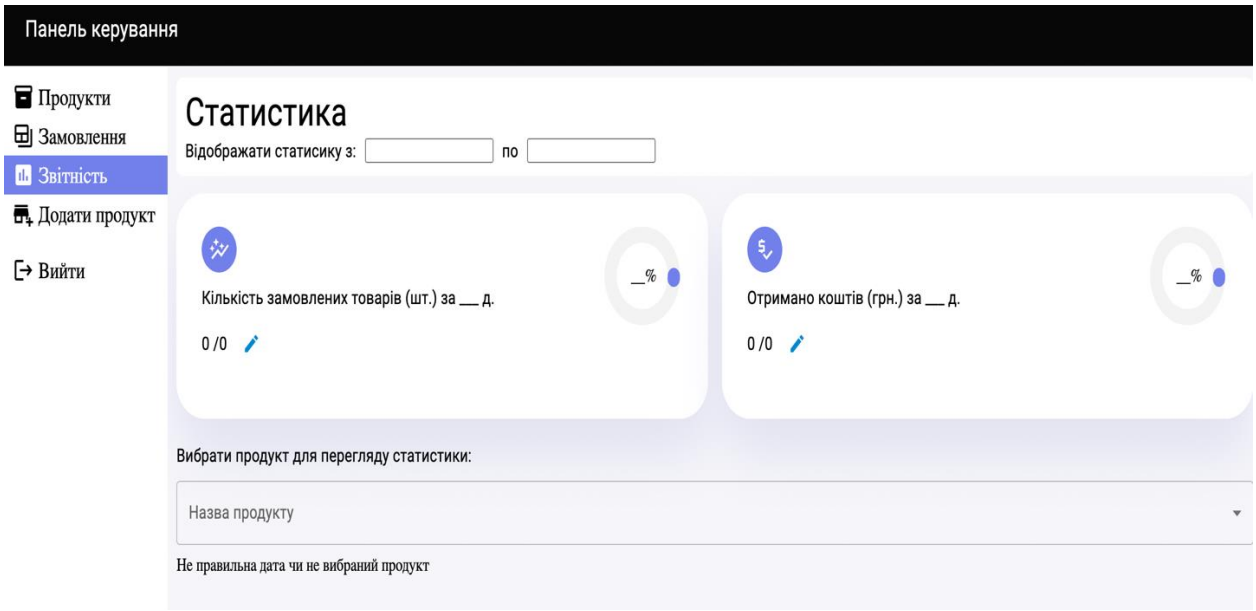


Рис.Б.7. Сторінка адмін-панелі для відображення статистики

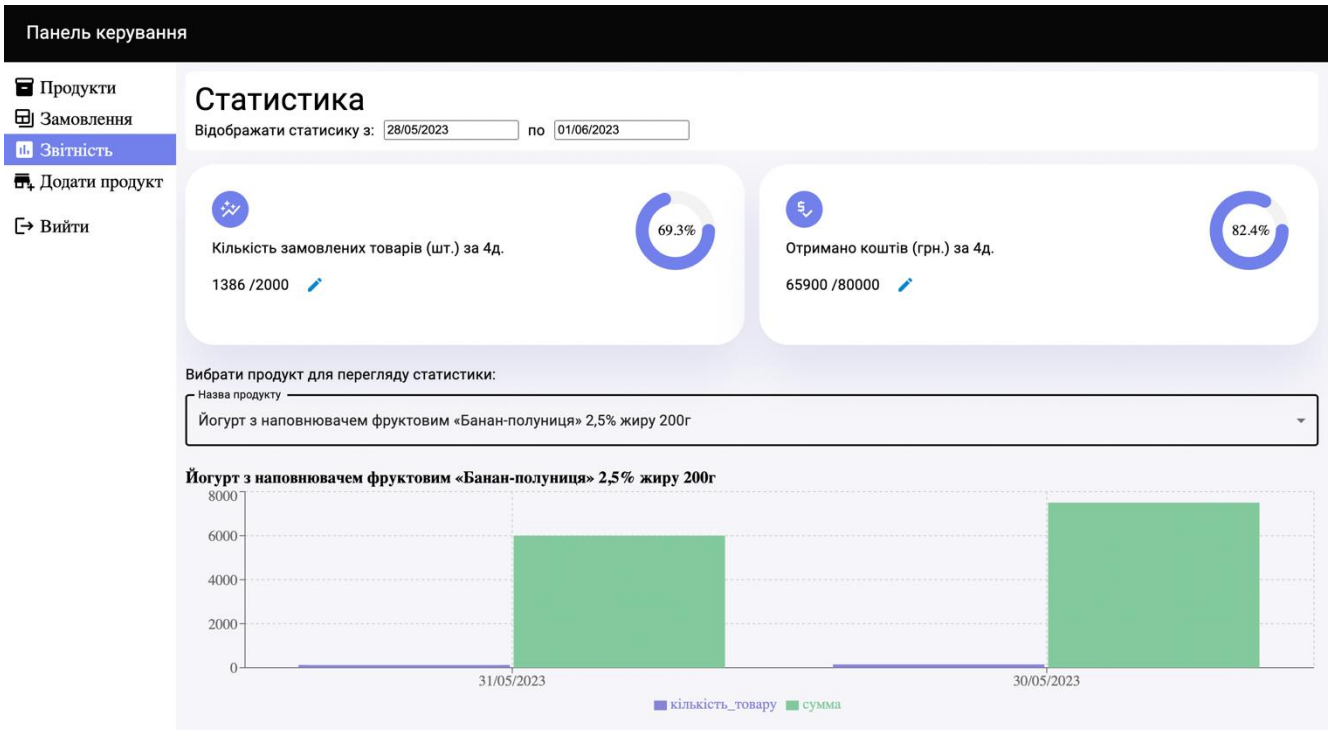


Рис. Б.8. Сторінка адмін-панелі з відображення статистики за обраний період

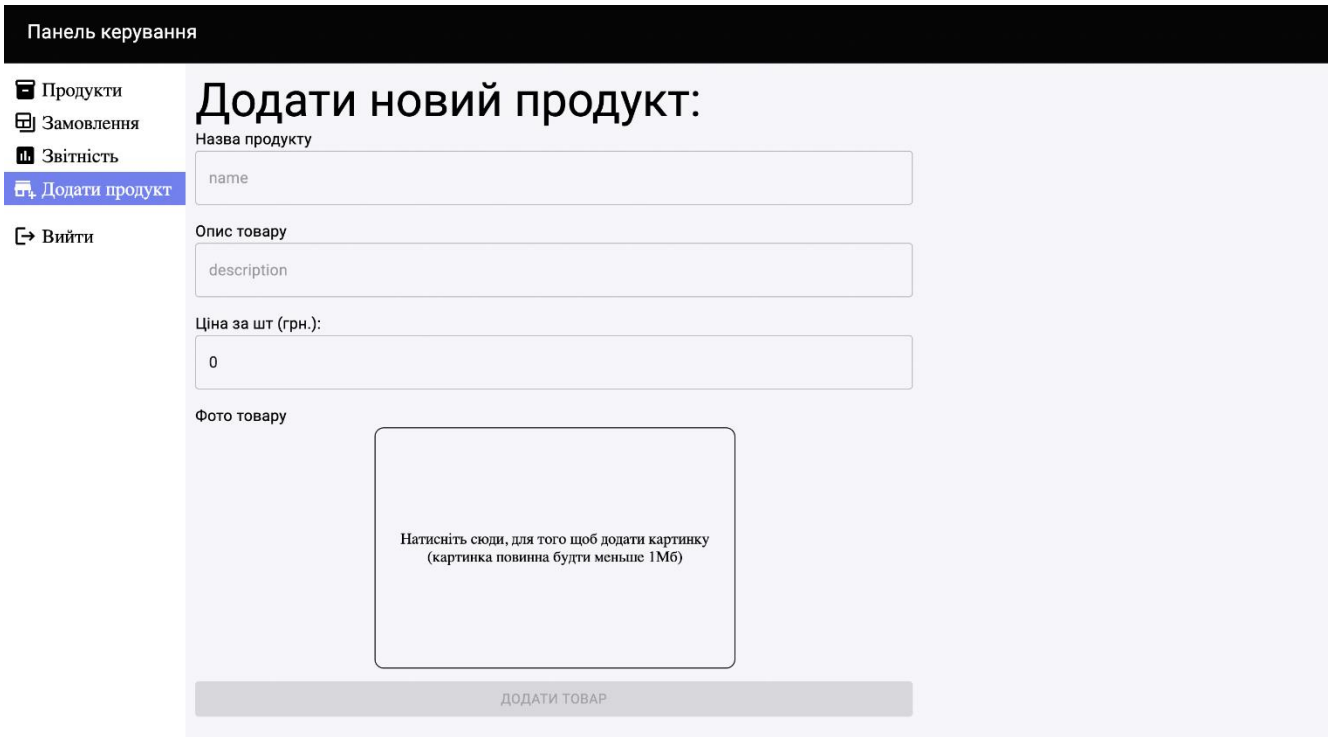
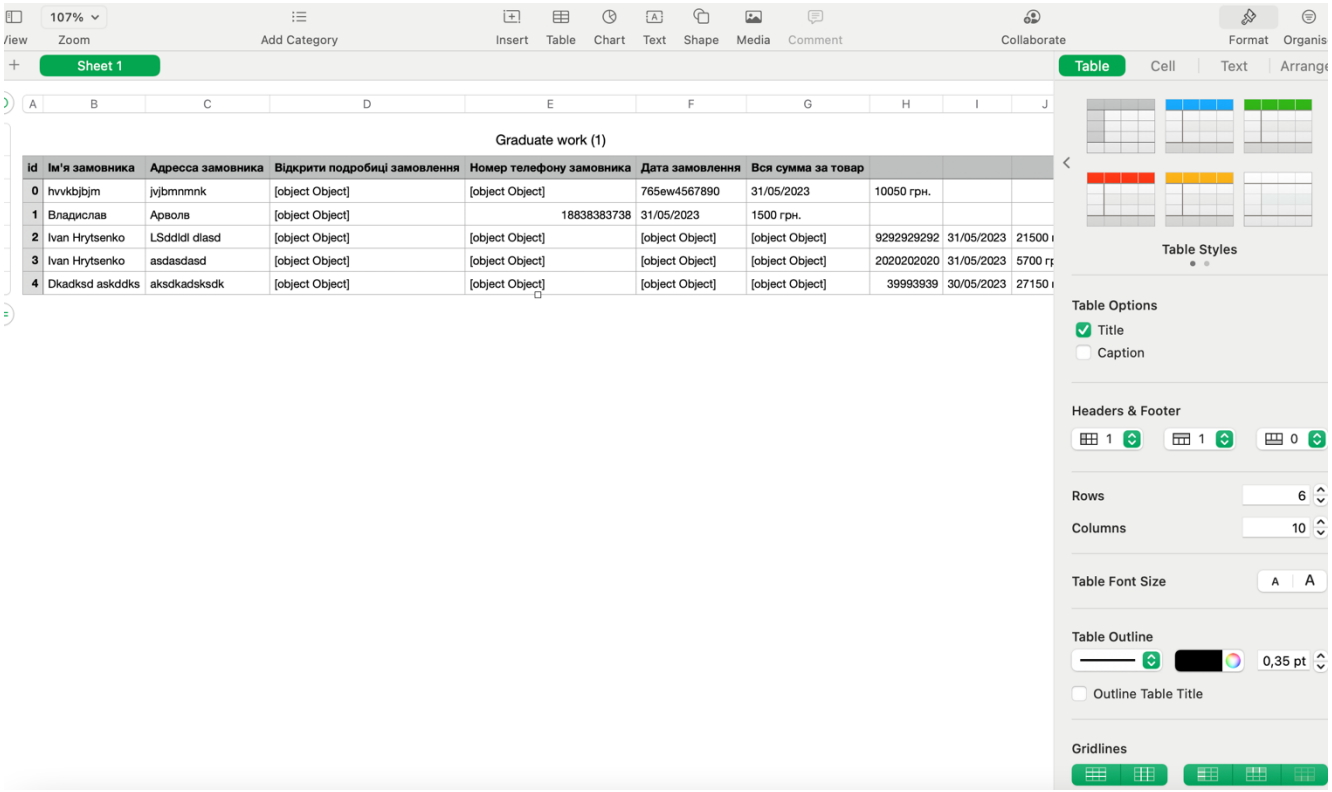


Рис.Б.9. Сторінка адмін-панелі на якій можна додати новий продукт до бази даних



id	Ім'я замовника	Адресса замовника	Відкрити подробиці замовлення	Номер телефону замовника	Дата замовлення	Вся сумма за товар			
0	hvvkbjbjm	jvjbmnmnk	[object Object]	[object Object]	765ew4567890	31/05/2023	10050 грн.		
1	Владислав	Арволв	[object Object]	18838383738	31/05/2023	1500 грн.			
2	Ivan Hrytsenko	LSddldl dlasd	[object Object]	[object Object]	[object Object]	[object Object]	9292929292	31/05/2023	21500 г
3	Ivan Hrytsenko	asdasdasd	[object Object]	[object Object]	[object Object]	[object Object]	2020202020	31/05/2023	5700 гр
4	Dkadksd askddks	aksdkadsksdk	[object Object]	[object Object]	[object Object]	[object Object]	39993939	30/05/2023	27150 г

Рис.Б.10. Експортований список замовників у форматі .CSV

Додаток В. Програмний код програми

1. Налаштування стилів проекту

```
import { createTheme } from "@mui/material/styles";

declare module "@mui/material/styles" {
  interface Palette {
    customBackground: Palette["primary"];
    customCardColor: Palette["primary"];
  }
  interface PaletteOptions {
    customBackground?: PaletteOptions["primary"];
    customCardColor?: PaletteOptions["primary"];
  }
}

const theme = createTheme({
  palette: {
    primary: {
      main: "#070707",
    },
    secondary: {
      main: "#7380ec",
    },
    customBackground: {
      main: "#f6f6f9",
      dark: "#363949",
    },
    customCardColor: {
      main: "#ffffff",
      dark: "#202528",
    },
  },
});

export default theme;
```

2. Налаштування підключення до Firebase

```
import { getFirestore } from "firebase/firestore";
import { getAuth } from "firebase/auth";
import { initializeApp } from "firebase/app";

const firebaseConfig = {
  apiKey: "AIzaSyAnkanuUGgzqIte9QEGRNMKnPRGvS-Y6lE",
  authDomain: "graduate-work-b8ea5.firebaseapp.com",
  projectId: "graduate-work-b8ea5",
  storageBucket: "graduate-work-b8ea5.appspot.com",
  messagingSenderId: "945204797469",
  appId: "1:945204797469:web:8ec74b959c6b569a1b26f4",
};
```

```
export const app = initializeApp(firebaseConfig);

export const auth = getAuth(app);
export const db = getFirestore();
```

1. Контекст продуктів

```
import { createContext, FC, useContext, useEffect, useState } from "react";
import {
  DateSate,
  PreparedDateSate,
  Product,
  ProductsContextProviderProps,
  ProductsContextProviderTypes,
} from ".";
import {
  collection,
  deleteDoc,
  doc,
  onSnapshot,
  setDoc,
  updateDoc,
} from "firebase/firestore";
import { db } from "../../firebase";
import { generateRandomId } from "../orders/helperFunctions/generateRandomId";

const ProductsContext = createContext<ProductsContextProviderTypes>(
  {} as ProductsContextProviderTypes
);

export const ProductsContextProvider: FC<ProductsContextProviderProps> = ({
  children,
}) => {
  const [products, setProducts] = useState<Product[]>([]);
  const [dateState, setDateState] = useState<DateSate>({
    end: null,
    start: null,
  });
  const [preparedDate, setPreparedDate] = useState<PreparedDateSate>({
    end: null,
    start: null,
  });
  const [isWrongDateOrder, setIsWrongDateOrder] = useState<boolean>(false);

  const handleDateStateChange = (date: Date | null, params: string) => {
    setDateState({ ...dateState, [params]: date });
  };
  const prepareDate = () => {
    if (!!dateState.end && !!dateState.start) {
      const preparedStartDate = dateState.start.toLocaleDateString();
      const preparedEndDate = dateState.end.toLocaleDateString();
      setPreparedDate({ end: preparedEndDate, start: preparedStartDate });
```

```
  } else return;
};

useEffect(() => {
  if (dateState.start && dateState.end && dateState.end < dateState.start) {
    setIsWrongDateOrder(true);
  } else {
    setIsWrongDateOrder(false);
    prepareDate();
  }
  // eslint-disable-next-line
}, [dateState]);

const handleAddProduct = async (product: Omit<Product, "id" | "fid">) => {
  const id = products.length + 1;

  try {
    await setDoc(doc(db, "products", generateRandomId()), {
      id,
      ...product,
    });
  } catch (error) {
    console.error("Error: ", error);
  }
};

const handleEditProduct = async (product: Product) => {
  const updateProduct = doc(db, "products", product.fid);
  // eslint-disable-next-line
  const { fid, ...pushedProduct } = product;
  await updateDoc(updateProduct, { ...pushedProduct });
};

const handleRemoveProduct = async (fid: string) => {
  const docRef = doc(db, "products", fid);

  deleteDoc(docRef).catch((error) => {
    console.error("Помилка при видаленні продукту:", error);
  });
};

useEffect(() => {
  const unsub = onSnapshot(collection(db, "products"), (doc) => {
    const products = doc.docs.map((data: any) => ({
      ...data.data(),
      fid: data.id,
    })) as Product[];

    setProducts(products);
  });

  return () => {
    unsub();
```

```
    };
  }, []);

  const value: ProductsContextProviderTypes = {
    products,
    handleAddProduct,
    handleEditProduct,
    handleRemoveProduct,
    handleDateStateChange,
    dateState,
    isWrongDateOrder,
    preparedDate,
  };
  return (
    <ProductsContext.Provider value={value}>
      {children}
    </ProductsContext.Provider>
  );
};

// eslint-disable-next-line
export const useProductsContext = () => {
  return useContext(ProductsContext);
};
```

2. Компонент відображення статистики

```
import { FC, useEffect, useState } from "react";
import { Box } from "@mui/material";
import { useProductsContext } from "../../../../../context/products/ProductsContext";
import { MyBarChart } from "../my-bar-chart";

type SalesProductAccordionStatisticProps = {
  productId: string;
};

export const SalesProductAccordionStatistic: FC<
  SalesProductAccordionStatisticProps
> = ({ productId }) => {
  const { products, dateState } = useProductsContext();
  const [isWrongDate, setIsWrongDate] = useState<boolean>(false);

  useEffect(() => {
    if (dateState.start && dateState.end) {
      setIsWrongDate(true);
    } else {
      setIsWrongDate(false);
    }
  }, [dateState]);

  return (
    <>
```

```
      {!isWrongDate || productId === "" || products.length === 0 ? (
        <Box mt={1}>Не правильна дата чи не вибраний продукт</Box>
      ) : (
        <Box mt={3}>
          <MyBarChart productId={productId} />
        </Box>
      )}
    </>
  );
};
```

3. Компонент маршрутизації по проекту

```
import { FC } from "react";
import { Route, Routes } from "react-router-dom";
import { PageNotFound } from "../pages/pageNotFound";
import { PRIVATE_ROUTES, PUBLIC_ROUTES } from "./list";
import { useAuthContext } from "../context/auth/AuthContext";
import { LinearProgress } from "@mui/material";

export const Root: FC = () => {
  const { isUserLogined, isDataFetched } = useAuthContext();

  return (
    <>
      {isDataFetched ? (
        <>
          {isUserLogined ? (
            <Routes>
              {PRIVATE_ROUTES.map(({ path, component }) => (
                <Route path={path} element={component} key={`Route-${path}`} />
              ))}
              <Route path="*" element={<PageNotFound />} />
            </Routes>
          ) : (
            <Routes>
              {PUBLIC_ROUTES.map(({ path, component }) => (
                <Route path={path} element={component} key={`Route-${path}`} />
              ))}
              <Route path="*" element={<PageNotFound />} />
            </Routes>
          )}
        </>
      ) : (
        <LinearProgress />
      )}
    </>
  );
};
```

4. Компонент з шляхами для маршрутизації

```
import { AdminPanelLayout } from "../pages/admin-panel-page";
import { DataGridOuter } from "../pages/admin-panel-page/components/data-grid";
import { Reports } from "../pages/admin-panel-page/components/repots";
import { OrderPage } from "../pages/order-page";
import { Route } from "./types";
import { Login } from "../pages/login";
import { AddProduct } from "../pages/admin-panel-page/components/addProduct";
import { Products } from "../pages/admin-panel-page/components/products";

export const PUBLIC_ROUTES: Route[] = [
  {
    path: "/",
    component: <OrderPage />,
  },
  {
    path: "/login",
    component: <Login />,
  },
];

export const PRIVATE_ROUTES: Route[] = [
  {
    path: "/table",

    component: (
      <AdminPanelLayout>
        <DataGridOuter />
      </AdminPanelLayout>
    ),
  },

  {
    path: "/reports",

    component: (
      <AdminPanelLayout>
        <Reports />
      </AdminPanelLayout>
    ),
  },
  {
    path: "/addProduct",

    component: (
      <AdminPanelLayout>
        <AddProduct />
      </AdminPanelLayout>
    ),
  },
  {
    path: "/",

    component: (
```

```
      <AdminPanelLayout>
        <Products />
      </AdminPanelLayout>
    ),
  },
];
```

5. Компонент авторизації

```
import { Box, Button, IconButton, TextField, Typography } from "@mui/material";
import { FC, useState } from "react";
import { Link } from "react-router-dom";
import { useStyles } from "./styles";
import VisibilityIcon from "@mui/icons-material/Visibility";
import VisibilityOffIcon from "@mui/icons-material/VisibilityOff";
import { useAuthContext } from "../../context/auth/AuthContext";
import { useFormik } from "formik";
import { validationSchema } from "./validationSchema";

export const Login: FC = () => {
  const [isPasswordShowed, setIsPasswordShowed] = useState<boolean>(false);
  const { handleLogin } = useAuthContext();

  const formik = useFormik({
    initialValues: {
      password: "123456",
      email: "admin@gmail.com",
    },
    onSubmit: (values) => {
      handleLogin(values);
      formik.setValues({ password: "", email: "" });
    },
    validationSchema,
  });

  const style = useStyles();
  return (
    <Box className={style.wrapper}>
      <form onSubmit={formik.handleSubmit} className={style.content}>
        <Typography variant="h4" sx={{ textAlign: "center" }}>
          Авторизуватись як адмін
        </Typography>
        <Box>
          <Typography>Почта:</Typography>
          <TextField
            fullWidth
            placeholder="Email"
            type="text"
            onChange={formik.handleChange}
            id="email"
            helperText={formik.errors.email && formik.errors.email}
            value={formik.values.email}
```

```
        />
      </Box>

      <Box>
        <Typography>Пароль:</Typography>
        <Box sx={{ display: "flex", gap: "10px" }}>
          <TextField
            fullWidth
            id="password"
            placeholder="Пароль"
            type={isPasswordShowed ? "text" : "password"}
            onChange={formik.handleChange}
            helperText={formik.errors.password && formik.errors.password}
            value={formik.values.password}
          />
          <IconButton onClick={() => setIsPasswordShowed(!isPasswordShowed)}>
            {isPasswordShowed ? <VisibilityIcon /> : <VisibilityOffIcon />}
          </IconButton>
        </Box>
      </Box>

      <Box sx={{ width: "100%" }}>
        <Button
          disabled={
            formik.values.email.length <= 0 ||
            formik.values.password.length <= 5 ||
            !!formik.errors.email ||
            !!formik.errors.email
          }
          fullWidth
          variant="contained"
          color="secondary"
          type="submit"
        >
          Авторизуватись
        </Button>
        <Typography sx={{ textAlign: "center", marginTop: "10px" }}>
          Якщо ви не маєте данних від аккаунту, то пропонуємо вам{" "}
          <Link to="/"> повернутись на головну</Link>.
        </Typography>
      </Box>
    </form>
  </Box>
 );
};
```