

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

на тему: **РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА
ВИРОБНИЦТВО СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

Стеблянко Владислав Вікторович

Керівник:

Агаджанова Світлани Володимирівна

к.т.н., доцент кафедри кібернетики та інформатики

Рецензент:

Дегтярьова Неля Валентинівна

к.п.н., доцент, завідувач кафедри інформатики СумДПУ ім. А.С. Макаренка

Суми – 2024

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	«Бакалавр»
Спеціальність	<u>126 «Інформаційні системи та технології»</u> <u>ОП «Інформаційні системи та технології»</u>

Завідувач
кафедри

Затверджую:
Світлана АГАДЖАНОВА

« » 202 р

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ *Стеблянко Владислав Вікторович*

1. Тема роботи: Розробка вебзастосунку обліку витрат на виробництво сільськогосподарської продукції.

2. База практичного дослідження: Сумський НАУ
керівник роботи к.т.н. доцент Агаджанова Світлани Володимирівна
затверджена наказом по університету від « »

3. Строк подання студентом закінченого проекту (роботи) «28» 06 2024 року

4. Вихідні дані до проекту(роботи):
Результати аналітичних досліджень зарубіжних та вітчизняних вчених, інтернет-джерела, нормативні документи за чинним законодавством.

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- проаналізувати предметну область та виконати постановку задачі;
- вибрати середовище для реалізації;
- розробити технічне завдання;
- розробити програмний продукт відповідно до поставлених задач;

Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту(роботи)	Строк виконання етапів проєкту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	серпень 2023 р.	
2	Підбір та вивчення літературних джерел, технічної документації	жовтень - листопад 2023 р.	
3	Розробка технічного завдання	грудень 2023 р.	
4	Розробка вебзастосунку	січень-квітень 2024 р.	
5	Оформлення пояснювальної записки	до 30 травня 2024 р.	
6	Оформлення презентації	до 1 червня 2024 р.	
7	Опрацювання зауважень керівника, перевірка на автентичність	до 2 червня 2024 р.	
8	Підготовка доповіді та її попередній захист	до 6 червня 2024 р.	
9	Перевірка з академічної доброчесності кваліфікаційної роботи	до 10 червня 2024 р.	
10	Представлення кваліфікаційної роботи на кафедрі, деканат	23 червня 2024 р.	
11	Захист кваліфікаційної роботи	28 червня 2024р.	

Студент

(підпис)

Владислав СТЕБЛЯНКО

Керівник роботи

(підпис)

Світлана АГАДЖАНОВА

Перевірку на автентичність проведено

(підпис)

Надія БАРАНИК

Дипломна робота допущена до захисту

(підпис)

Маргарита ЛИШЕНКО

АНОТАЦІЯ

Стеблянюк В.В. Розробка вебзастосунку обліку витрат на виробництво сільськогосподарської продукції.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП «Інформаційні системи та технології» СНАУ Суми-2024р. – Рукопис.

Ця робота присвячена розробці вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції. Метою розробки є автоматизація процесів обліку витрат, що забезпечує ефективне управління ресурсами аграрних підприємств.

Вебзастосунок дозволяє аграрним підприємствам вести детальний облік витрат, що сприяє більш точному контролю за використанням ресурсів та підвищує прозорість фінансових операцій. Система включає в себе різноманітні функції, зокрема авторизацію та реєстрацію користувачів, генерацію таблиць витрат, збереження та відображення даних розрахунків.[10]

Розробка вебзастосунку включала аналіз сучасних технологій та інструментів, таких як React, Node.js, Express, MongoDB, та TypeScript. Клієнтська частина реалізована з використанням React.js для створення сучасного і інтерактивного інтерфейсу користувача, а серверна частина – на Node.js з фреймворком Express для забезпечення швидкого та гнучкого бекенду. Для зберігання даних використовується база даних MongoDB.[1]

Впровадження цього вебзастосунку забезпечить аграрним підприємствам інструмент для автоматизації процесів обліку витрат, що підвищить точність і швидкість обробки даних та знизить вплив людського фактора.

Ключові слова: вебзастосунок, облік витрат, сільськогосподарська продукція, автоматизація, MongoDB, Node.js, React.

ABSTRACT

Steblyanko V.V. Development of the web-based budget for the production of agricultural products.

Qualification of work in this specialty 126 “Information systems and technologies” EP “Information systems and technologies” SNAU Sumi-2024r. - Manuscript.

This work is dedicated to the development of a web-based solution to invest in the production of agricultural products. The method of development is the automation of the processes of input payments, which will ensure the effective management of resources of agricultural enterprises.

The web-based system allows agricultural enterprises to maintain a detailed view of their inputs, which allows for more precise control over their resources and increases visibility of financial transactions. The system includes a variety of functions, including authorization and registration of accountants, generation of a tax table, saving and updating of account data.

The development of the web site included an analysis of current technologies and tools, such as React, Node.js, Express, MongoDB, and TypeScript. The client part is implemented using React.js to create a daily and interactive backend interface, and the server part is implemented using Node.js with the Express framework to provide a smooth and flexible backend. To store the data, a MongoDB database is used.

The advanced web application will provide agricultural enterprises with a tool for automating the processes of cash flows, which will increase the accuracy and speed of data processing and reduce the input of human factors.

Keywords: webzastosunok, vitrat cloud, agricultural products, automation, MongoDB, Node.js, React.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Дослідження актуальності проблеми	9
1.2 Аналіз програмних продуктів – аналогів	11
1.3 Постановка задачі	12
РОЗДІЛ 2 МОДУЛЮВАННЯ ТА ПРОЄКТУВАННЯ	15
2.1 Формули для створення плану робіт на земельній ділянці	15
2.2 Вибір засобів реалізації	19
2.3 Проектування бази даних проекту	21
РОЗДІЛ 3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	25
3.1 Архітектура програмного додатку	25
3.2 Методи розробки клієнтської частини додатку зі сторони розробника	28
3.3 Реалізація вебзастосунку на практиці	34
ВИСНОВКИ	56
СПИСОК ЛІТЕРАТУРИ	57
ДОДАТКИ	59
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ	60
ДОДАТОК Б ПЛАНУВАННЯ РОБІТ	66
ДОДАТОК В КОПІЇ ПУБЛІКАЦІЙ	70
ДОДАТОК Г ЛІСТИНГИ ПРОГРАМ	75

ВСТУП

Сучасні аграрні підприємства вже активно інтегрують інформаційні системи в свою щоденну діяльність, використовуючи їх для управління виробничими процесами, моніторингу ресурсів та оптимізації витрат. В умовах динамічного ринку і постійної потреби в підвищенні ефективності, роль інформаційних технологій стає ключовою для успішного функціонування агробізнесу.

Існуючі інформаційні системи надають аграріям можливість автоматизувати широкий спектр завдань, включаючи управління земельними ділянками, контроль за станом врожаю, планування сівозмін і облік витрат. Впровадження таких систем дозволяє не лише знижувати операційні витрати, але й підвищувати точність і швидкість прийняття управлінських рішень.

Однак, незважаючи на численні переваги існуючих систем, потреба в їх удосконаленні та розширенні функціональності залишається актуальною. Одним з важливих напрямків розвитку є створення спеціалізованих вебзастосунків, які дозволяють вести детальний облік витрат на виробництво сільськогосподарської продукції. Такий підхід забезпечує більш точний контроль за використанням ресурсів і підвищує прозорість фінансових операцій.

Розробка вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції стане важливим доповненням до існуючих інформаційних систем аграрних підприємств. Цей додаток буде інтегруватися з іншими управлінськими програмами, що дозволить створити єдину інформаційну екосистему для комплексного управління всіма аспектами виробництва.

Головною метою розробки є забезпечення малих аграрних підприємств інструментом, який дозволить автоматизувати процеси обліку витрат, підвищити точність і швидкість обробки даних, а також знизити вплив людського фактора. Вебзастосунок буде надавати можливість детального моніторингу витрат, аналізу економічної ефективності виробництва та прогнозування майбутніх витрат.

Крім того, вебзастосунок сприятиме підвищенню рівня прозорості фінансових операцій і допоможе виявляти неефективні витрати. Завдяки інтеграції з іншими інформаційними системами, він забезпечить єдиний доступ до всіх необхідних даних, що значно полегшить управління аграрними підприємствами.

Таким чином, розробка вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції є важливим кроком у напрямку вдосконалення інформаційних систем агробізнесу. Впровадження такого додатку дозволить аграріям більш ефективно управляти своїми ресурсами, знижувати витрати і підвищувати продуктивність виробництва, що в кінцевому підсумку сприятиме підвищенню конкурентоспроможності на ринку.

Апробація дослідження – доповіді на науково -практичних конференціях студентів, аспірантів та викладачів Сумського НАУ.

1. Стеблянюк В. В. Аналітичний огляд іт-технологій обліку витрат на виробництво сільсько-господарської продукції. Постановка задачі: матеріали всеукраїнської наукової конференції студентів та аспірантів, присвяченої міжнародному дню студента, 13-17 листопада 2023 р. Суми, 2023. С. 330. [20]
2. Стеблянюк В. В. Розробка вебзастосунку обліку витрат на виробництво продукції. Реалізація проєкту: матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ, 14-16 травня 2024 р. Суми: СНАУ. С. 424[21]

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Актуальність розробки вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції обумовлена необхідністю підвищення ефективності управління аграрними підприємствами та оптимізації витрат. Використання сучасних технологій дозволяє автоматизувати облік та аналіз витрат, що є ключовим фактором у прийнятті обґрунтованих управлінських рішень.

Сучасні аграрні підприємства стикаються з викликами, пов'язаними з ефективним управлінням витратами на виробництво. Традиційні методи обліку часто є трудомісткими, схильними до помилок та не забезпечують оперативного доступу до даних. Вебзастосунок дозволяє:

1. Автоматизувати процеси обліку: Зменшення обсягу ручної праці та зниження ризику помилок завдяки автоматичному збору та обробці даних.
2. Забезпечити оперативний доступ до інформації: Всі дані зберігаються в єдиній системі, що дозволяє швидко отримувати актуальну інформацію про витрати.

Використання вебзастосунку сприяє підвищенню ефективності управління за рахунок:

1. Прозорості обліку: Деталізований облік витрат дозволяє керівникам підприємств отримувати повну картину про використання ресурсів та виявляти неефективні витрати.
2. Аналізу та прогнозування: Можливість аналізу історичних даних і прогнозування майбутніх витрат на основі різних сценаріїв розвитку подій. Вебзастосунок допомагає оптимізувати виробничі процеси шляхом:

1. Точного планування ресурсів: Визначення оптимальної кількості необхідних матеріалів та обладнання для виконання робіт.
2. Підвищення продуктивності праці: Автоматизація рутинних операцій дозволяє звільнити час працівників для виконання більш важливих завдань.[11]

Автоматизація обліку витрат призводить до зниження загальних витрат на виробництво за рахунок:

1. Зменшення витрат на персонал: Скорочення кількості необхідних працівників для ведення обліку та зниження витрат на їх навчання.
2. Оптимізації використання ресурсів: Зниження витрат і перевитрат завдяки точному обліку та контролю використання ресурсів.

Розробка вебзастосунку сприяє підвищенню конкурентоспроможності аграрного підприємства на ринку:

1. Швидка адаптація до змін: Оперативний доступ до інформації дозволяє швидко реагувати на зміни ринкових умов і приймати обґрунтовані рішення.
2. Підвищення якості продукції: Оптимізація виробничих процесів і ефективне використання ресурсів сприяють підвищенню якості виробленої продукції.

Таким чином, розробка вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції є актуальною і необхідною для сучасних аграрних підприємств. Вона забезпечує автоматизацію процесів обліку, підвищує ефективність управління, оптимізує виробничі процеси, знижує витрати та підвищує конкурентоспроможність на ринку.

1.2 Аналіз програмних продуктів - аналогів

Український державний портал Дистанційного Зондування Землі створив вебзастосунок, який має широкий функціонал..[6]

Функції, які підтримує додаток розміщені в таблиці 1.1:

Таблиця 1.2 – Огляд функціоналу схожих додатків

LST Змінна температури земної поверхні	Проводиться моніторинг температури підстильної поверхні (земної поверхні, водної поверхні, верхнього шару хмарного покриву). Під температурою земної поверхні розуміється радіаційна температура поверхні. Температура позначається в градусах за шкалою Цельсія
Поверхнева температура Азово- Чорноморського басейну	Карта поверхневої температури Азово-Чорноморського басейну, створена за методикою Морського гідрофізичного інституту НАН України, призначена для науково-методичного й технологічного забезпечення використання аерокосмічних технологій дистанційного зондування Землі в практиці господарської і управлінської діяльності, для вирішення тематичних завдань моніторингу морських і океанських акваторій
Прогноз дощу та снігу	Для попередження виникнення надзвичайних ситуацій формуються тематичні карти прогнозування атмосферних опадів на території України. Тематична карта дає можливість наочно оцінити прогнозований рівень опадів і скоординувати роботу відповідних служб України задля попередження та ліквідації надзвичайних ситуацій.
Прогноз пожежонебезпеки	З метою попередження виникнення пожеж на території України в Центрі створюються прогностичні карти пожежонебезпечних зон із використанням коефіцієнта горимості, розробленого В.Г.Нестеровим. Прогноз надає можливість координувати роботу пожежоохоронних служб.
Моніторинг стану посухи на території України	Проводиться регулярний сезонний моніторинг танення снігового покриву та весняного водопілля на території України за даними порталу Дистанційного Зондування Землі. За допомогою шкали навігації в правому кутку демонструється дев'ять можливих станів покриття ґрунту — від суцільного сніжного покриву до сухого ґрунту

Джерело: запропоновано автором

Функціонал додатків вражає своїми можливостями. Вони корисні не лише для обробки земельних ділянок, а й у науковій сфері. Хоча робота над додатками ще триває, оскільки вони знаходяться у тестовому режимі, вже зараз очевидно, що їх можна доповнити й зробити доступним для ширшого кола користувачів, включаючи тих, хто не займається наукою. Додаткові можливості можуть включати:

1. Авторизацію та реєстрацію користувачів;
2. Розрахунок вартості посіву на різних земельних ділянках з урахуванням області, розміру посівного поля та курсу валют ;
3. Базу статей, яка надаватиме користувачам відповіді на їхні питання.

Такий функціонал буде дуже корисним для користувачів додатку і дозволить викласти його у вільний доступ.

1.3 Постановка задачі

Метою даної дипломної роботи є розробка інформаційного вебзастосунку, який забезпечить автоматизований облік витрат на виробництво сільськогосподарської продукції. Додаток повинен мати інтуїтивно зрозумілий інтерфейс, ефективну базу даних, а також відповідати сучасним вимогам безпеки і продуктивності. Реалізація цього проєкту сприятиме підвищенню ефективності управління аграрними підприємствами та зменшенню витрат на виробництво за рахунок оптимізації ресурсів.

Основні задачі

Огляд інструментів для розробки:

1. Провести аналіз сучасних технологій і інструментів для розробки вебдодатків, таких як React, Node.js, Express, MongoDB та TypeScript.
2. Обрати оптимальні інструменти, які забезпечать високу продуктивність, надійність та зручність у розробці та експлуатації додатку.

Розробка клієнтської частини:

1. Створити користувацький інтерфейс, який забезпечить інтуїтивно зрозуміле та зручне використання додатку.

2. Використовувати адаптивний дизайн, щоб додаток коректно відображався на різних пристроях, включаючи комп'ютери, планшети та смартфони.

Проектування та реалізація серверної частини

1. Спроекувати базу даних, яка забезпечить ефективне зберігання і швидкий доступ до даних про витрати на виробництво.
2. Реалізувати серверну частину додатку з використанням Node.js та Express, забезпечуючи безпечну і стабільну роботу серверу.[1]

Інтеграція з базою даних:

1. Налаштувати взаємодію клієнтської та серверної частини з базою даних MongoDB.
2. Забезпечити зручний і швидкий доступ до даних для проведення розрахунків і генерації звітів.

Реалізація функціональних можливостей

1. Впровадити функцію авторизації та реєстрації користувачів, що забезпечить захист даних та персоналізацію користувацького досвіду.
2. Розробити механізм обрахунку вартості посіву на основі обраної області, розміру земельної ділянки та актуального курсу долара.
3. Додати можливість збереження і обробки даних про виконані роботи, витрати та використані матеріали.
4. Створити базу статей, яка буде надавати користувачам корисну інформацію і допомагати у вирішенні виникаючих питань.

Тестування та оптимізація:

1. Провести тестування всіх функціональних можливостей додатку для виявлення і виправлення помилок.
2. Оптимізувати роботу додатку для забезпечення високої швидкості і надійності роботи при різних навантаженнях.

Реалізація цих задач забезпечить створення ефективного та зручного у використанні вебзастосунку для обліку витрат на виробництво сільськогосподарської продукції. Це дозволить аграрним підприємствам

знижувати витрати, підвищувати продуктивність і ефективність управління виробничими процесами. Завдяки інноваційним технологіям та сучасним підходам до розробки програмного забезпечення, даний додаток може бути розширений та вдосконалений для широкого використання в аграрному секторі.

РОЗДІЛ 2

МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ

2.1 Формули для створення плану робіт на земельній ділянці

Розрахунок витрат для обробітку пшениці

1. Посів ділянки:

- Посів та внесення добрив: насіння = 200 кг, міндобрива NPK = 150 кг
- Внесення сульфату амонію на земельну ділянку в 1 га: 100 кг
- Внесення аміачної селітри на земельну ділянку в 1 га: 150 кг

2. Захист рослини:

- Листкове підживлення з гербіцидом: гербіцид = 0.5 л, фунгіцид = 0.5 л, мікроелементи = 0.2 л
- Листкове підживлення з інсектицидом: фунгіцид = 0.5 л, інсектицид = 0.15 л
- Підживлення під час колосіння: фунгіцид = 0.5 л, карбамід = 12 кг

Формула витрат:

$$V = 200n_1 + 150p + 100s + 150c + 0.5g + 0.5f + 0.2m + 0.5f + 0.15i + 0.5f + 12k$$

Умовні позначки:

- n_1 - насіння пшениці (1 кг), вартість = 10 грн
- p - міндобрива NPK (1 кг), вартість = 20 грн
- s - сульфат амонію (1 кг), вартість = 20 грн
- c - аміачна селітра (1 кг), вартість = 28 грн
- g - гербіцид (1 л), вартість = 1000 грн
- f - фунгіцид (1 л), вартість = 800 грн
- m - мікроелементи (1 л), вартість = 350 грн
- i - інсектицид (1 л), вартість = 1000 грн
- k - карбамід (1 кг), вартість = 34 грн

Розрахунок витрат для обробітку кукурудзи

1. Підготовка ґрунту на 1 га:

- Внесення NPK: 150 кг
- Внесення КАС-32: 200 кг

2. Посів ділянки:

- Посів: насіння = 1 п.о.

3. Захист рослин:

- Листкове підживлення з гербіцидом: гербіцид = 1.2 л, мікроелементи = 0.2 л
- Листкове підживлення з інсектицидом: цинк = 0.5 л, інсектицид = 0.15 л

Формула витрат:

$$V = 150p + 200q + n_2 + 1.2g + 0.2m + 0.5x + 0.15i$$

Умовні позначки:

- n_2 - насіння кукурудзи (1 п.о.), вартість = 5000 грн
- p - мінеральна NPK (1 кг), вартість = 20 грн
- q - КАС-32 (1 кг), вартість = 20 грн
- g - гербіцид (1 л), вартість = 1000 грн
- m - мікроелементи (1 л), вартість = 350 грн
- i - інсектицид (1 л), вартість = 1000 грн
- x - цинк (1 л), вартість = 400 грн

Розрахунок витрат для обробітку соняшника

1. Підготовка ґрунту на 1 га:

- Внесення NPK: 150 кг
- Внесення КАС-32: 100 кг

2. Посів ділянки:

- Посів: насіння = 1 п.о.

3. Захист рослин:

- Листкове підживлення з гербіцидом: гербіцид = 1.2 л, мікроелементи = 0.2 л
- Листкове підживлення з фунгіцидом: фунгіцид = 0.5 л

Формула витрат:

$$V = 150p + 100q + n_2 + 1.2g + 0.2m + 0.5f$$

Умовні позначки:

- n2 - насіння соняшника (1 п.о.), вартість = 3000 грн
- p - міндобрива NPK (1 кг), вартість = 20 грн
- q - КАС-32 (1 кг), вартість = 20 грн
- g - гербіцид (1 л), вартість = 1000 грн
- m - мікроелементи (1 л), вартість = 350 грн
- f - фунгіцид (1 л), вартість = 800 грн

Побудова Вебдодатків: Вибір Технологій

Для створення вебдодатків сьогодні існує широкий спектр технологій, які можуть бути адаптовані до конкретних завдань. Окрім традиційного використання JavaScript, розробники активно використовують інші мови програмування, наприклад, Python. Однак сформувалися і інші фаворити серед інструментів для веброзробки.

Клієнтська Частина: React.js, Angular, Vue.js

При розробці клієнтської частини вебдодатків перевага надається трьом основним інструментам: React.js, Angular та Vue.js. React є бібліотекою, тоді як Angular і Vue є фреймворками. Основна відмінність між фреймворками і бібліотеками полягає у тому, що фреймворки включають більшість необхідних інструментів "з коробки". У випадку з бібліотеками, такими як React, додаткові модулі (наприклад, для маршрутизації) потрібно встановлювати окремо за допомогою npm або yarn.

Серверна Частина: Java, C#, Node.js

Для серверної частини вебдодатків можна використовувати більш широкий спектр технологій. Наприклад, Java і C# є довготривалими і актуальними інструментами на ринку. Якщо ж є бажання використовувати одну мову програмування для клієнтської і серверної частин, часто вибирають Node.js. Для Node.js популярними є такі фреймворки, як Express та Nest. Express є легким і мінімалістичним фреймворком, а Nest, написаний на TypeScript, пропонує більш структурований підхід до розробки, підтримуючи сучасні концепції, такі як DDD (Domain-Driven Design), подійно-орієнтована архітектура та мікросервіси.

Вибір Бази Даних

При розробці вебдодатків також постає питання вибору бази даних. Найбільш поширені реляційні бази даних, такі як MySQL, Oracle DB та PostgreSQL. Водночас, бази даних, побудовані на основі документів, такі як MongoDB, набирають популярність завдяки своїй гнучкості та простоті у використанні для менш навантажених систем.

Node.js - програмна платформа, яка перетворює з JavaScript на мову загального призначення, додаючи можливість взаємодії з пристроями через свій API та підключення зовнішніх бібліотек, написаних на інших мовах. Node.js використовується переважно на серверній частині застосунку, виконуючи роль веб-сервера. Однак, його можна використовувати і для розробки десктопних додатків, використовуючи NW.js, AppJS або Electron, а також для програмування мікроконтролерів, таких як Tessel і Espruino. Node.js базується на асинхронному програмуванні з неблокуючим введенням-виведенням.

NestJS - фреймворк, який пропонує структурований підхід до розробки на базі Node.js. Він побудований на TypeScript і підтримує передові архітектурні підходи, такі як DDD, подійно-орієнтована архітектура та мікросервіси. NestJS дозволяє використовувати як всю платформу, так і окремі її компоненти, що забезпечує гнучкість у розробці.

Express.js - популярний фреймворк для Node.js, який відзначається своєю швидкістю, мінімалізмом та гнучкістю. Це фактичний API для написання вебдодатків використовуючи Node.js, який надає базові функції для створення веб та мобільних додатків. Express забезпечує стабільну маршрутизацію та допоміжні інструменти для роботи з HTTP-запитами, систему перегляду з підтримкою понад 14 шаблонів, а також інші утиліти для полегшення розробки надійних API. Завдяки простоті використання, Express часто вибирають для швидкої генерації програм.

За даними офіційних сайтів, Express завантажують набагато частіше, ніж інші фреймворки для Node.js, що свідчить про його популярність серед розробників для створення програмних засобів.

Вибір технологій для розробки вебдодатків залежить від конкретних вимог проекту та уподобань команди розробників. Використання сучасних фреймворків та бібліотек для клієнтської та серверної частин, а також вибір відповідної бази даних, може значно вплинути на ефективність та успішність розробки вебдодатків.

2.2 Вибір засобів реалізації

Для реалізації нашого проекту було обрано кілька інструментів та технологій, які забезпечують високу продуктивність, надійність та зручність у розробці і подальшій експлуатації додатку. Розглянемо їх детальніше, враховуючи їхні переваги та недоліки.

Серверна частина

Node.js

- Переваги:

- Швидкість та ефективність завдяки подієво-орієнтованій архітектурі.
- Єдина мова програмування (JavaScript) на клієнтській та серверній частині, що спрощує розробку.
- Велика спільнота та багато доступних модулів і бібліотек.

- Недоліки:

- Асинхронна природа може бути складною для розуміння та відлагодження.
- Однопоточовість може бути обмеженням для задач, що вимагають великої обчислювальної потужності.

Express.js

- Переваги:

- Легкий і мінімалістичний фреймворк, що забезпечує високу продуктивність.
- Гнучкість у налаштуванні маршрутів та middleware.
- Велика кількість утиліт для HTTP-запитів.

- Недоліки:

- Вимагає більше ручного налаштування порівняно з більш комплексними фреймворками.
- Відсутність стандартної структури проекту, що може призвести до неоднорідного коду.

База даних

MongoDB

1. Переваги:

- Гнучкість у роботі з неструктурованими даними завдяки документ-орієнтованій моделі.
- Висока масштабованість і продуктивність для великих обсягів даних.
- Простота інтеграції з Node.js через Mongoose або інші ORM.

2. Недоліки:

- Відсутність підтримки транзакцій до версії 4.0, що може бути критично для деяких додатків.
- Відносно складніша модель даних порівняно з реляційними базами даних.

Клієнтська частина

React.js

1. Переваги:

- Висока продуктивність завдяки віртуальному DOM.
- Компонентний підхід, що сприяє повторному використанню коду та кращій організації проекту.
- Велика спільнота і багатий набір інструментів для розробки.

2. Недоліки:

- Високий поріг входження для новачків через складність JSX та потребу в додаткових інструментах для повноцінної розробки.
- Часті оновлення і зміни в бібліотеці можуть вимагати постійного вивчення нововведень.

CSS-in-JS (Styled-Components)

1. Переваги:

- Інкапсуляція стилів у компоненти, що запобігає конфліктам стилів.
- Можливість динамічної зміни стилів у залежності від стану компонентів.

2. Недоліки:

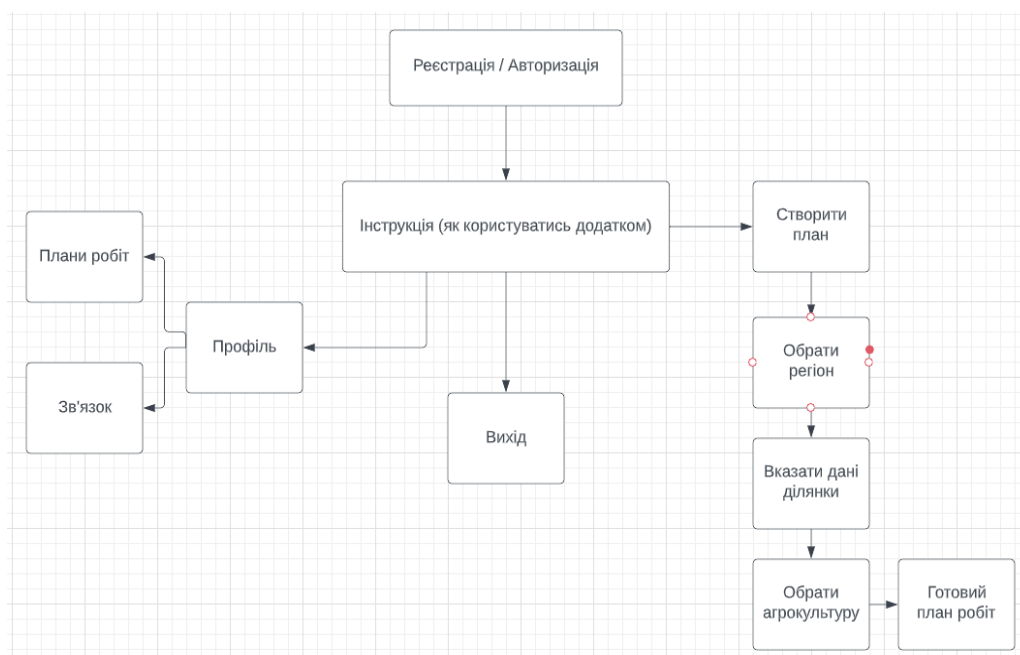
- Зниження продуктивності при великій кількості стилізованих компонентів.
- Більш складна відладка стилів порівняно з традиційними CSS-файлами.

Обрані технології дозволяють ефективно реалізувати наш проект, забезпечуючи високу продуктивність та зручність у розробці. Node.js і Express.js на серверній частині дозволяють створити швидкий та гнучкий бекенд, тоді як MongoDB забезпечує надійне зберігання даних. На клієнтській частині, використання React.js та CSS-in-JS дозволяє створити сучасний і інтерактивний інтерфейс користувача.

2.3 Проектування бази даних проекту

Моделювання інформаційної системи

2.1



Рисунок

Архітектура навігації клієнтської частини вебзастосунку

Джерело: запропоноване автором

На рисунку 3.1 показано навігацію по вебзастосунку, яка буде використовуватись користувачами.

Коли користувач зареєструється та авторизується в додатку, він потрапить на сторінку, де будуть представлені рекомендації щодо ефективного використання додатка та його основного призначення. У панелі навігації користувач зможе перейти до свого профілю або розпочати створення плану робіт.

Якщо користувач обирає створення плану робіт, йому потрібно буде вказати регіон земельної ділянки, надати інформацію про неї та визначити агрокультуру, яку планується використовувати. З цих даних додаток автоматично сформує приблизну таблицю робіт які потрібно виконати.

При натисканні на головну сторінку користувача, зможе переглянути свої плани робіт та зв'язатися з автором додатка для додаткових консультацій або допомоги. У базі даних додатка буде зберігатися кілька основних документів. Серед них основними є:

1. User – містить інформацію про користувачів, включаючи їх персональні дані, плани робіт та налаштування профілю.
2. Articles – включає статті та матеріали, які можуть бути корисні для користувачів під час планування та виконання робіт.
3. Areas – містить дані про різні земельні ділянки, їх характеристики та відповідні агрокультури.

Таблиця 2.1 – Модель бази даних для роботи з модулем User

Поле	Тип даних	Обов'язковість	Унікальність
Email	рядок	істинний	істинний
Username	рядок	істинний	-
Password	рядок	істинний	-
Areas	Масив	-	-

Джерело: запропоноване автором

Опис полів моделі User:

1. Email: Електронна пошта користувача, повинна бути унікальною, щоб забезпечити унікальність кожного облікового запису.

2. Username: Ім'я користувача, обов'язкове для заповнення.
3. Password: Пароль користувача, обов'язковий для заповнення, зберігається у захищеному (захешованому) вигляді.
4. Areas: Масив об'єктів, що містить плани робіт, створені користувачем.

Таким чином, ми виключаємо можливість реєстрації декількох користувачів з однаковою електронною поштою, що дозволяє уникнути багатьох потенційних проблем. Унікальність цього поля важлива для збереження цілісності та безпеки даних користувачів, а також для запобігання можливим конфліктам під час аутентифікації та авторизації в системі.

Таблиця 2.2 – Модель бази даних для роботи з модулем Areas

	Регіон	Ціна	Агро культура
Тип	рядок	рядок	рядок
Вимагається:	істинний	істинний	істинний

Джерело: запропоноване автором

В таблиці 2.2 вписано декілька полів, поле «Регіон» для вибору місця розташування земельної ділянки, «Ціна» для розрахунку ціни за певним регіоном та «Агро культура» для розрахунку по певній культурі, яку планують бути засаджені поля.

Кожен запис в базі даних має свої унікальні поля та атрибути, які визначають їх характеристики, такі як тип даних, обов'язковість для заповнення, а також можливість бути унікальними. MongoDB додатково надає кожному запису унікальний ідентифікатор під назвою `\_id`, який можна використовувати для посилання на цей запис з інших записів у базі даних. Це особливо корисно при створенні зв'язків між різними документами, наприклад, між документами "Користувач" та "Області".

Таблиця 2.3 – Модель бази даних для роботи з модулем Articles

	Назва	Тіло
--	-------	------

Тип	рядок	рядок
Вимога	істинний	істинний
Унікальність	істинний	-

Джерело: запропоноване автором

В таблиці 2.3 розміщаємо поля «назва» та «тіло». «Назва» слугує для закріплення назви статті, а «тіло» в свою чергу відповідає за з міст статті за певною назвою.

РОЗДІЛ 3

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Архітектура програмного додатку

Архітектура програмного додатку складається з клієнтської та серверної частин, які взаємодіють між собою через HTTP запити. Додаток має бути адаптований для використання на різних пристроях, включаючи комп'ютери, планшети та смартфони.

Клієнтська частина додатку реалізована за допомогою HTML, CSS та JavaScript з використанням фреймворку React. Для управління станом додатку застосовується Redux Toolkit. Основні компоненти клієнтської частини:

1. Інтерфейс користувача:

- Головне меню: адаптивне, з "меню-бургером" для мобільних пристроїв.
- Сторінка профілю: дозволяє користувачу переглядати та управляти своїми планами робіт.
- Сторінка створення плану робіт: форма для введення даних про земельні ділянки та агрокультури.

2. Обробка даних:

- Redux reducers: для відокремлення логіки обробки даних від компонентів, що відповідають за відображення контенту.

Серверна частина

Серверна частина реалізована на базі Node.js з використанням фреймворку Express. Основні компоненти серверної частини:

1. API:

- RESTful API: для обробки запитів від клієнтської частини та виконання CRUD операцій над даними у базі даних.[14]

2. База даних:

- MongoDB: використовується для зберігання даних про користувачів, статті та земельні ділянки. Структура бази даних представлена моделями User, Areas та Articles.

Переваги та недоліки архітектури проаналізовано нижче.

Переваги:

- Модульність: чітке розділення клієнтської та серверної частин спрощує розробку та підтримку додатку.
- Масштабованість: використання React та Redux на клієнтській частині дозволяє легко масштабувати додаток та додавати нові функціональні можливості.
- Продуктивність: Express та Node.js забезпечують високу продуктивність серверної частини та ефективну обробку запитів.

Недоліки:

- Складність налаштування: для новачків налаштування та підтримка такого додатку може бути складним завданням через використання декількох технологій.
- Високі вимоги до ресурсів: при великій кількості користувачів може знадобитися масштабування серверної частини для забезпечення стабільної роботи.

Схеми архітектури

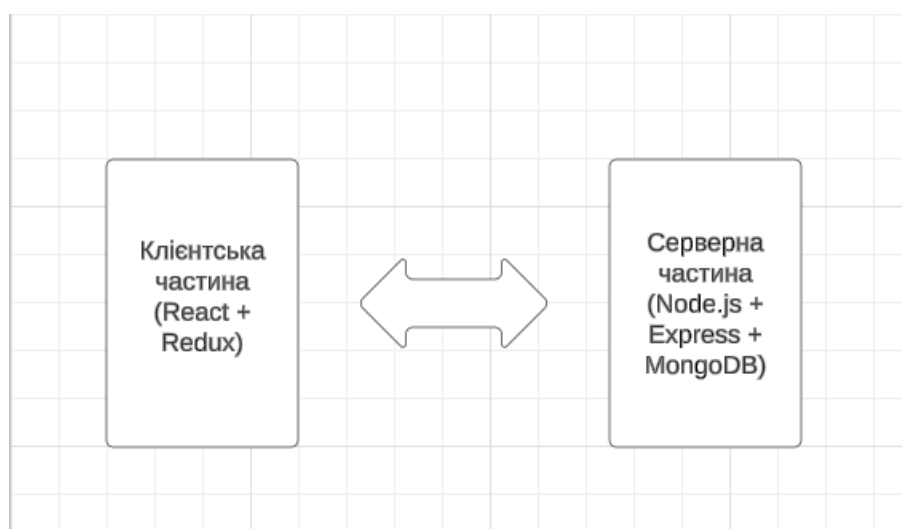


Рисунок 3.1 - Загальна схема архітектури додатку

Джерело: запропоноване автором

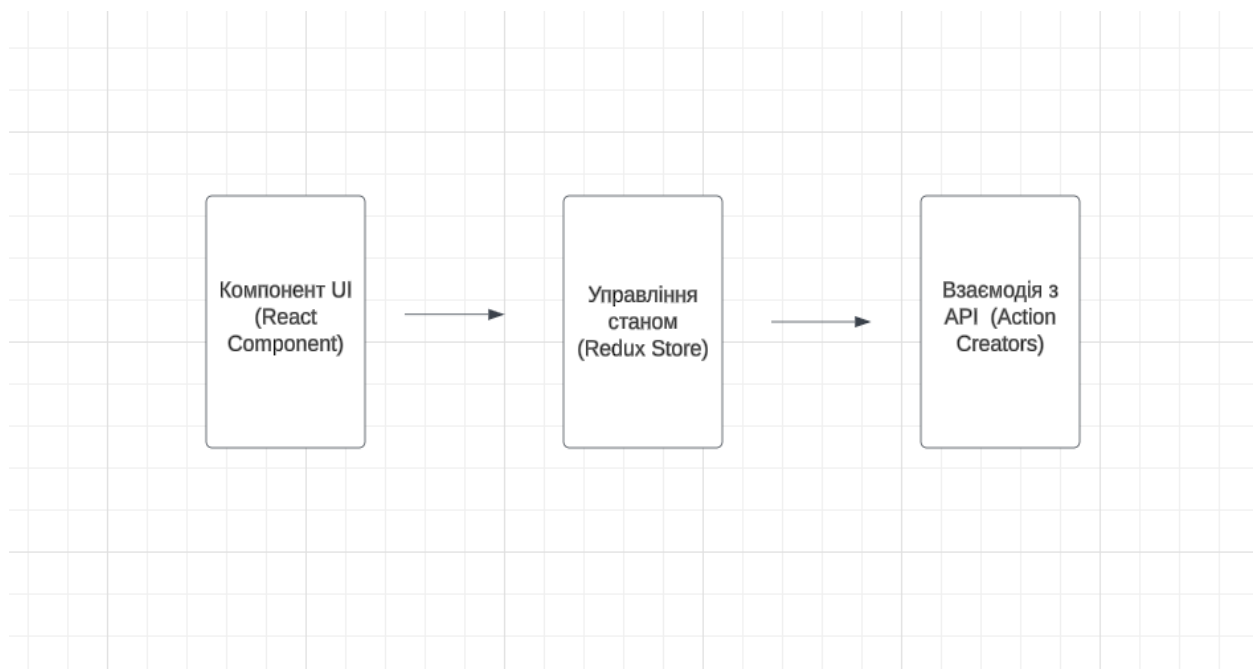


Рисунок 3.2- Схема взаємодії компонентів клієнтської частини

Джерело: запропоноване автором

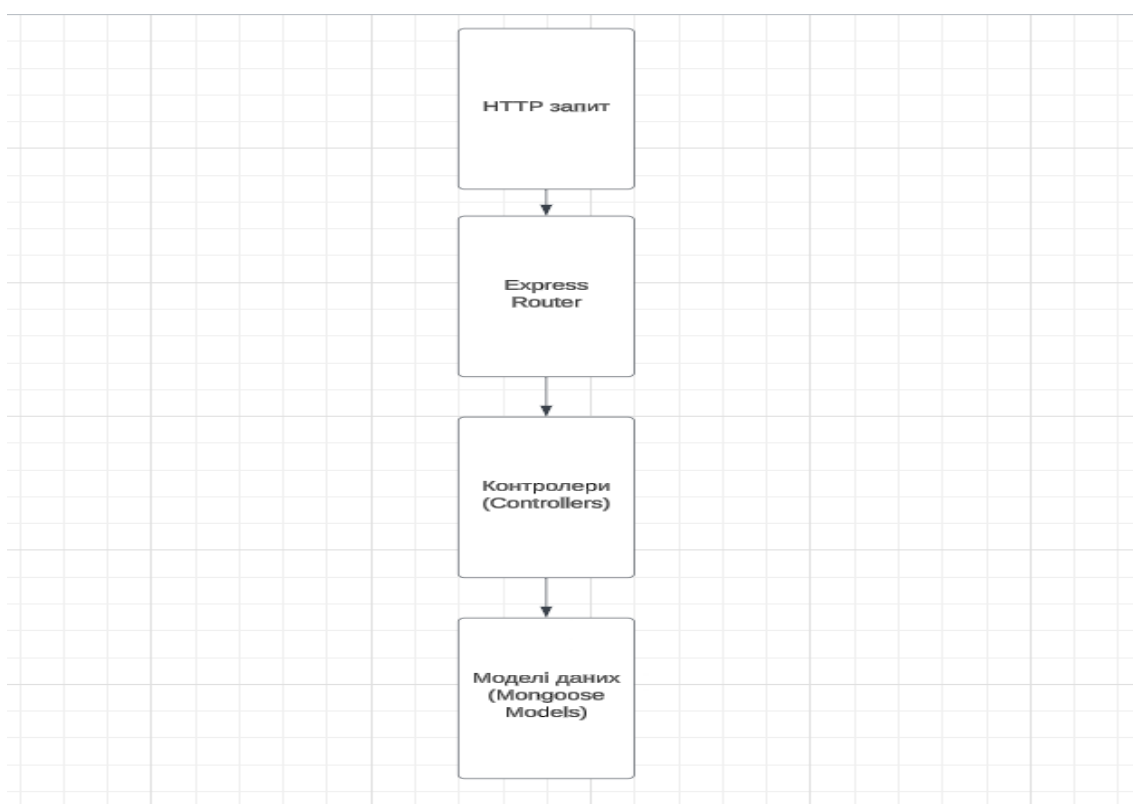


Рисунок 3.3 - Схема обробки запитів на серверній частині

Джерело: запропоноване автором

Ці схеми ілюструють основні компоненти архітектури та взаємодію між ними, що сприяє кращому розумінню структури та функціонування програмного додатку.

3.2 Методи розробки клієнтської частини додатку зі сторони розробника

Розглядаючи клієнтську частину додатку з точки зору кінцевого користувача, основне завдання полягає у забезпеченні якісного та ефективного відображення даних, які надходять від серверної частини. Однак, з точки зору розробників, ключовою задачею є створення такого додатку, який буде легко підтримувати та оновлювати в майбутньому.

У сучасній Frontend-розробці існує декілька стратегій для досягнення цієї мети. Перша і, мабуть, найважливіша стратегія полягає у розділенні структури відображення контенту від логіки обробки інформації. По суті, головна функція компонентів у сучасних фреймворках та бібліотеках полягає у відображенні розмітки. Ідеальною ситуацією було б, якби вони займалися виключно цим. Найвідомішим прикладом є Redux, однак існують і його аналоги, такі як MobX та Effector. Більш просунутою версією Redux є Redux Toolkit, яка надає додаткові можливості для спрощення процесу розробки.

Суть таких утиліт полягає в тому, що вони дозволяють повністю відокремити логіку обробки даних та передавати результати обробки лише в ті компоненти додатку, де вони дійсно необхідні. Це досягається шляхом створення централізованого сховища стану додатку, де зберігаються всі необхідні дані та логіка їх обробки. Компоненти додатку можуть підписуватися на зміни в цьому сховищі та оновлювати своє відображення відповідно до змін у стані.[12]

Використання таких підходів має кілька ключових переваг. По-перше, це дозволяє зробити код додатку більш структурованим та легко підтримуваним. Оскільки логіка обробки даних відокремлена від компонентів, її можна легко змінювати та розширювати без необхідності вносити зміни у компоненти, які

відповідають за відображення. По-друге, це дозволяє зменшити кількість помилок у коді, оскільки вся логіка зосереджена в одному місці, що спрощує її тестування та налагодження. Нарешті, це підвищує продуктивність додатку, оскільки компоненти оновлюються тільки тоді, коли це дійсно необхідно, що зменшує навантаження на браузер та покращує користувацький досвід.[15]

Таким чином, застосування стратегій розділення логіки обробки даних від компонентів відображення є важливим аспектом розробки сучасних вебдодатків. Використання утиліт для управління станом, таких як Redux, MobX або Redux Toolkit, дозволяє створювати більш ефективні, надійні та легко підтримувані додатки, що є ключовим фактором для успішної реалізації будь-якого проекту у сфері веброботки. (див. рис.3.4)

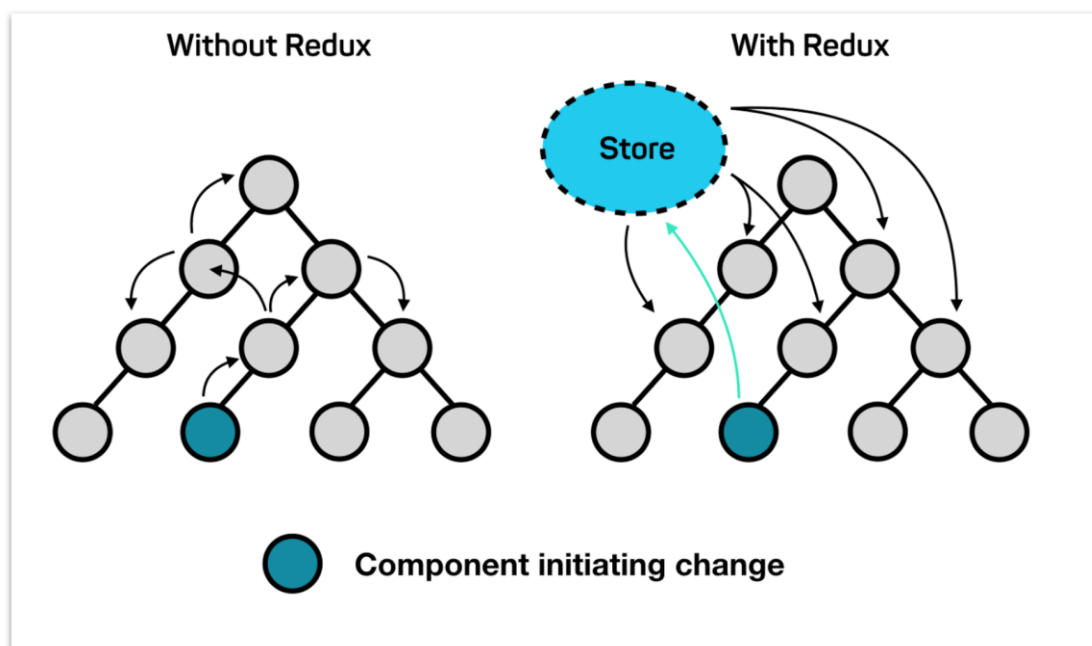


Рисунок 3.4 —Наглядне відображення впливу присутності технології Redux.

Джерело: запропоноване автором

Заповненням прогалин у безпеці JavaScript є використання TypeScript. JavaScript, через свою динамічну типізацію, залишає багато простору для неочікуваних помилок, тоді як TypeScript вводить більшу стабільність і прозорість у коді. Коли ви взаємодієте з сервером і отримуєте відповідь, часто важко передбачити, яка саме структура даних буде повернута. Це може призвести до суттєвих проблем при обробці цих даних. TypeScript дозволяє

визначити типи даних, які ви очікуєте отримати, що значно полегшує розуміння структури даних та запобігає непередбаченим помилкам.

Хоча TypeScript додає додатковий шар абстракції, який пізніше компілюється в JavaScript, цей підхід вартий витрачених зусиль. Завдяки визначенню типів на початкових етапах розробки, ви заощаджуєте багато часу, який інакше було б витрачено на виправлення помилок у великих проектах. TypeScript надає можливість розробникам чітко бачити, які типи даних використовуються в коді, і таким чином знижує ризик виникнення помилок, пов'язаних з неочікуваними змінами структури даних.

Наприклад, при взаємодії з сервером, коли відповідь надходить у формі даних, розробник може не знати точної структури цих даних. Це може створити труднощі в обробці та призвести до багатьох помилок. TypeScript дозволяє точно визначити, які типи даних очікуються, що робить обробку даних більш передбачуваною та безпечною.

Витративши трохи більше часу на визначення типів за допомогою TypeScript, ви зможете уникнути багатьох помилок у майбутньому. Це особливо важливо у великих проектах, де складність коду зростає і виправлення помилок може вимагати значних ресурсів. Таким чином, TypeScript допомагає зберегти час і ресурси, що витрачаються на підтримку та відлагодження коду, забезпечуючи більш надійну і стабільну розробку програмного забезпечення.

Використані технології для розробки

Клієнтська частина:

- React – використовується для створення користувацького інтерфейсу додатку, разом із рядом додаткових інструментів.[18]
- Redux – для управління станом додатку, що дозволяє відокремити бізнес-логіку від UI-компонентів.
- TypeScript – забезпечує типізацію коду на клієнтській стороні, допомагаючи уникнути багатьох помилок на етапі розробки.
- Додаткові бібліотеки для React:
 - React-router-dom – для побудови SPA-додатків.

- Axios – для зручного відправлення запитів на сервер.
- React-redux – забезпечує інтеграцію React-додатку з Redux store.
- Node-sass – дозволяє використовувати синтаксис SCSS у React-додатку.
- Swiper – надає гнучкий і налаштовуваний слайдер із можливістю кастомізації через SCSS.

Серверна частина:

- NodeJS – використовується для створення серверної логіки додатку на JavaScript.
- Express – легкий фреймворк для NodeJS, що спрощує розробку та робить код більш зрозумілим.
- MongoDB – документ-орієнтована база даних, яка підходить для менш навантажених додатків.

Додаткові бібліотеки:

- Mongoose – бібліотека для взаємодії Node.js з MongoDB.
- Config – використовується для зберігання конфігураційних даних у форматі JSON, що застосовуються в різних модулях додатку.
- Express-validator – для валідації даних на сервері.
- Jsonwebtoken – для зберігання даних користувача у вигляді токенів, що використовуються для авторизації.
- Bcryptjs – для хешування паролів користувачів при реєстрації, щоб не зберігати їх у незахищеному вигляді.

Серверна частина вебзастосунку виконує низку важливих завдань, необхідних для забезпечення ефективного функціонування системи. Основні завдання серверної частини включають наступні:

Зв'язок з базою даних: Сервер забезпечує взаємодію з базою даних, через яку проходять всі операції, пов'язані з обробкою даних. Це включає:

- Реєстрацію та авторизацію користувачів: Сервер обробляє запити на реєстрацію нових користувачів і їх авторизацію в системі, забезпечуючи доступ до персональних даних і планів робіт.

- Роботу з довідником статей: Сервер обробляє запити на отримання статей з довідника, надаючи користувачам необхідну інформацію.
- Отримання планів робіт: Сервер виконує запити на отримання планів робіт, зберігаючи і надаючи дані про виконані та заплановані роботи.

Перед тим, як дані зберігаються в базі даних, сервер перевіряє їх на відповідність встановленим критеріям. Це включає:

- Перевірка username та password: Перевірка кількості символів у введених даних. Username та password повинні відповідати встановленим вимогам щодо довжини.
- Перевірка email: Валідація адреси електронної пошти на наявність символу @, що гарантує коректність введених даних.

Серверна частина реалізує різні endpoints, через які користувачі надсилають запити для взаємодії з системою. Кожен endpoint відповідає за виконання конкретних завдань, таких як реєстрація, авторизація, отримання та обробка даних.[3]

На сервері розроблені моделі для роботи з базою даних, що дозволяють виконувати такі операції:

- Пошук даних: Використовуючи моделі, сервер здійснює пошук необхідної інформації в базі даних.
- Створення та видалення елементів: Сервер забезпечує можливість додавання нових записів і видалення існуючих даних в базі даних, що дозволяє підтримувати актуальність інформації.

Серверна частина також використовує middleware для покращення роботи додатку та забезпечення комфортного користувацького досвіду:

- Обхід політики захисту CORS: Middleware допомагає обійти політику захисту CORS, що дозволяє додатку працювати коректно при взаємодії з різними доменами.
- Інші сценарії: Використання додаткових middleware для оптимізації роботи серверу та покращення взаємодії з користувачами, включаючи логування, обробку помилок та інші функції.

Огляд підходів до стилізації вебдодатку

Зараз існує кілька популярних підходів до роботи зі стилями у вебдодатках. Зокрема, це CSS або Sass/SCSS, CSS-in-JS та CSS-бібліотеки. Розглянемо кожен з цих підходів детальніше, зосередившись на їхніх перевагах та недоліках, а також на причинах, які вплинули на вибір конкретного методу для нашого проекту.

Використання традиційних CSS або розширеного препроцесора Sass/SCSS залишається одним з найбільш поширених підходів. Основна перевага цього методу полягає в можливості гнучко й ефективно створювати стилі для компонентів, забезпечуючи високий рівень контролю над візуальним виглядом додатку.

Sass/SCSS додає ряд корисних можливостей:

- Вкладеність: Дозволяє вкладати селектори один в один, що робить код більш організованим.
- Змінні: Забезпечують можливість використання повторюваних значень, таких як кольори та розміри, в одному місці.
- Міксини: Це попередньо визначені блоки стилів, які можна включати в різні частини CSS, не дублюючи код.

Недоліком цього підходу є те, що розробка може займати більше часу, оскільки розробникам потрібно вручну писати більшу кількість коду. Проте, це компенсується чітким розумінням, де саме і які стилі застосовуються, що підвищує якість і підтримуваність коду.

Цей метод пов'язаний з написанням стилів вже в самому JavaScript більш наглядним прикладом є бібліотека styled-components. Цей метод має кілька важливих переваг:

- Локалізація стилів: Стилi знаходяться у тому ж файлі, що і компонент, що полегшує їх пошук та зміну.
- Динамічність: Можливість легко створювати динамічні стилі, використовуючи логіку JavaScript.

Проте, цей підхід має і недоліки:

- Нагромадження коду: Стили та логіка компонента можуть змішуватись, що ускладнює підтримку та читабельність коду.
- Додаткова компіляція: Необхідність компіляції JavaScript у звичайний CSS, зрозумілий для браузера, додає ще один шар складності.

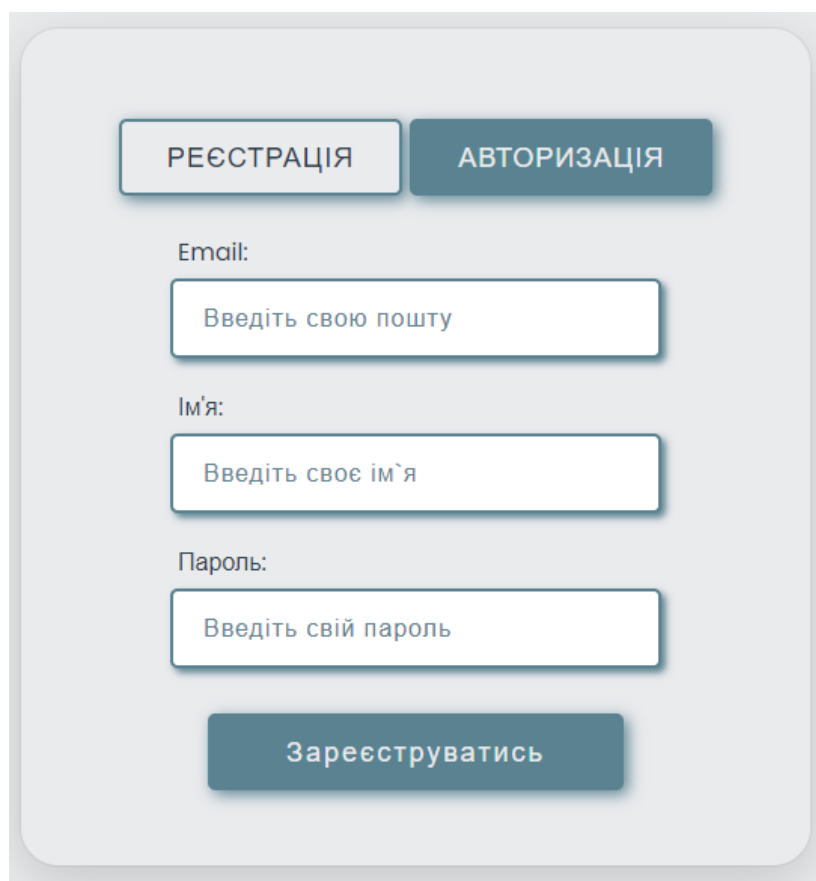
CSS-бібліотеки, такі як Bootstrap, Tailwind.css, Material UI та Ant Design, набувають все більшої популярності. Вони пропонують готові стилі, які можна швидко підключити до проекту. Основні переваги цих бібліотек:

- Швидкість розробки: Використання готових стилів значно скорочує час розробки.[19]
- Уніфікований вигляд: Забезпечують послідовний та професійний вигляд додатка завдяки використанню стандартизованих стилів.

Враховуючи всі переваги та недоліки вищезазначених підходів, було обрано класичний метод написання стилів SCSS. Хоча розробка за цим методом займає трохи більше часу, він забезпечує чіткий контроль над стилями та їх застосуванням. SCSS також пропонує потужні інструменти для організації стилів, такі як вкладеність, змінні та міксини, що значно полегшує підтримку коду та прискорює процес розробки.

3.3 Реалізація вебзастосунку на практиці

Першим кроком створимо форму реєстрації користувача та зробимо доступ до входу в особистий обліковий запис з вже розробленими планами.



The image shows a registration form interface. At the top, there are two buttons: 'РЕЄСТРАЦІЯ' (Registration) and 'АВТОРИЗАЦІЯ' (Authorization). Below these, there are three input fields with labels: 'Email:', 'Ім'я:' (Name), and 'Пароль:' (Password). Each input field contains a placeholder text: 'Введіть свою пошту' (Enter your email), 'Введіть своє ім'я' (Enter your name), and 'Введіть свій пароль' (Enter your password). At the bottom of the form, there is a large button labeled 'Зареєструватись' (Register).

Рисунок 3.5— Вікно для реєстрації користувачів

Джерело: запропоноване автором

На рисунку 3.5 зображена форма реєстрації користувача в додатку. Для створення облікового запису потрібно вказати електронну пошту, ім'я та придумати пароль. Перевірка введених даних здійснюється на сервері за допомогою express-validator. Під час валідації перевіряється, чи правильна електронна пошта, чи відповідає ім'я вимогам довжини від 2 до 10 символів, а пароль - від 3 до 12 символів.

```
[
  check('email', 'Incorrect email').isEmail(),
  check(
    'username',
    'Username must be longer than 2 and shorter than 10'
  ).isLength({ min: 2, max: 10 }),
  check(
    'password',
    'Password must be longer than 3 and shorter than 12'
  ).isLength({ min: 3, max: 12 })
]
```

Приклад коду 3.1 express-validator повертає набір даних, де ми за допомогою функції check перевіряємо значення полів форми.

Джерело: запропоноване автором



The image shows a login interface with a light gray background. At the top, there are two buttons: 'РЕЄСТРАЦІЯ' (Registration) and 'АВТОРИЗАЦІЯ' (Authorization). Below these, there are two input fields. The first is labeled 'Пошта:' (Email) and contains the placeholder text 'Введіть свою пошту' (Enter your email). The second is labeled 'Пароль:' (Password) and contains the placeholder text 'Введіть свій пароль' (Enter your password). At the bottom, there is a button labeled 'Увійти' (Login).

Рисунок 3.6— Вікно для входу вже зареєстрованих користувачів

Джерело: запропоноване автором

У наведеному прикладі коду на сервері відбувається авторизація користувача. Після отримання запиту з електронною поштою та паролем користувача, сервер перевіряє ці дані та, у разі успіху, повертає токен авторизації та інформацію про користувача. Давайте розглянемо цей процес детальніше.

Опис процесу авторизації

1. Перевірка пароля

```
const isValidPassword = bcrypt.compareSync(password, user.password);
if (!isValidPassword) {
  return res.status(400).json({ message: 'Invalid password' });
}
```

Приклад коду 3.2 Цей фрагмент коду порівнює введений користувачем пароль із хешованим паролем, що зберігається в базі даних.

Джерело: запропоноване автором

Для цього використовується бібліотека `bcrypt`. Якщо паролі не співпадають, сервер повертає помилку з повідомленням "Invalid password" та статусом 400.

2. Генерація токена авторизації

```
const token = jwt.sign({ id: user.id }, config.get('secretKey'), {
  expiresIn: '1h'
});
```

Приклад коду 3.3 Якщо пароль правильний, сервер створює токен авторизації за допомогою бібліотеки `jsonwebtoken` (JWT). Токен містить ідентифікатор користувача (`id`) і підписується секретним ключем, що зберігається в конфігураційному файлі. Токен налаштований на термін дії в 1 годину.

Джерело: запропоноване автором

3. Відправка відповіді

```
return res.json({
  token,
  user: {
    id: user.id,
    email: user.email,
    username: user.username,
    areas: user.areas
  }
});
```

Приклад коду 3.4 Сервер повертає відповідь, яка містить токен та інформацію про користувача: його ідентифікатор, електронну пошту, ім'я користувача та масив з планами користувача.

Джерело: запропоноване автором

Зв'язок з базою даних. Коли надходить запит на відповідний маршрут, сервер звертається до бази даних для пошуку користувача за його унікальним ідентифікатором (`_id`), який було присвоєно MongoDB під час реєстрації. Ця інформація використовується для перевірки пароля та створення відповіді з токеном.

Розглянемо приклад авторизації у вебзастосунку.

Припустимо, ми маємо користувача з такими даними в базі даних:

- `_id`: 60d21b4667d0d8992e610c85
- `email`: user@example.com
- `password`: хешований пароль
- `username`: user123
- `areas`: масив планів користувача

Коли користувач вводить свій email та пароль у форму авторизації, сервер:

1. Знаходить користувача за його `_id`.
2. Перевіряє, чи введений пароль співпадає з хешованим паролем в базі даних.

3. Якщо паролі співпадають, створює токен авторизації.
4. Повертає клієнту токен та інформацію про користувача у форматі JSON.

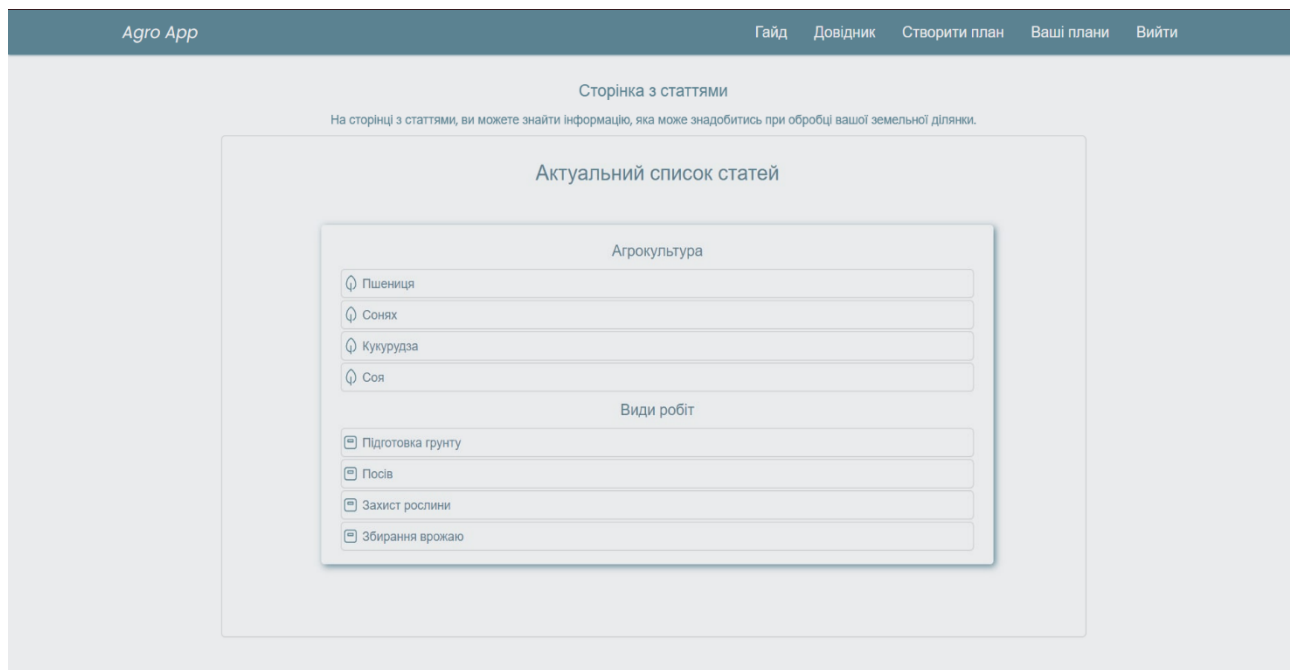


Рисунок 3.7— Головна сторінка вебзастосунку

Джерело: запропоноване автором

На рисунку 3.7 зображено головний екран додатку, який включає слайдер, що дозволяє користувачам ознайомитися з основними функціями додатку. Для реалізації слайдера було використано бібліотеку Swiper, яка забезпечує швидке і якісне створення слайдерів. Перемикання між слайдами здійснюється за допомогою жестів: достатньо затиснути ліву кнопку миші та потягнути вліво або вправо.

```
import { Swiper, Navigation, A11y } from 'swiper/react';
import 'swiper/swiper-bundle.css';

<Swiper
  modules={[Navigation, A11y]}
  spaceBetween={50}
  slidesPerView={1}
  onSwiper={(swiper) => console.log(swiper)}
  onSlideChange={() => console.log('slide change')}
>
  {/* Ваші слайди тут */}
</Swiper>
```

Приклад коду 3.5 – Налаштування слайдеру за допомогою Swiper

Джерело: запропоноване автором

У наведеному коді (приклад 3.5) параметр `modules` відповідає за підключення необхідних модулів для роботи слайдера, `slidesPerView` визначає кількість слайдів, що відображаються одночасно на екрані, `onSwiper` є зворотнім викликом, який спрацьовує при ініціалізації слайдера, а `onSlideChange` викликається при кожній зміні слайда.

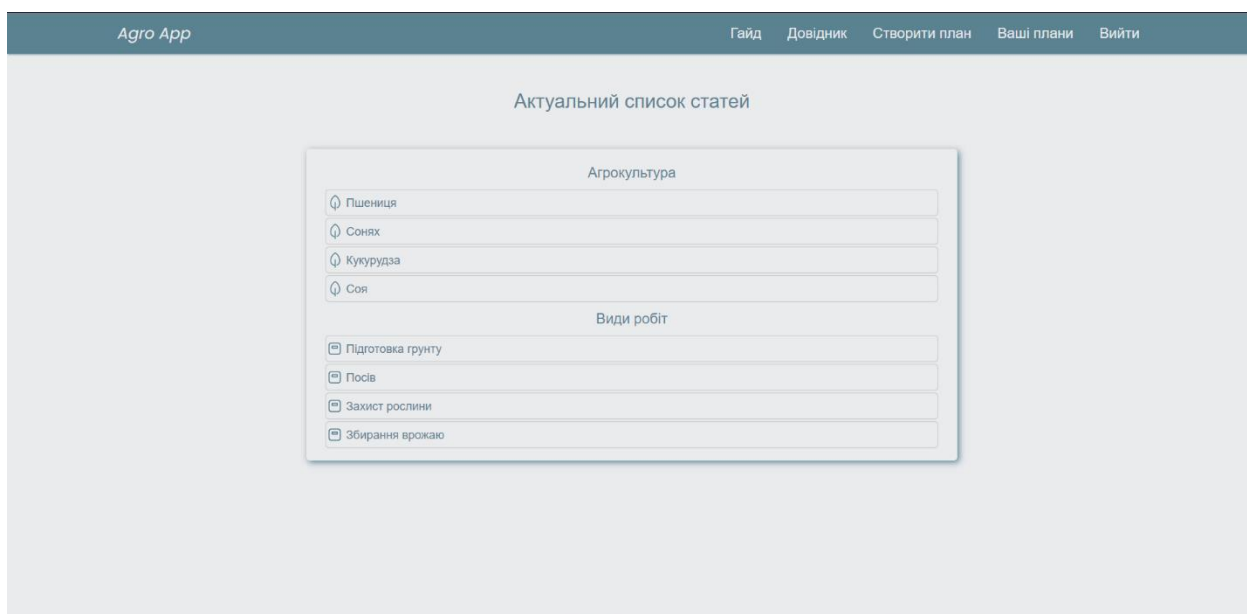


Рисунок 3.8— Демонструє доступні статті до кожної з культур

Джерело: запропоноване автором

У рисунку 3.8 показана сторінка, на якій відображені різні статті до кожної з культур. Користувач може вибрати будь-який пункт у списку, який його зацікавив, для переходу до перегляду самого тексту статті. Ці пункти списку, а також самі статті, зберігаються у базі даних.

```
{articles.map(title => (
  <li className='handbook__list-item' key={title._id}>
    <Link
      to='/home/handbook/article'
      onClick={(event) =>
        dispatch(
          getArticle(event.currentTarget.textContent)
        )
      }
    >
      <PlantIcon />
      {title.title}
    </Link>
  </li>
)})}
```

Приклад коду 3.4 – Створення масиву заголовків статей

Джерело: запропоноване автором

У прикладі коду 3.4 ми маємо масив заголовків статей, який перебирається методом `map`, щоб кожен заголовок відображався на екрані. При кліку на будь-який заголовок відбувається передача його вмісту в сторону `store`.

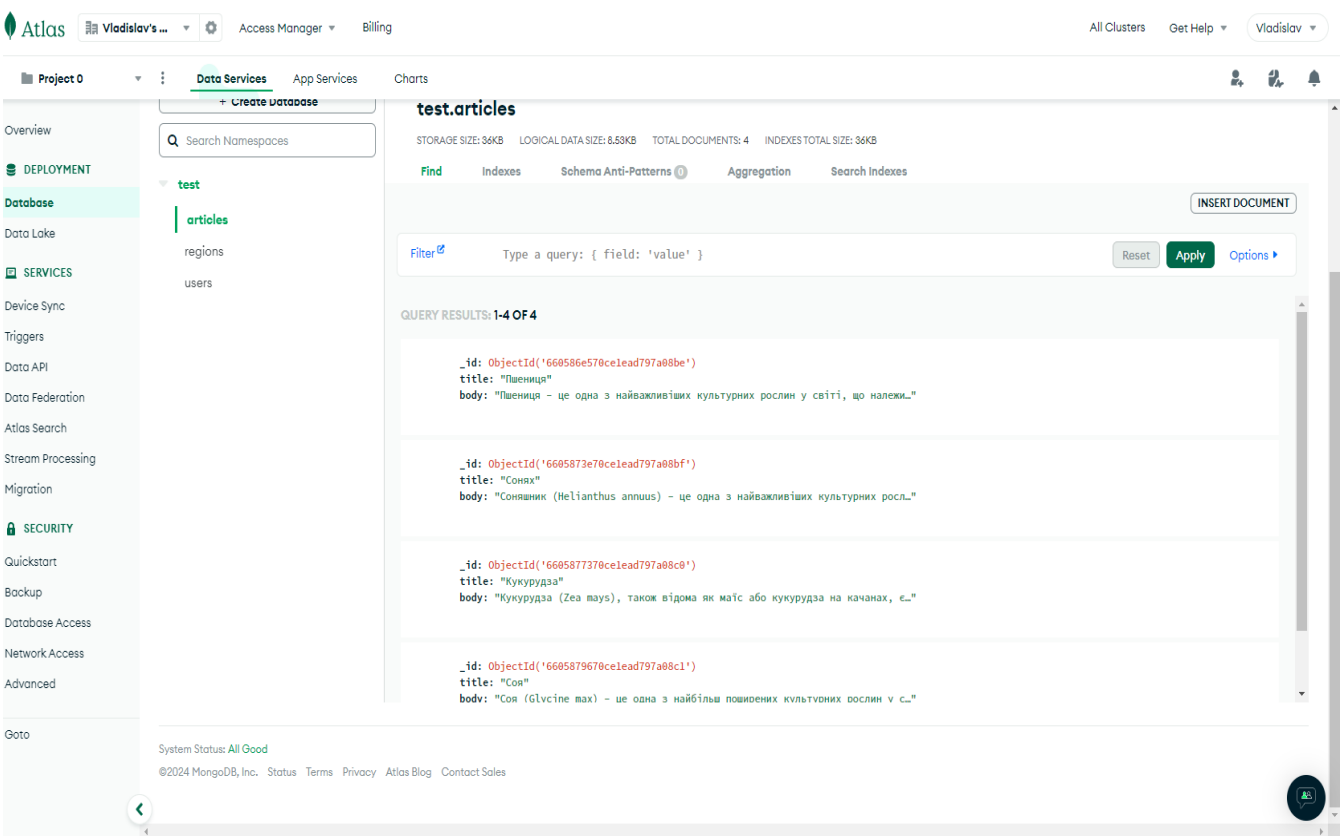


Рисунок 3.9—Приклад бази даних в якій зберігаються усі статті
Джерело: запропоноване автором

На рисунку 3.9 показано структуру документа в базі даних, призначеного для зберігання статей. Інформація організована у вигляді «масиву об’єктів». Кожен об’єкт містить унікальний `_id` для кожної статті, який генерується автоматично MongoDB, `title` – назву статті, що також використовується при створенні списку статей, та `body` – текст статті, який відображається при виборі конкретного пункту зі списку.

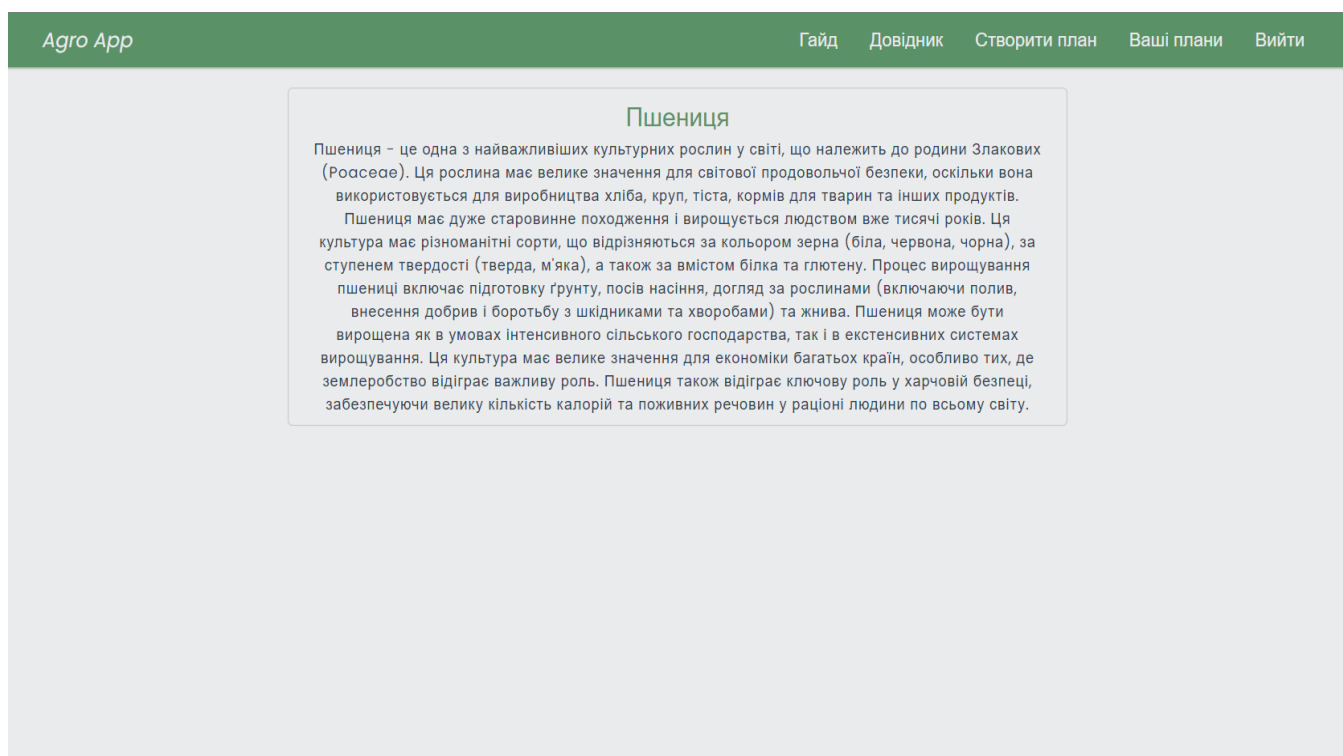


Рисунок 3.10—Приклад самої статті збереженої в базі даних та показаної на сторінці вебзастосунку

Джерело: запропоноване автором

На рисунку 3.10 показано, як реалізувати відображення окремої статті. Дані надходять з бази даних і зберігаються в сховищі за допомогою Redux Toolkit. У компоненті, який відповідає за відображення статті, ми використовуємо хуки, надані state-manager, такі як `useSelector()` та `useDispatch()`, щоб отримати необхідні дані та вивести їх на екран.

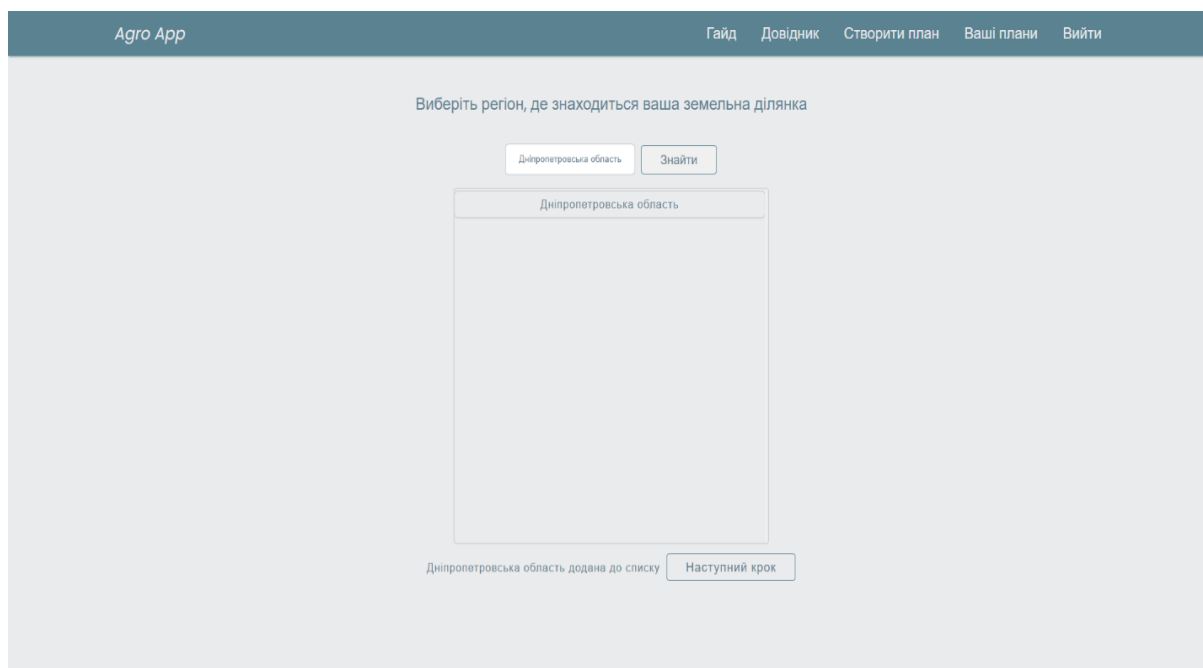
```
export function Article() {
  const articles = useAppSelector(state => state.articlesReducer.articles);
  const currentArticle = useAppSelector(state => state.articlesReducer.title);
  const article = articles.find(obj => obj.title === currentArticle);
  return (
    <div className='article'>
      <h3 className='article__title'>{article?.title}</h3>
      <p className='article__text'>{article?.body}</p>
    </div>
  );
}
```

Приклад коду 3.5 – приклад коду для відображення
Джерело: запропоноване автором

У прикладі коду 3.5 зі сховища витягуються всі статті та заголовок обраної статті. Потім за допомогою методу `find` знаходиться стаття з відповідним заголовком, і ця стаття відображається користувачу.

Рисунок 3.11— Кроки 1 по створенню свого плану робіт. Вибір регіону
розташування земельної ділянки

Джерело: запропоноване автором



На рисунку 3.11 зображена сторінка, де користувач може обрати регіон розташування своєї ділянки. На цій сторінці присутнє поле для пошуку регіону, а також список доступних регіонів.

Рисунок 3.12— Кроки 2 по створенню свого плану. Пошук потрібного регіону
Джерело: запропоноване автором

Рисунок 3.12 демонструє реалізацію функції пошуку регіону. Дані про регіони надходять з бази даних у вигляді масиву регіонів з яких він вітягає певний за запитом користувача. Для виконання пошуку в цьому масиві використовується метод `filter`. Якщо введене у пошукове поле значення збігається з назвою одного з регіонів у масиві, у результатах пошуку відображається лише цей регіон. Для того щоб при видаленні тексту з пошукового поля знову показувалися всі регіони, використовується умовне рендеринг за допомогою тернарного оператора.

```
const searchInputRegion = () => {
  if (!search) return;
  const searchedRegion = regions.filter(
    region => region.regionName === search
  );
  dispatch(searchRegion(searchedRegion));
};
```

Приклад коду 3.6 – Приклад коду для реалізації пошуку певного регіону
Джерело: запропоноване автором

За допомогою цього коду представленого в прикладі 3.6 ми отримуємо введення користувачем значення зі стану (state). Потім, за допомогою методу `filter`, фільтруємо масив з назвами регіонів і виводимо ті, що відповідають умові.

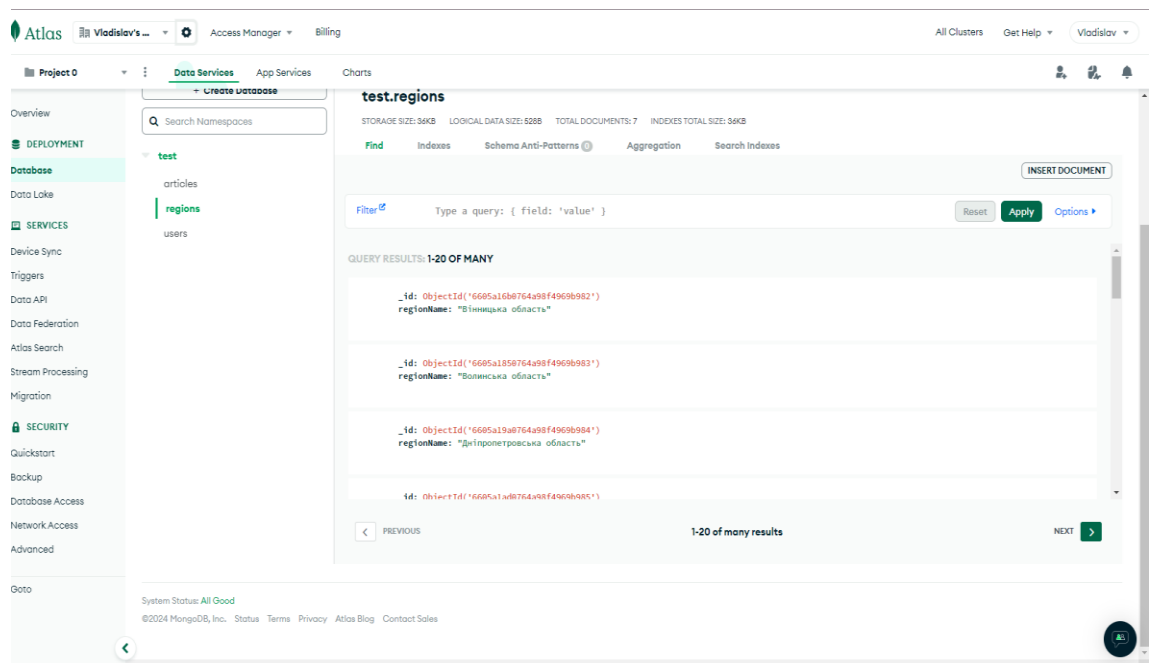


Рисунок 3.13 — Приклад вігляду бази даних з збереженням переліку доступних регіонів

Джерело: запропоноване автором

На рисунку 3.13 показано структуру бази даних для зберігання регіонів. Кожен запис містить поле `_id`, яке автоматично генерується базою даних, та `regionName`, що містить назву області.

Рисунок 3.14— Кроки 3 Вибір культури для розрахунку
Джерело: запропоноване автором

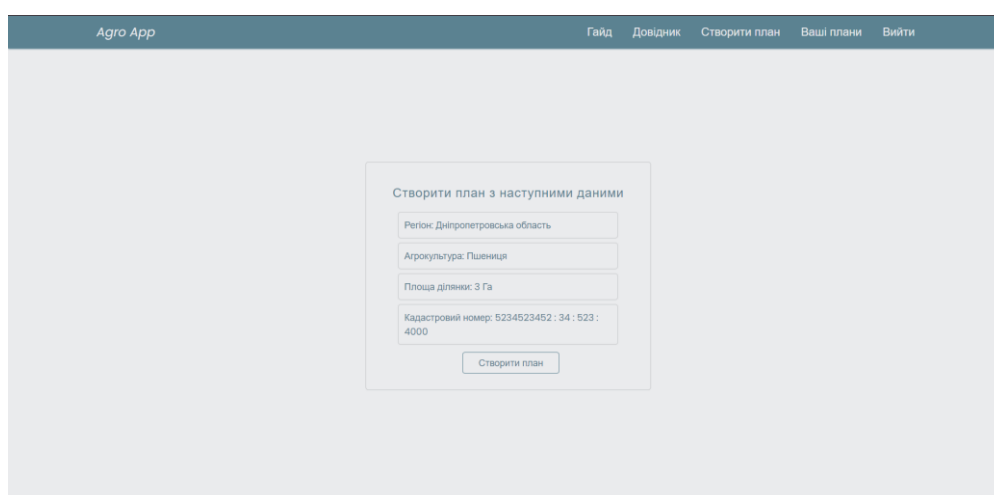
На рисунку 3.14 представлена сторінка вибору агрокультури. Залежно від обраної культури, змінюється загальна вартість обробки.

Рисунок 3.15—Крок 4 введення кадастрового номеру ділянки

Джерело: запропоноване автором

На рисунку 3.15 показана сторінка для введення кадастрового номера у форматі **XXXXXXXXXX : XX : XXX** та площі земельної ділянки.

Площа земельної ділянки безпосередньо впливає на формування вартості обробки. Після введення даних необхідно натиснути кнопку "Додати дані" перш ніж переходити до наступного кроку, щоб додана інформація зберіглась до загального сховища, яке надає **redux-toolkit**.



The screenshot shows a web application titled "Agro App" with a navigation bar containing links: "Гайд", "Довідник", "Створити план", "Ваші плани", and "Вийти". The main content area displays a form titled "Створити план з наступними даними" (Create plan with the following data). The form contains four input fields with the following values: "Регіон: Дніпропетровська область" (Region: Dnipropetrovsk region), "Агрокультура: Пшениця" (Crop: Wheat), "Площа ділянок: 3 Га" (Area: 3 Ha), and "Кадастровий номер: 5234523452 : 34 : 523 : 4000" (Cadastre number: 5234523452 : 34 : 523 : 4000). A "Створити план" (Create plan) button is located at the bottom of the form.

Рисунок 3.16 — Крок 5 Натискаємо «Наступний крок» та переходимо безпосередньо до створення плану

Джерело: запропоноване автором

На рисунку 3.16 зображена сторінка підтвердження даних. Користувач перевіряє введені дані на правильність вводу і, якщо всі данні були введені вірно, натискає "Створити план". Після цього відбувається обчислення за допомогою формул, збережених у **redux-store**.

Агро App Гайд Довідник Створити план Ваші плани Вийти

План

Регіон: Дніпропетровська область

Агрокультура: Пшениця

Площа ділянки: 3 Га

Кадастровий номер: 5234523452 : 34 : 523 : 4000

Деталі

Загальна вартість: 48864 грн.	Витрати пального: 6510 грн.	Насіння: 6000 грн.
Міндобрива: 9000 грн.	Сульфат амонію: 6000 грн.	Селітра аміачна: 12600 грн.
Гербіцид: 1500 грн.	Фунгіцид: 2400 грн.	Мікро мінерали: 210 грн.
Інсектицид: 900 грн.	Карбамід: 1224 грн.	Цинк: 0 грн.

Зберегти план

Рисунок 3.17 — Крок 6 Переходимо до готового плану робіт та готової таблиці вартості кожного кроку.

Джерело: запропоноване автором

На рисунку маємо виведення обчислених даних за зазначеними даними користувача. У верхній частині таблиці представлені дані про ділянку та її параметри, а в нижній - конкретні дані обчислення, необхідні користувачу для планування витрат. Таких як загальна вартість обробки землі та приблизні витрати на кожен етап обробки. Це відображення дозволяє клієнту зручніше керувати фінансами, які будуть виділені на обробку даного земельного участку.

Після ознайомлення з даними їх потрібно зберегти . Натиснувши кнопку "Зберегти план", користувачу буде сформовано об'єкт даних, що включає інформацію про ділянку, обчислені значення та `_id` користувача. Ці дані будуть надіслані на сервер через post-запит на відповідний маршрут, де відбудеться пошук клієнта у базі даних за його `_id` та додавання об'єкту до його масиву `areas`.

[15]

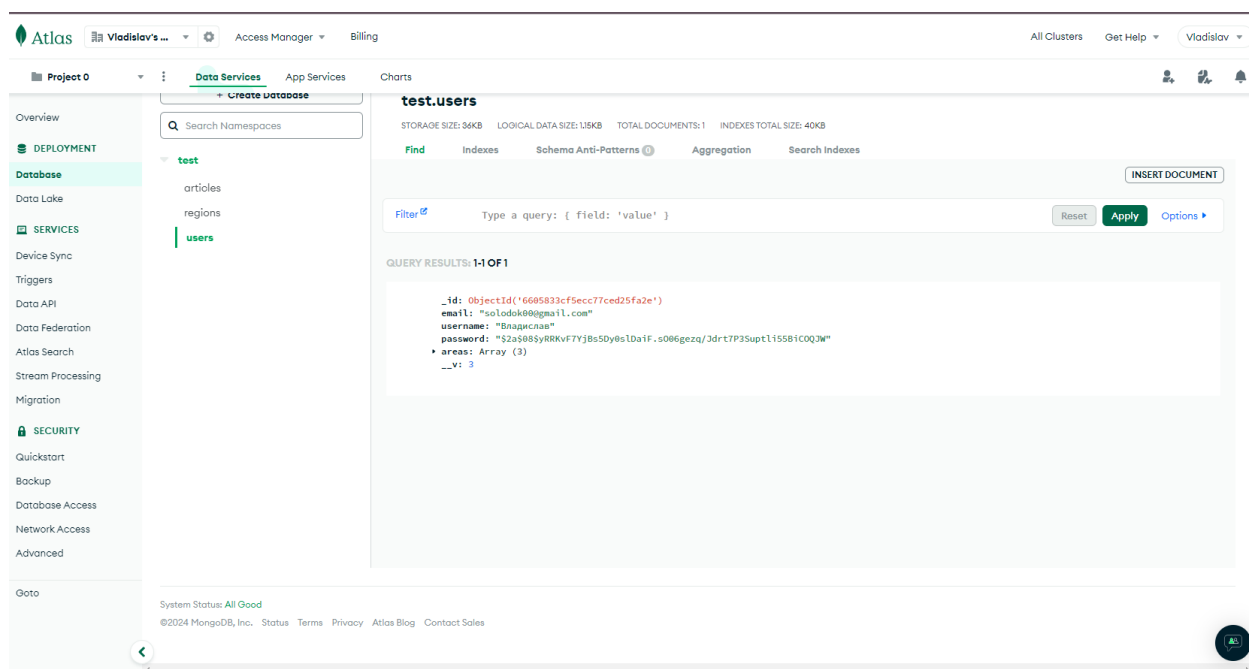


Рисунок 3.18— Приклад документу в базі даних по кожному з клієнтів
Джерело: запропоноване автором

На рисунку 3.18 показано структуру документа бази даних для кожного окремого користувача. Документ містить `_id` — унікальний ідентифікатор, що надається MongoDB, `email` — електронну пошту, введену користувачем при реєстрації, `username` — ім'я користувача, введене під час реєстрації, `password` — захешований пароль для підвищення безпеки акаунту, та `areas` — масив з планами робіт, створеними користувачем.

```

export const auth = () => {
  return async (dispatch: AppDispatch) => {
    try {
      const response = await axios.get(
        'http://localhost:5000/api/auth/auth',
        {
          headers: {
            Authorization: `Bearer ${localStorage.getItem('token')}`
          }
        }
      );
      dispatch(authUser(response.data.user));
      localStorage.setItem('token', response.data.token);
    } catch (e) {
      localStorage.removeItem('token');
    }
  }
};

```

Приклад коду 3.7 – Приклад функції оновлення даних

Джерело: запропоноване автором

У прикладі 3.7 представлено функцію для оновлення даних які закріплені за певним користувачем. Таких оновлень як створення нових планів або оновлення, перезавантаження сторінки. Викликається така функція командою `useEffect` але масив потрібно залишати порожнім.

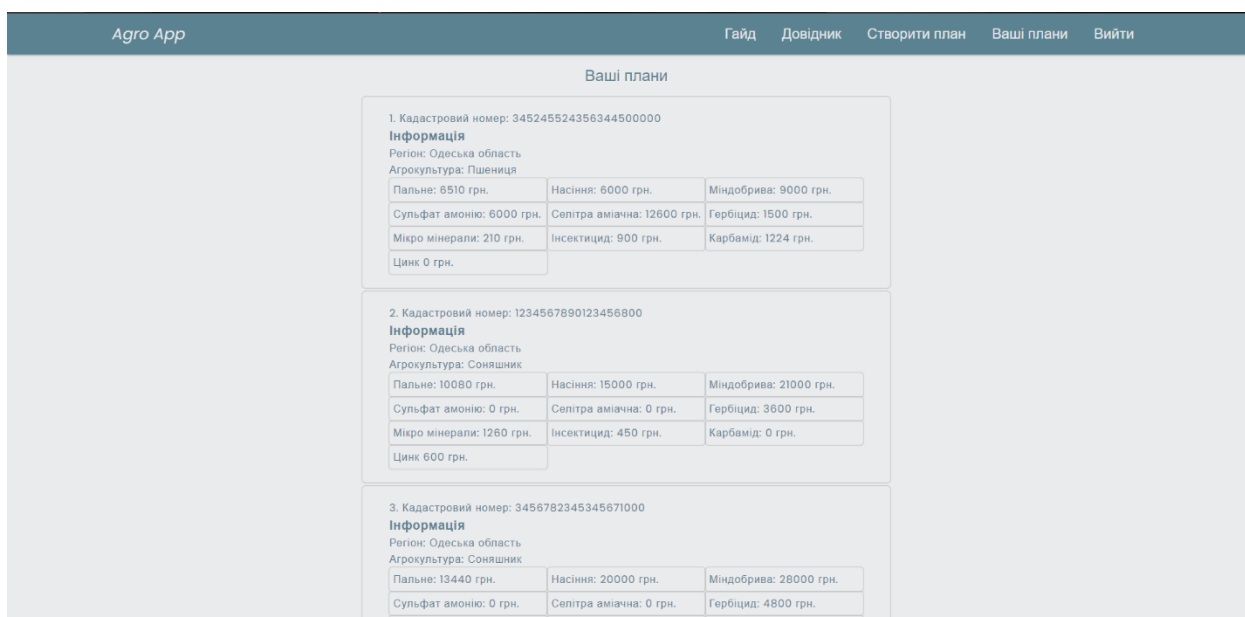


Рисунок 3.19—Вкладка з створеними планами

Джерело: запропоноване автором

На рисунку 3.19 показано, як відображаються дані про всі плани, створені користувачем. Ця інформація витягується з бази даних і показується на відповідній сторінці. Коли користувач створює новий план, дані одразу не оновлюються на клієнтській стороні, тому під час першого переходу на сторінку з планами надсилається GET-запит на сервер, щоб отримати актуальну інформацію. Додаток також має адаптивний дизайн для ноутбуків, планшетів та мобільних пристроїв.

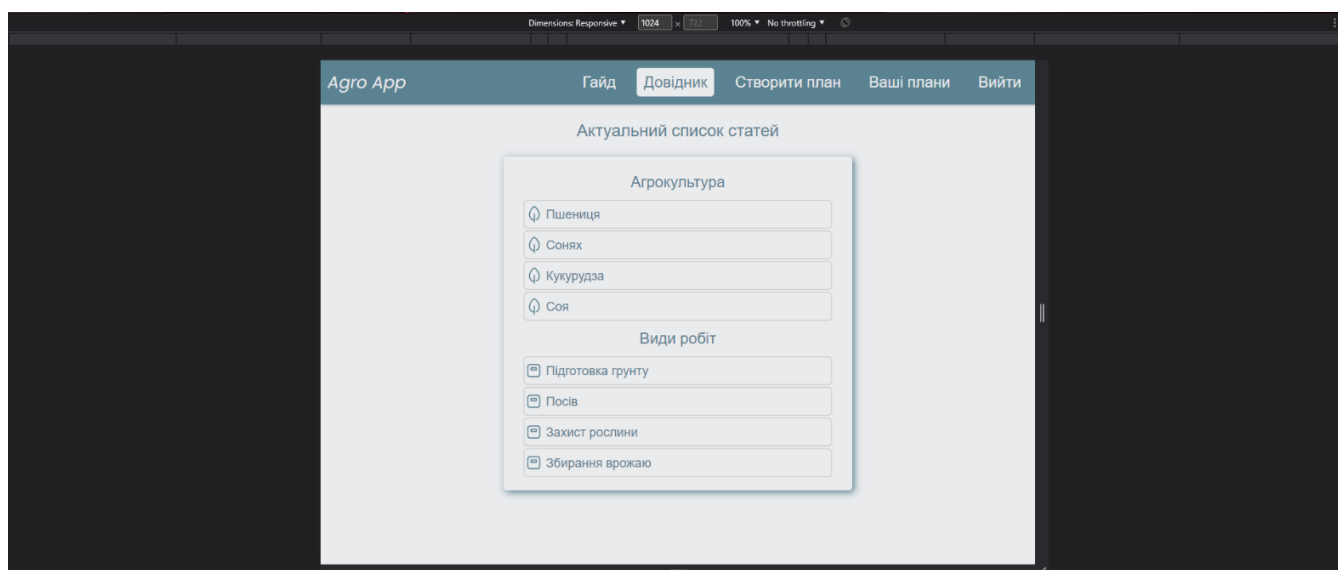


Рисунок 3.20 — Інтерфейс додатку на екрані ноутбука

Джерело: запропоноване автором

На рисунку 3.20 показано, як адаптивний інтерфейс додатку відображається на екрані ноутбука. Це досягнуто шляхом зменшення значення `max-width` при певній ширині екрану.

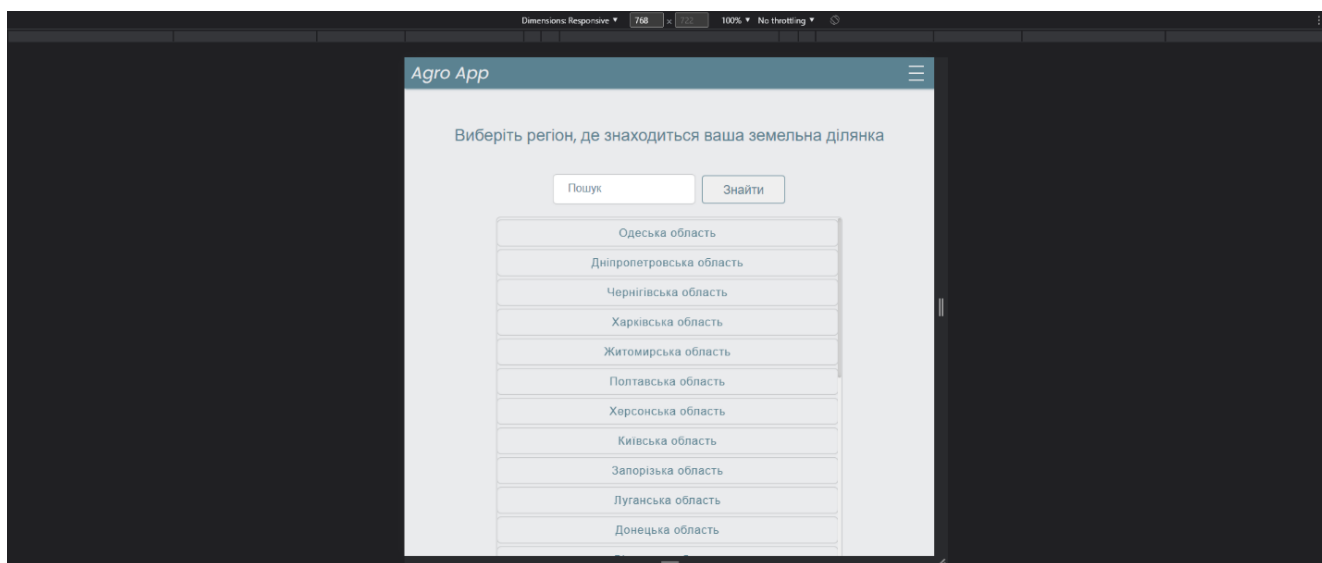


Рисунок 3.21 — Інтерфейс додатку на екрані планшета

Джерело: запропоноване автором

Цей рисунок демонструє, як додаток виглядає на екрані планшета. Деякі компоненти сітки `flex/grid` були змінені для коректного відображення.

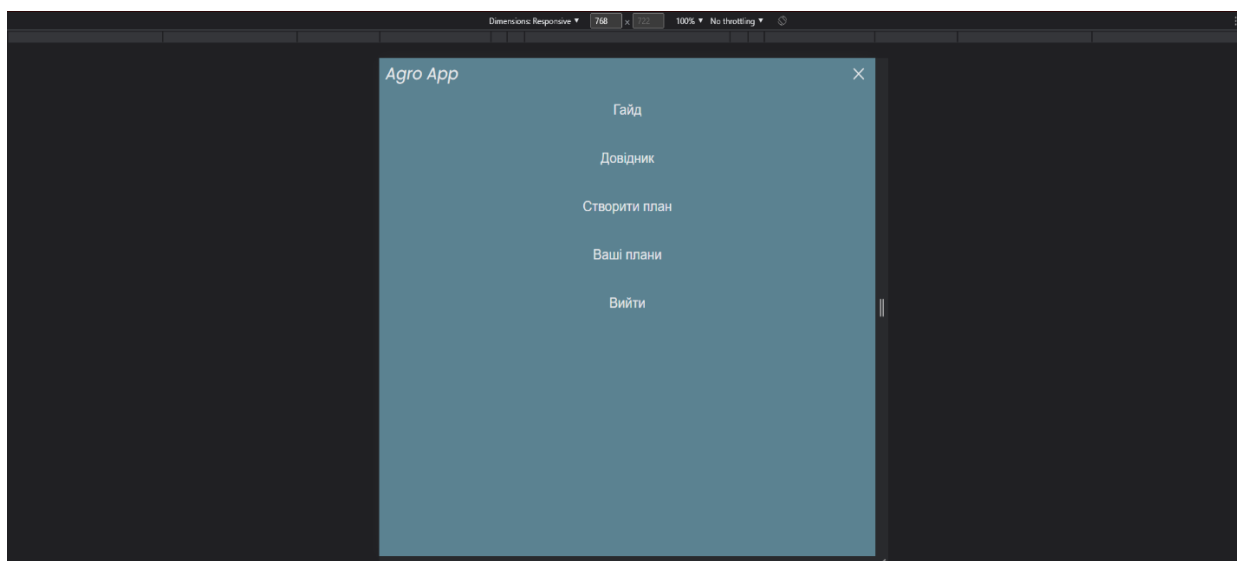


Рисунок 3.22 — Адаптивне меню

Джерело: запропоноване автором

На планшетному розмірі екрана потрібно було адаптувати головне меню додатку. На рисунку 3.22 показано, як реалізовано адаптивність: головне меню приховується за допомогою `display: none` і показується в новому форматі, змінивши `flex`-сітку. Також додано "меню-бургер", яке відкриває адаптоване меню при кліку.

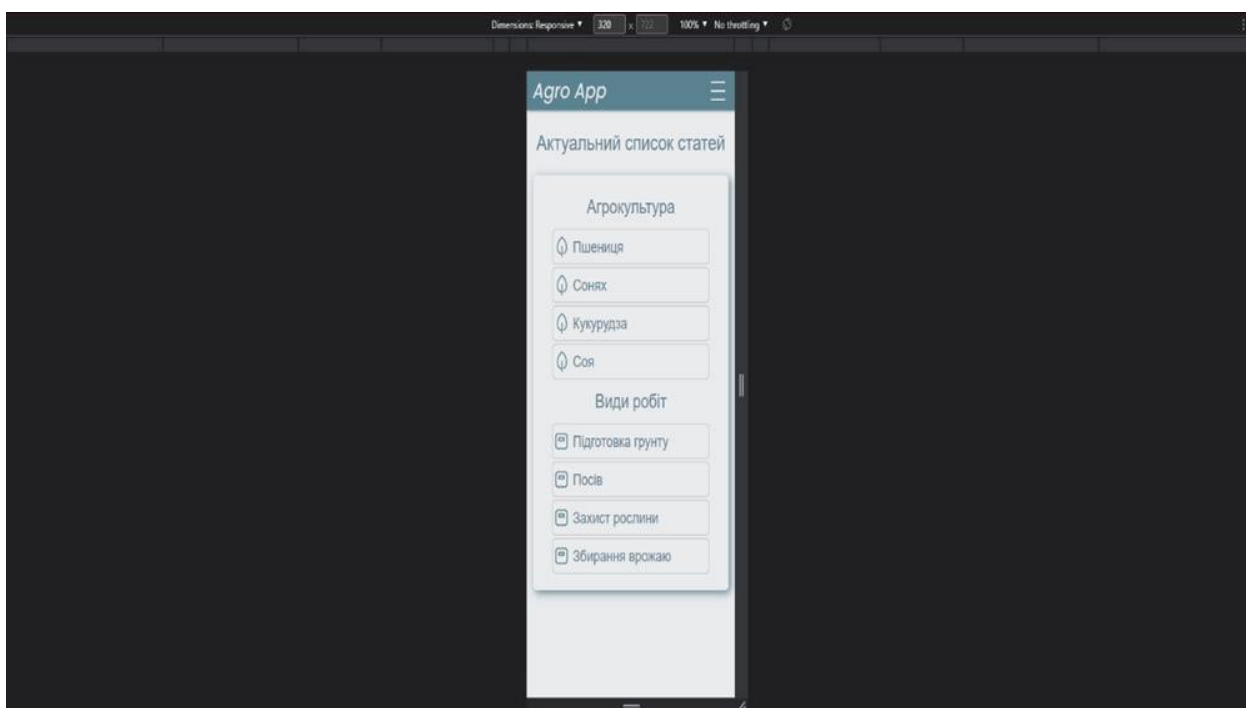


Рисунок 3.23 — Інтерфейс додатку на найменшому мобільному екрані

Джерело: запропоноване автором

На рисунку 3.23 зображено, як реалізовано адаптивний інтерфейс для найменшої ширини екрану, яку підтримує додаток. Це включає зміну `max-width`, зменшення розміру шрифтів, зміну сітки `flex/grid` та додавання адаптивного головного меню.

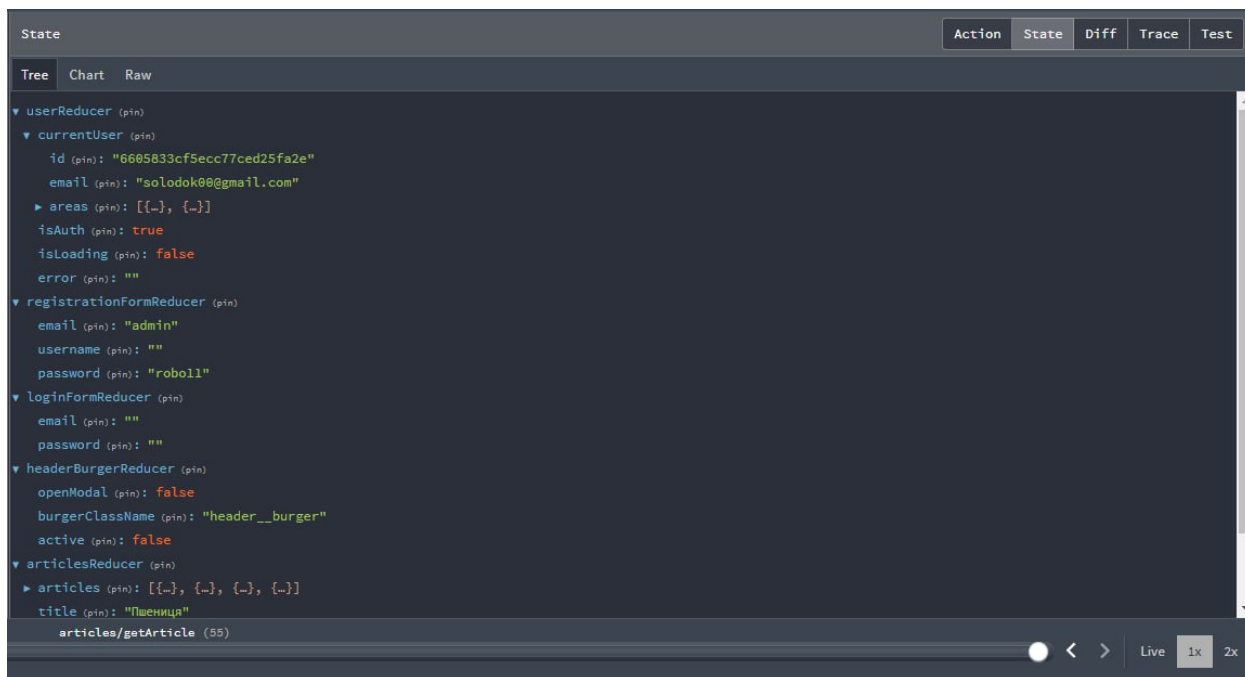


Рисунок 3.24 — Управління даними Redux Toolkit з клієнтської сторони
Джерело: запропоноване автором

На рисунку 3.21 представлено набір ред'юсерів, які застосовуються в додатку. Це дозволяє відокремити логіку обробки даних від компонентів, що відображають контент для користувача. Коли додаток надсилає запити на сервер, використовується функція `dispatch`, яка викликає відповідну дію (action). Ця дія запускає логіку, визначену в ред'юсерах.

```
export const userSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    authUser(state, action: PayloadAction<IUser>) {
      state.currentUser = action.payload;
      state.isAuth = true;
    },
    logoutUser(state) {
      localStorage.removeItem('token');
      state.currentUser.id = '';
      state.currentUser.email = '';
      state.currentUser.username = '';
      state.isAuth = false;
    },
    onLoading(state) {
      state.isLoading = true;
    },
    offLoading(state) {
      state.isLoading = false;
    }
  }
});
```

Приклад коду 3.8 –Управління даними користувача за допомогою slice

Джерело: запропоноване автором

ВИСНОВКИ

У ході виконання даної роботи було досягнуто кілька важливих результатів. Було проведено детальне дослідження різних інструментів та методів розробки вебзастосунків. Особлива увага приділялась вибору оптимального стеку технологій для створення ефективного та масштабованого додатку. Завдяки використанню сучасних технологій, таких як React, NodeJS, Express, MongoDB та TypeScript, вдалося створити додаток, який забезпечує точні розрахунки вартості обробки земельних ділянок.

В процесі розробки була продемонстрована важливість адаптивного дизайну, що дозволяє додатку коректно відображатися на різних пристроях, починаючи від великих екранів і закінчуючи мобільними телефонами. Використання Redux Toolkit для управління станом додатку допомогло структурувати код та забезпечити зручну роботу з даними.

Окрім технічних аспектів, було проведено аналіз існуючих рішень на ринку, що дозволило виявити сильні та слабкі сторони конкурентних продуктів і, відповідно, покращити власну розробку. Враховуючи всі отримані дані, можна стверджувати, що мета роботи досягнута: створено функціональний вебзастосунок, який відповідає сучасним вимогам і потребам користувачів.[13]

Ця робота також показала важливість інтеграції новітніх технологій у процес розробки та підкреслила значення безперервного вдосконалення знань і навичок для підтримки конкурентоспроможності на ринку.[2] В майбутньому є перспективи для подальшого розвитку та розширення функціоналу додатку, що відкриває нові можливості для користувачів та покращення їх досвіду.

Таким чином, виконана робота має значний практичний внесок у сферу розробки вебзастосунків та демонструє успішне застосування теоретичних знань на практиці.

СПИСОК ЛІТЕРАТУРИ

1. Антоненко, М. (2019). Розробка веб-додатків з використанням Node.js. Київ: Видавництво Київського університету. (date of access: 04.06.2024)
2. Бойко, І. (2020). Сучасні технології веб-розробки. Львів: Світ. (date of access: 05.06.2024)
3. Гавриленко, А. (2018). Проектування веб-додатків. Харків: Фоліо. (date of access: 08.06.2024)
4. Зайцев, О. (2021). React та його застосування у веб-розробці. Дніпро: Дніпрокнига. (date of access: 07.06.2024)
5. Коломієць, Ю. (2022). TypeScript для професіоналів. Одеса: Маяк. (date of access: 08.06.2024)
6. Kowalski, J. (2020). Web Application Development with MongoDB. Warsaw: Wydawnictwo Naukowe PWN. (date of access: 04.06.2024)
7. Martin, R. (2019). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Addison-Wesley. (date of access: 08.06.2024)
8. Mern, J. (2020). Building Full-Stack Applications with MERN. New York: Apress. (date of access: 07.06.2024)
9. Noring, J. (2021). Node.js Web Development. Birmingham: Packt Publishing. (date of access: 08.06.2024)
10. Roberts, M. (2018). Mastering React. Birmingham: Packt Publishing. (date of access: 06.06.2024)
11. Sholokh, A. (2019). Advanced TypeScript Programming Projects. Birmingham: Packt Publishing.
12. Smith, E. (2021). JavaScript: The Good Parts. Sebastopol: O'Reilly Media. (date of access: 08.06.2024)
13. Taylor, D. (2020). Building Scalable Web Applications. New York: McGraw-Hill. (date of access: 08.06.2024)
14. Thomas, M. (2022). JavaScript Patterns. Sebastopol: O'Reilly Media. (date of access: 10.06.2024)

15. Thompson, A. (2021). Express.js Guide. Birmingham: Packt Publishing. (date of access: 03.06.2024)
16. Ward, R. (2018). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. New York: Apress. (date of access: 08.06.2024)
17. White, T. (2019). MongoDB: The Definitive Guide. Sebastopol: O'Reilly Media. (date of access: 07.06.2024)
18. Wolf, D. (2019). Learning React: Modern Patterns for Developing React Apps. Sebastopol: O'Reilly Media. (date of access: 06.06.2024)
19. Young, B. (2021). Understanding ECMAScript 6. Sebastopol: O'Reilly Media. (date of access: 05.06.2024)
20. Стеблянюк В. В. Аналітичний огляд іт-технологій обліку витрат на виробництво сільсько-господарської продукції. Постановка задачі: матеріали всеукраїнської наукової конференції студентів та аспірантів, присвяченої міжнародному дню студента, 13-17 листопада 2023 р. Суми, 2023. С. 330. (date of access: 07.06.2024)
21. Стеблянюк В. В. Розробка вебзастосунку обліку витрат на виробництво продукції. Реалізація проєкту: матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ, 14-16 травня 2024 р. Суми: СНАУ. С. 424 (date of access: 10.06.2024)

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

НА РОЗРОБКУ вебзастосунку обліку витрат на виробництво сільськогосподарської продукції.

22. Призначення й мета створення вебзастосунку

1.1 Призначення веб застосунку

Розробка вебзастосунку обліку витрат на виробництво сільськогосподарської продукції.

1.2 Мета створення вебзастосунку

Метою створення веб застосунку полягає в автоматизації розрахунку витрат на виробництво сільськогосподарської продукції.[8]

1.3 Цільова аудиторія застосунку

Цільовою аудиторією вебзастосунку аграрні підприємства.[5]

23.Вимоги до вебзастосунку

2.1 Вимоги до застосунку в цілому

2.1.1 Вимоги до структури й функціонування вебзастосунку

- Реєстрація та авторизація користувачів.
- Генерація таблиць витрат.
- Збереження та відображення даних розрахунків.

2.1.2 Вимоги до персоналу

2.1.3 Вимоги до збереження вебзастосунку

Використання MongoDB забезпечує швидку та просту роботу за базою даних та управління інформацією, збереженням та обробкою даних. [1]

2.1.4 Вимоги до розмежування доступу

Доступ до вебзастосунку є відкритим та загально доступним.[9]

2.2 Структура вебзастосунку

2.2.1 Навігація

Сторінка сервісу містить загальні данні що до певних культур інструкцію по використанню та особисті плани.

2.2.2 Наповнення вебзастосунку

Включає сторінки для введення, перегляду, загальна інформація, видалення даних, а також формування звітів на основі отриманих даних.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Забезпечення простого, інтуїтивного та зручного користувацького інтерфейсу для введення ,створення та перегляду сформованих звітів.

2.3.2 Функціональні вимоги

- Реєстрація та авторизація користувачів.
- Генерація таблиць витрат.
- Збереження та відображення даних розрахунків.

2.3.3 Системні вимоги

Системні вимоги повинні відповідати вимогам користувачів:

- Наявність авторизації та реєстрації.
- Внесення даних для розрахунку.
- Перегляд та збереження звітів

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація вебзастосунку відбувається з використанням:

- Node.js
- Express.js
- MongoDB
- React.js
- CSS-in-JS (Styled-Components)
- JS

2.4.2 Вимоги до лінгвістичного забезпечення

Для розробки вебзастосунку використовуються такі програмні компоненти та технології:

Node.js і Express.js на серверній частині дозволяють створити швидкий та гнучкий бекенд,

MongoDB забезпечує надійне зберігання даних.

На клієнтській частині, використання React.js та CSS-in-JS дозволяє створити сучасний і інтерактивний інтерфейс користувача.

3. Склад і зміст робіт зі створення застосунку

Докладний опис етапів роботи зі створення вебсервісу наведено в таблиці А.1.

№	Склад і зміст робіт	Строк розробки
1	Постановка загальної задачі	1 днів
2	Формулювання задач та вимог що до функціоналу, які повинен вирішити вебзастосунок.	1 днів

3	Підготовка наповнення вебзастосунку: текст сайту, таблиці і т.д.	1 днів
4	Розробка технічного завдання на створення вебзастосунку	3 днів

Продовження таблиці А.1

5	Узгодження технічного завдання на створення вебзастосунку	1 днів
6	Розробка програмного коду і зовнішнього оформлення вебзастосунку	15 днів
7	Розробка оптимальної та спрощеної системи навігації вебзастосунку	5 днів
8	Розробка оптимальної системи навігації вебзастосунку	5 днів
9	Вибір оптимального типу гіперпосилань вебзастосунку (кнопок системи управління)	3 днів
10	Вибір основної палітри кольорів вебзастосунку	2 днів
11	Вибір типу і розміру основних шрифтів вебзастосунку	1 днів
12	Вибір засобу реалізації БД	2 днів
13	Проектування БД	5 днів
14	Розробка прототипу БД	5 днів
15	Наповнення БД довідковою інформацією	5 днів
16	Створення основних запитів БД	5 днів
17	Підключення БД до сайту	2 днів
18	Наповнення вебзастосунку додатковим контентом	15 днів
19	Поширення вебзастосунка в мережу	5 днів
20	Тестування вебзастосунку	3 днів

2 1	Виявлення і усунення помилок	2 днів
2 2	Перевірка вебзастосунку замовником	2 днів
2 3	Внесення змін за запитом замовника	2 днів
2 4	Запис і перевірка відкритого програмного коду вебзастосунку замовнику	2 днів
2 5	Отримання відкритого програмного коду вебзастосунку замовником	1 днів
	Загальна тривалість робіт	96 днів

Продовження таблиці А.1

Джерело: запропоноване автором

4. Вимоги до складу й змісту робіт із введення вебзастосунку в експлуатацію

Для забезпечення доступу фермерів до вебзастосунку, його необхідно розмістити в мережі Інтернет. Це передбачає придбання доменного імені, яке буде використовуватися як адреса вебзастосунку, та оренду серверного простору на хостингу для його розміщення. Після цього вебзастосунок буде завантажено на хостинг разом з початковою базою даних. Після перенесення даних база буде поповнюватися новою інформацією та проходити необхідні доопрацювання для забезпечення повноцінної та ефективної роботи сервісу.

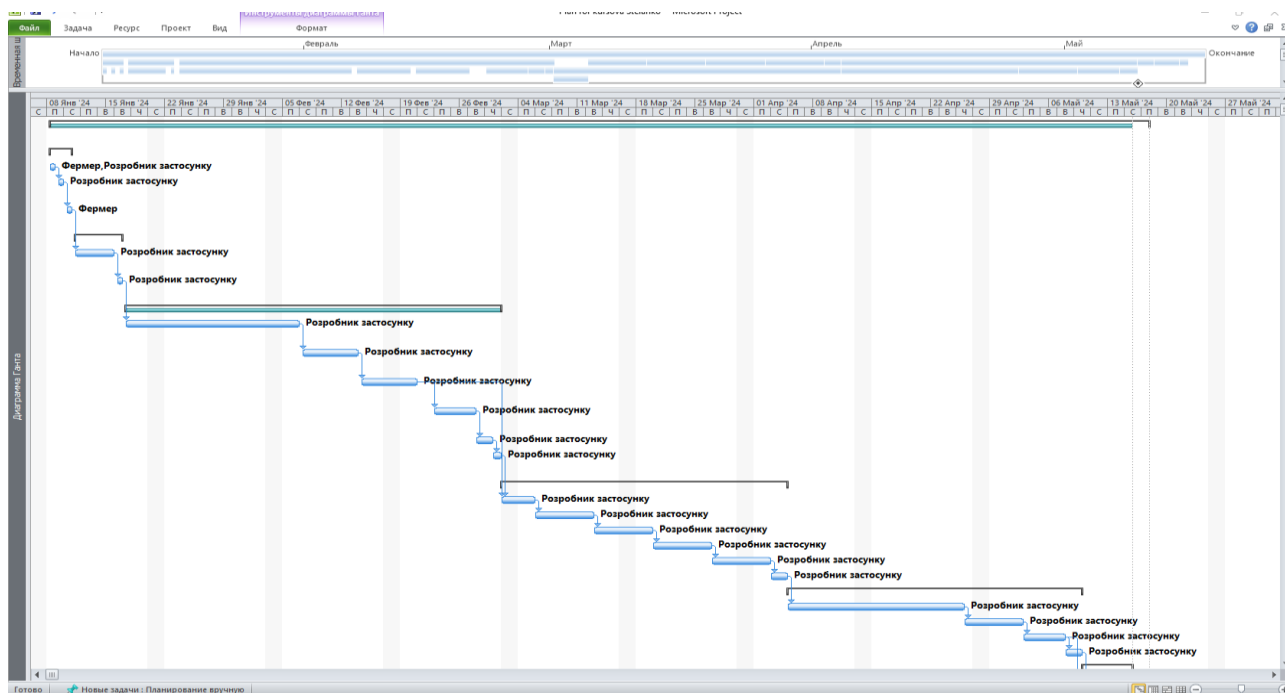
ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Визначення старту та завершення завдань, розподіл робочого навантаження, встановлення контрольних точок, моніторинг виконання в порівнянні з початковим планом, корекція графіка за необхідності, оптимізація використання людських і матеріальних ресурсів, а також інформування команди про статус проекту, заплановані завдання та терміни їх виконання — все це є критично важливим для успішної роботи, і з усім цим допомагає діаграма Ганта.

Основними задачами є:

1. Тривалість проекту не повинна перевищувати 5 місяців.
2. У календарі робочого часу внесено такі зміни: на останньому етапі кожна парна субота вважається робочим днем.
3. Під час перегляду діаграми Ганта необхідно виконати складне сортування за початковою датою у порядку зростання.
4. Діаграма Ганта проекту створення вебсервісу наведена на рисунку Б.1.



Рисунку Б.1 –Діаграма Ганта

Джерело: запропоноване автором

Оцінка ризиків в ІТ-проєкті зі створення вебзастосунку обліку витрат на виробництво сільськогосподарської продукції є надзвичайно важливою для забезпечення успішного виконання проєкту. Цей процес включає в себе визначення, аналіз та управління ризиками, які можуть вплинути на досягнення цілей проєкту. Основними складовими оцінки ризиків є опис ризику, ймовірність його виникнення, вплив на проєкт, стратегії управління та відповідальні за їх реалізацію особи.[6]

Таблиця Б.1 – Оцінка ризиків ІТ-проєкту

Ризики	Імовірність виникнення	Вплив на проєкт	Стратегії управління	Відповідальні особи
Втрата даних	Низька	Висока	Регулярна перевірка резервне копіювання даних, застосування шифрування	Розробник застосунку
Відмова обладнання	Низька	Середня	Планування та впровадження запасних/резервних пристроїв, регулярне технічне обслуговування	Розробник застосунку
Відмова сервера через перевищення навантаження	Середня	Високий	Масштабування серверної інфраструктури, впровадження механізмів автоматичного моніторингу	Розробник застосунку
Затримка в розробці через зміни вимог	Середня	Середній	Узгодження докладних вимог до функціоналу, впровадження гнучких методологій розробки	Фермер
Проблеми з безпекою	Середня	Високий	Регулярні аудити безпеки, впровадження стандартів безпеки, навчання персоналу з питань безпеки	Розробник застосунку

Зміна нормати в них вимог	Низька	Середній	Моніторинг законодавства, залучення юридичних консультантів, адаптація до нових вимог	Фермер
Збої в ПЗ	Низька	Низька	Регулярна перевірка ПЗ, тестування, планування альтернативних варіантів	Розробник застосунку

Продовження таблиці Б.1

Джерело: запропоноване автором

Втрата даних може статися через кілька причин, включаючи відсутність або недостатнє резервне копіювання, кібератаки або шкідливе програмне забезпечення, помилки співробітників (ненавмисне видалення або зміна даних), апаратні збої, такі як вихід з ладу жорстких дисків, а також природні катастрофи, такі як пожежі чи повені. Також до цього переліку можна додати людські фактори, як-от неправильно налаштоване обладнання або некоректне управління даними.

Апаратні збої зазвичай спричинені фізичним зносом компонентів, неналежним технічним обслуговуванням, використанням обладнання, яке не відповідає вимогам проєкту, перепадами напруги або іншими електричними проблемами, а також перегрівом через недостатнє охолодження. Крім того, такі збої можуть бути результатом некоректної інсталяції або недостатньо професійного монтажу обладнання.

Серверні проблеми через перевантаження можуть виникати через неправильну оцінку навантаження під час планування, раптовий сплеск активності користувачів, неоптимізований код або налаштування серверів, недостатні ресурси (процесор, оперативна пам'ять, пропускна здатність мережі) або відсутність механізмів масштабування. Також важливо враховувати можливі недоліки в системі балансування навантаження та невідповідність до пікових періодів.

Затримка у розробці через зміни вимог часто виникає через нечітке визначення початкових вимог, зміну бізнес-цілей або стратегій, зовнішні

чинники, що впливають на вимоги (наприклад, нові закони або регулювання), недостатню комунікацію між зацікавленими сторонами, а також появу нових конкурентних продуктів або технологічних змін, які потребують адаптації. Часто такі затримки можуть бути спричинені непередбаченими змінами в ринкових умовах або новими запитами клієнтів.

Проблеми з безпекою можуть виникнути через недостатню увагу до безпеки на етапі проектування, відсутність регулярних аудитів безпеки, використання застарілого або вразливого програмного забезпечення, людський фактор (соціальна інженерія, недостатня обізнаність співробітників) та відсутність чітких політик безпеки. Крім того, постійно еволюціонуючі методи атак зловмисників потребують регулярного оновлення та удосконалення захисних заходів.[7]

Зміна нормативних вимог може бути спричинена оновленням законодавства або регулюючих актів, новими галузевими стандартами або вимогами до сертифікації, політичними або економічними змінами, що призводять до нових регулювань, а також вимогами щодо захисту даних і конфіденційності (наприклад, GDPR). Такі зміни можуть також включати екологічні норми або вимоги до корпоративної соціальної відповідальності.

Збої в програмному забезпеченні можуть виникнути через використання ненадійних або недостатньо протестованих компонентів, відсутність регулярних оновлень та виправлень, конфлікти між різними компонентами або версіями програмного забезпечення, недостатнє тестування нових релізів, а також помилки програмування. [4] До цього також можуть додаватися проблеми, пов'язані з недостатньою документацією або відсутністю кваліфікованих фахівців, здатних оперативно виявляти та усувати неполадки.

ДОДАТОК В
КОПІЇ ПУБЛІКАЦІЙ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ
ВСЕУКРАЇНСЬКОЇ НАУКОВОЇ
КОНФЕРЕНЦІЇ СТУДЕНТІВ
ТА АСПІРАНТІВ, ПРИСВЯЧЕНОЇ
МІЖНАРОДНОМУ ДНЮ СТУДЕНТА

(13 - 17 листопада 2023 р., м. Суми)

Wang Baichao. MANAGEMENT CRISIS IN AGRICULTURAL COMPANIES IN CURRENT SITUATIONS	414
Wang Ruijie. HUMAN RESOURCES MANAGEMENT OPTIMIZATION AND INNOVATION BASED ON ARTIFICIAL INTELLIGENCE	415
Yu Hang. RESEARCH ON THE INTRODUCTION OF MODERN MANAGEMENT TECHNOLOGY INTO ENTERPRISES	416
Zhou Kunming. PECULIARITIES OF MANAGING HIGH-TECH ENTERPRISES IN A TRANSPARENT ECONOMY	417
Бусенко І.В. АНАЛІЗ ОСНОВНИХ РІЗНОВИДІВ SQL-ІН'ЄКЦІЙ, АНАЛІТИКА ЗАСТОСУВАННЯ ТА СПОСОБИ ЗАХИСТУ ВІД НИХ	418
Белохонь П.А. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ПІДТРИМКИ РОБОТИ HR-МЕНЕДЖЕРА	419
Стеблянко В. В. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО ПРОДУКЦІЇ. РЕАЛІЗАЦІЯ ПРОЄКТУ	420
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. ПОСТАНОВКА ЗАДАЧІ	421
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. РЕАЛІЗАЦІЯ ПРОЄКТУ	422
Биченко Є.В. РОЗРОБКА ПРОТОТИПУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ДІЯЛЬНІСТЮ ПІДПРИЄМСТВА	423
Корецький В.А. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ МОНИТОРИНГУ ІНТЕРНЕТ-ТРАФІКУ	424
Лалченко І. В. ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ	425
Хоменко Р.М. ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ЧАТ-БОТА ДЛЯ ЗРУЧНИХ ТА ЕФЕКТИВНИХ ОНЛАЙН-ЗАМОВЛЕНЬ ЕЛЕКТРОНІКИ	426
Блаженко Д. О. ПРОБЛЕМИ ПРОЄКТУВАННЯ БАЗИ ДАНИХ ПІДСИСТЕМИ ВЕДЕННЯ ЗАМОВЛЕНЬ ДЛЯ ВИГОТОВЛЕННЯ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ	427
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	428
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	429
Руденко О. А. МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ	430
Сіденко А. С. ПРОБЛЕМИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ОБЛІКУ ПРИРОДООХОРОННИХ ЗАХОДІВ	431
Терещенко С.С. АВТОМАТИЗАЦІЯ ОБЛІКОВИХ ПРОЦЕСІВ: КЛЮЧ ДО ЕФЕКТИВНОГО УПРАВЛІННЯ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	432
Михайлюк О.С. ОГРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОЇ ПІДСИСТЕМИ ПІДТРИМКИ ОБЛІКОВИХ ПРОЦЕСІВ РЕГІОНАЛЬНОГО ІНТЕРНЕТ-ПРОВАЙДЕРА	433
Олійник А.В. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО-МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	434
Чубун М.П. АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СУЧАСНОГО ВЕБ-САЙТУ	435
Дубінчин В. С. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	436
Савжаків А. А. НОВИЙ ПІДХІД РОЗРОБКИ ДИЗАЙНУ ВЕБ-ЗАСТОСУНКІВ – MATERIAL 3	437
Федірко Д.Д. ПРОБЛЕМИ ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКІВ ОЦІНКИ ТАРИФНИХ ПРОПОЗИЦІЙ	438
МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	439
Федірко Д.Д. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	440
Маслак О. М. ДЕРЖАВНЕ РЕГУЛЮВАННЯ ПРОЦЕСІВ АГРОПРОМИСЛОВОГО ВИРОБНИЦТВА НА РЕГІОНАЛЬНОМУ РІВНІ (НА ПРИКЛАДІ СУМСЬКОЇ ОБЛАСТІ)	441
Горбань С.В. ІМІДЖ ПІДПРИЄМСТВА: ВПЛИВ, ФОРМУВАННЯ ТА ОЦІНКА	442
Заболотна А.І. НОРМАТИВНО-ПРАВОВЕ ЗАБЕЗПЕЧЕННЯ ЕКОНОМІЧНОГО РОЗВИТКУ ДРУЖІВСЬКОЇ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	443
Гулиніна О.Ю. ШЛЯХИ ОПТИМІЗАЦІЇ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ УПРАВЛІННЯ ЗАКЛАДУ ОХОРОНИ ЗДОРОВ'Я	444
Сущенко В. О. ПРОЦЕС РОЗВИТКУ ПЕРСОНАЛУ, ЯК ЧИННИК КОНКУРЕНТОЗДАТНОСТІ ПІДПРИЄМСТВА В РИНКОВИХ УМОВАХ	445
Каплун В.М. УПРАВЛІННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ ІЗ ЗАСТОСУВАННЯ ПРОГРАМНОГО КОНТРОЛЮ І МОНИТОРИНГУ	446
Бондар А.В. ТЕОРЕТИКО-МЕТОДИЧНІ ПІДХОДИ ДО ВПРОВАДЖЕННЯ ТА ОЦІНКИ СТРАТЕГІЧНОГО УПРАВЛІННЯ РОЗВИТКОМ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ	447
Романенко Т.А. СУЧАСНІ НАПРЯМИ ІНТЕНСИФІКАЦІЇ ВИРОБНИЦТВА ПРОДУКЦІЇ РОСЛИННИЦТВА В КОНТЕКСТІ ПРОДОВОЛЬНОЇ БЕЗПЕКИ	448
Мішенін Є. В., Ярова І. Є. ОРГАНІЗАЦІЙНО ЕКОНОМІЧНИЙ МЕХАНІЗМ ЗАБЕЗПЕЧЕННЯ СТАЛОГО ПРОСТОРОВОГО РОЗВИТКУ ЛІСОВОГО ГОСПОДАРСТВА	449
Hu Guannan CRISIS MANAGEMENT THEORY	450
Zhang Liming HOW DOES BYD COMPANY IMPROVE ITS COMPETITIVENESS?	451
Zhao Jing THE INFLUENCE OF CORPORATE CULTURE ON EMPLOYEE SATISFACTION	452
Cui Xiaoping DEVELOPMENT PATH OF ENTERPRISE MANAGEMENT IN THE ERA OF ARTIFICIAL INTELLIGENCE	453

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО ПРОДУКЦІЇ. РЕАЛІЗАЦІЯ ПРОЄКТУ

Стеблянюк В. В. студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.т.н., доц. С.В. Агаджанова
Сумський НАУ

Вступ. Фермерські господарства постійно потребують ефективного управління витратами на обробку землі. Веб-додаток може автоматизувати процес прогнозування та розрахунку витрат, допомагаючи фермерам приймати більш обґрунтовані рішення.

Мета розробки: створити веб-застосунок, який спростить і автоматизує процес прогнозування витрат, щоб фермери могли краще керувати ресурсами. До додаткових завдань входить покращення планування та надання інтуїтивно зрозумілого інтерфейсу для аналізу даних.

Технології: для фронкенду використовуються сучасні фреймворки, такі як React.js, Angular та Vue.js [1-4]. Кожен із них має свої переваги:

- React.js – швидкий та має велику спільноту розробників.
- Angular – пропонує вбудовані інструменти для складних веб-застосунків із чіткою архітектурою.
- Vue.js – простий у використанні та гнучкий, що особливо зручно для невеликих команд розробників.

Для беккенду рекомендується Node.js через його швидкість обробки запитів, а як базу даних можна обрати MongoDB або PostgreSQL, залежно від потреб у зберіганні даних.

Основні функції:

1. Прогнозування витрат: Користувач вводить дані про земельну ділянку, а веб-додаток генерує прогнозований розрахунок витрат на її обробку.

2. Реєстрація користувача та зберігання даних: Додаток дозволяє користувачам реєструватися і зберігати інформацію про земельні ділянки та їх витрати.

3. Візуалізація даних: Інтерфейс відображає дані у вигляді графіків та діаграм, що допомагає фермерам візуально оцінювати витрати та загальну інформацію про культури.

Переваги для фермерів: Автоматизація розрахунків економить час і зменшує ризик помилок.

Крім того, веб-додаток сприяє кращому плануванню ресурсів та забезпечує зручний інтерфейс для користувачів з різним рівнем технічної підготовки.

Методологія розробки: Використання Agile-підходу дозволяє гнучко реагувати на зміни у вимогах і забезпечує тісну взаємодію з кінцевими користувачами. Розробка включає регулярні ітерації та тестування для підтримки високої якості продукту.

Висновок: Веб-застосунок для розрахунку витрат на обробку земельних ділянок може значно покращити ефективність управління у фермерських господарствах.

Використання сучасних технологій, таких як React.js, Angular та Vue.js, дозволяє створити продуктивний, гнучкий та зручний інструмент, який допоможе фермерам працювати ефективніше та підвищить їх рентабельність.

Створено повністю функціонуючий додаток, який виконує поставлену перед ним задачу. Інтерфейс адаптовано під різні пристрої. Дані зберігаються у базі даних, логіка по обробці даних відділена від відображення інтерфейсу.

При розробці було використано актуальний стек технологій та створений додаток має великий потенціал для майбутнього додавання функціоналу.

Література

1. Знайомство з React URL: <https://redux.js.org/tutorials/essentials/part-1-overview-concepts>
2. Знайомство з Nest.js URL: <https://habr.com/ru/post/439434/>
3. Порівняння фреймворків Express та Nest URL: <https://conf.ztu.edu.ua/wp-content/uploads/2021/01/3-2.pdf>
4. Знайомство з TypeScript URL: <https://www.typescriptlang.org/docs/handbook/intro.html>

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ
АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ

науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ

(14-16 травня 2024 р.)

ВИКОРИСТАННЯ АГРОДРОНІВ	68
Пігуль О.В. МЕТОДОЛОГІЧНІ ЗАСАДИ ОЦІНКИ ЕКОНОМІЧНИХ УМОВ ЗРОСТАННЯ ВПЛИВУ ШТУЧНОГО	
Поздирко В.С. НАПРЯМИ ПІДВИЩЕННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ГАЛУЗІ	313
ЗЕРНОВИРОБНИЦТВА	
Поздирко В.С. НАПРЯМИ РОЗВИТКУ ГАЛУЗІ РОСЛИННИЦТВА	314
Сьомушкін В.О. УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ БУДІВЕЛЬНОЇ ОРГАНІЗАЦІЇ ЯК	315
ІНСТРУМЕНТ СТАЛОГО РОЗВИТКУ	
Щенякін Д.О. ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ СОНЯЧНИХ ПАНЕЛЕЙ НА	316
НАУКОВО-ДОСЛІДНОМУ ПІДПРИЄМСТВІ	317
Белоконь П.А. СУЧАСНІ АВТОМАТИЗОВАНІ ІНФОРМАЦІЙНІ СИСТЕМИ РЕКРУТИНГУ	318
Стебляк В.В. АНАЛІТИЧНИЙ ОГЛЯД ІТ-ТЕХНОЛОГІЙ ЩОДО ОБЛІКУ ВИТРАТ НА	319
ВИРОБНИЦТВО СІЛЬСЬКО-ГОСПОДАРСЬКОЇ ПРОДУКЦІЇ. ПОСТАНОВКА ЗАДАЧІ	320
Агєєв О.А. ОГЛЯД ПЛАТФОРМ ЕЛЕКТРОННОГО НАВЧАННЯ	321
Блаженко Д.О. МЕТОДИ ТА ПРОБЛЕМИ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ ПІДСИСТЕМ	322
ВЕДЕННЯ ЗАМОВЛЕНЬ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ TRIZ	
Гончар Д.О. АВТОМАТИЗАЦІЯ РОЗРАХУНКУ БАЛАНСУ ГУМУСУ ТА ПОЖИВНИХ РЕЧОВИН В	323
ГРУНТІ	
Доля А.Р. СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ АДМІНІСТРАТИВНИХ МОДУЛІВ	324
Дубінчин В.С. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ	325
КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	
Крамський А.С., Бусенко І.В. ПРОГНОЗУВАННЯ КІЛЬКОСТІ СТУДЕНТІВ В УКРАЇНІ З	326
ВИКОРИСТАННЯМ ЕКОНОМЕТРИЧНИХ МЕТОДІВ	327
Олійник А.В. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО-	
МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	328
Руденок О.А. ПОШУК ІНФОРМАЦІЇ В ТЕКСТОВИХ ДОКУМЕНТАХ З УРАХУВАННЯМ	329
КОНТЕКСТУ	
Санжаків А. А. АНАЛІТИЧНИЙ ОГЛЯД ВИКОРИСТАННЯ ІТ- ТЕХНОЛОГІЙ У МОБІЛЬНІЙ	330
РОЗРОБЦІ ДЛЯ ОС АНДРОЇД	
Сіденко А.С. ПРОБЛЕМИ РЕГІСТРАЦІЇ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ МОНІТОРИНГУ	331
НАВКОЛИШНЬОГО СЕРЕДОВИЩА	
Терещенко С.С. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ПІДСИСТЕМИ АВТОМАТИЗАЦІЇ	332
ОБЛІКОВИХ ПРОЦЕСІВ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	
Чубун М.П. СУЧАСНІ ТЕНДЕНЦІЇ У СТВОРЕННІ ВЕБ-САЙТІВ ДЛЯ ІНТЕРНЕТ-МАГАЗИНІВ	333
КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	
Шелковський С.В. СУЧАСНІ ТЕНДЕНЦІЇ ПРОЕКТУВАННЯ ПІДСИСТЕМИ ОБЛІКУ	334
ПІДПРИЄМСТВ ПЕРЕРОБКИ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ	
Горбань С.В. ТЕОРЕТИЧНІ АСПЕКТИ ФОРМУВАННЯ ПОЗИТИВНОГО ІМІДЖУ	335
ПІДПРИЄМСТВА	
Miao Yinhai INFLUENCE OF MANAGEMENT DECISIONS ON THE ENTERPRISE'S INNOVATION	336
STRATEGY	
Zhao Min THE EFFICIENT MANAGEMENT OF TALENTS FOR THE DEVELOPMENT OF	337
ENTERPRISE	
Заболотна А.І. УПРАВЛІННЯ ПРОЦЕСАМИ ПІДВИЩЕННЯ ЯКОСТІ ЖИТТЯ РЕПОНУ ЗА	338
МАТЕРІАЛАМИ ДРУЖБІВСЬКОЇ ТГ	339
Глек О.А. ОРГАНІЗАЦІЯ КЕЙТЕРИНГОВОГО ОБСЛУГОВУВАННЯ	340
Малюк Н.М. СТАЛІЙ РОЗВИТОК ЛІСОВОГО ГОСПОДАРСТВА	341
Дутов М.М. СТІЙКИЙ РОЗВИТОК ПІДПРИЄМСТВА	342
Бондар В.М. ТЕОРЕТИЧНІ ОСНОВИ КОНКУРЕНТОСПРОМОЖНОСТІ ДІЯЛЬНОСТІ	343
ПІДПРИЄМСТВ	
Макарчук П.В. ОРГАНІЗАЦІЙНА КУЛЬТУРА ЯК ОБ'ЄКТ УПРАВЛІНСЬКОЇ ДІЯЛЬНОСТІ	344
Могиленець В.М. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПЕРСОНАЛОМ В СИСТЕМІ	345
МЕНЕДЖМЕНТУ ПІДПРИЄМСТВА	346
Борщов А.О. ЗАСТОСУВАННЯ ДІАГНОСТИЧНИХ ЗАСОБІВ У ЛОГІСТИЦІ ТА ЇХ ПОШИРЕННЯ	
НА СИСТЕМІ ТРАНСПОРТУ	
Гризодуб Д.Г. АСПЕКТИ УПРАВЛІННЯ ЯКІСТЮ ПОСЛУГ В СУЧАСНИХ УМОВАХ	

АНАЛІТИЧНИЙ ОГЛЯД ІТ-ТЕХНОЛОГІЙ ЩОДО ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО СІЛЬСЬКО-ГОСПОДАРСЬКОЇ ПРОДУКЦІЇ. ПОСТАНОВКА ЗАДАЧІ

Стеблянко В. В. студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник :ж.т.н., доц. Агаджанова С.В.
Сумський НАУ

Тема використання ІТ-технологій, що дозволяють автоматизувати облік витрат на виробництво сільсько-господарської продукції є актуальною на сьогодні. Сучасні агрогосподарства, особливо, малі та середні, в більшості, не користуються сучасними ІТ-технологіями обліку, а вираховують витрати власноруч. Необхідним є розробка сучасного інструментарію, веб-застосунків, які дозволяють забезпечити наступні вимоги:

1. Простота інтерфейсу та інтуїтивність функціоналу;
2. Можливість оновлення та модернізації;
3. Мінімальні затрати на обслуговування;
4. Точність розрахунків та актуальність інформації;
5. Можливість змінювати та оновлювати дані;
6. Розробити інформаційний веб-додаток який включатиме в себе візуальну та серверну частину з певним функціоналом.

Для реалізації проекту необхідно реалізувати наступні задачі:

1. Виконати огляд найбільш популярних інструментів для виконання задачі. Зробити їх аналіз та визначити більш вигідні для розробки варіант.;
2. Розробити клієнтський інтерфейс, який зможе надати користувачеві зручний та візуально приємний користувацький досвід;
3. Спроекувати базу даних для зручного використання та обробки даних в клієнтській частині додатка;
4. Для комфортного користування, додати адаптивний інтерфейс під мобільні пристрої;
- Огляд схожих додатків.
5. Розробити формули, які будуть використовуватися у додатку, для вирішення та створення плану робіт.

6. Розробити та відобразити у швидкому доступі інформаційну інструкцію для користувача, яка дозволить швидко зрозуміти, яким чином потрібно використовувати додаток.

Деякі приклади програмних додатків та рішень, які можуть бути використані для обліку витрат у сільському господарстві:

1. **QuickBooks**: QuickBooks є популярним фінансовим програмним засобом, який може бути використаний для ведення обліку витрат на сільськогосподарську продукцію, закупки ресурсів, оплати праці та багато іншого.
2. **Farmbrite**: Farmbrite - це програмний продукт, спеціально розроблений для сільського господарства. Він надає інструменти для обліку витрат, управління запасами, виробництвом та продажами продукції.
3. **AgriXP**: AgriXP - це інтегрована платформа для управління сільським господарством, яка включає в себе інструменти для обліку витрат, аналізу рентабельності культур, управління ресурсами та багато іншого.
4. **FarmLogs**: FarmLogs - це програмне рішення, яке дозволяє сільським господарям вести облік витрат на паливо, добрива, насіння та інші ресурси, а також вносити дані про врожай та витрати на робочу силу.
5. **Granular**: Granular - це система для сільського господарства, яка надає інструменти для обліку витрат, ведення обліку фінансів та аналізу рентабельності різних аспектів сільського господарства.
6. **Trimble Ag Software**: Ця програма розроблена для сільського господарства і включає в себе функції для обліку витрат, контролю над запасами, аналізу виробництва та збуту продукції.
7. **FarmERP**: FarmERP - це програмний комплекс для сільського господарства, який включає в себе інструменти для обліку витрат, моніторингу виробництва та аналізу фінансів.
8. **Agrivi**: Agrivi - це програмне рішення, спеціально розроблене для обліку витрат та управління сільського господарства. Воно надає інструменти для планування виробництва, контролю над витратами та аналізу врожаю.

Ці програмні додатки можуть допомогти сільським господарям з веденням обліку витрат, оптимізацією процесів та підвищенням ефективності виробництва. Вони забезпечують можливість збереження даних, ведення фінансового обліку та аналізу результатів виробництва для прийняття обґрунтованих управлінських рішень.

ДОДАТОК Г

ЛІСТИНИГИ ПРОГРАМ

Файл index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1"
  />

    <title>Agro App</title>
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link

      href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,400;0,70
0;1,400;1,700&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Файл auth.ts

```
import axios from 'axios';

import { AppDispatch } from '../store/store';
```

```

import { authUser } from '../store/reducers/userSlice';

export const auth = () => {
  return async (dispatch: AppDispatch) => {
    try {
      const response = await axios.get(
        `http://localhost:5000/api/auth/auth`,
        {
          headers: {
            Authorization: `Bearer
${localStorage.getItem('token')}`
          }
        }
      );
      dispatch(authUser(response.data.user));
      localStorage.setItem('token', response.data.token);
    } catch (e) {
      localStorage.removeItem('token');
    }
  };
};

```

Файл **fetchArticles.ts**

```

import axios from 'axios';

import { AppDispatch } from '../store/store';
import { getAllArticles } from '../store/reducers/articlesSlice';

export const fetchArticles = () => {

```

```

return async (dispatch: AppDispatch) => {
  try {
    const response = await axios.get(
      'http://localhost:5000/api/articles'
    );
    dispatch(getAllArticles(response.data.articles));
  } catch (e) {}
};
};

```

Файл fetchRegions.ts

```

import axios from 'axios';

import { AppDispatch } from '../store/store';
import { getAllRegions } from '../store/reducers/regionsSlice';

export const fetchRegions = () => {
  return async (dispatch: AppDispatch) => {
    try {
      const response = await axios.get(
        'http://localhost:5000/api/regions'
      );
      dispatch(getAllRegions(response.data.regions));
    } catch (e) {}
  };
};

```

Файл login.ts

```

import axios from 'axios';

```

```
import { AppDispatch } from '../store/store';

import { authUser, offLoading, onLoading } from '../store/reducers/userSlice';
import { clearFields } from '../store/reducers/loginFormSlice';

export const login = (email: string, password: string) => {
  return async (dispatch: AppDispatch) => {
    try {
      dispatch(onLoading());
      const response = await axios.post(
        `http://localhost:5000/api/auth/login`,
        {
          email,
          password
        }
      );
      dispatch(authUser(response.data.user));
      localStorage.setItem('token', response.data.token);
      dispatch(clearFields());
      dispatch(offLoading());
    } catch (e) {}
  };
};
```

Файл registration.ts

```
import axios from 'axios';

import { AppDispatch } from '../store/store';

import { clearFields } from '../store/reducers/registrationFormSlice';
```

```

export const registration = (
  email: string,
  username: string,
  password: string
) => {
  return async (dispatch: AppDispatch) => {
    try {
      const response = await axios.post(
        `http://localhost:5000/api/auth/registration`,
        {
          email,
          username,
          password
        }
      );
      dispatch(clearFields());
      alert(response.data.message);
    } catch (e) {}
  };
};

```

Файл Login.tsx

```

import { FormEvent } from 'react';

import { useAppDispatch, useAppSelector } from '../hooks/redux';

import {
  changeEmailValue,
  changePasswordValue
} from '../store/reducers/loginFormSlice';

```

```

import { login } from '../asyncActions/login';

import '../scss/Auth.scss';

export function Login() {
  const dispatch = useAppDispatch();
  const { email, password } = useAppSelector(state => state.loginFormReducer);
  const isLoading = useAppSelector(state => state.userReducer.isLoading);
  return (
    <form
      className='login-form'
      onSubmit={(event: FormEvent<HTMLFormElement>) =>
        event.preventDefault()
      }
    >
      <div className='login-form__email'>
        <label
          className='login-form__email-label'
          htmlFor='login-email'
        >
          Почта:
        </label>
        <input
          className='login-form__email-input'
          id='login-email'
          value={email}
          onChange={(event:
FormEvent<HTMLInputElement>) =>

```

```

dispatch(changeEmailValue(event.currentTarget.value))
    }
    type='email'
    placeholder='Введіть свою пошту'
  />
</div>
<div className='login-form__password'>
  <label
    className='login-form__password-label'
    htmlFor='login-password'
  >
    Пароль:
  </label>
  <input
    className='login-form__password-input'
    id='login-password'
    value={password}
    onChange={(event:
FormEvent<HTMLInputElement>) =>

    dispatch(changePasswordValue(event.currentTarget.value))
    }
    type='password'
    placeholder='Введіть свій пароль'
  />
</div>
<div className='login-form__sign-in'>
  {!isLoading ? (
    <button

```

```

        className='login-form__sign-in-button'
        onClick={() => dispatch(login(email,
password)))}
      >
        Увійти
      </button>
    ): (
      <div className='loading'>Loading...</div>
    )}
  </div>
</form>
);
}

```

Файл Registration.tsx

```

import { FormEvent } from 'react';

import { useAppDispatch, useAppSelector } from '../hooks/redux';

import {
  changeEmailValue,
  changePasswordValue,
  changeUsernameValue
} from '../store/reducers/registrationFormSlice';

import { registration } from '../asyncActions/registration';

import '../scss/Auth.scss';

export function Registration(): JSX.Element {
  const { email, username, password } = useAppSelector(

```

```

        state => state.registrationFormReducer
    );
    const dispatch = useAppDispatch();
    return (
        <form
            className='registration-form'
            onSubmit={{(event: FormEvent<HTMLFormElement>) =>
                event.preventDefault()
            }}
        >
            <div className='registration-form__email'>
                <label
                    className='registration-form__email-label'
                    htmlFor='registration-email'
                >
                    Email:
                </label>
                <input
                    className='registration-form__email-input'
                    id='registration-email'
                    value={email}
                    onChange={{(event:
FormEvent<HTMLInputElement>) =>

                    dispatch(changeEmailValue(event.currentTarget.value))
                }}
                    type='email'
                    placeholder='Введіть свою пошту'
                />
            </div>

```

```

<div className='registration-form__username'>
  <label
    className='registration-form__username-label'
    htmlFor='registration-name'
  >
    Ім'я:
  </label>
  <input
    className='registration-form__username-input'
    id='registration-name'
    value={username}
    onChange={(event:
FormEvent<HTMLInputElement>) =>

    dispatch(changeUsernameValue(event.currentTarget.value))
    }
    type='name'
    placeholder='Введіть своє ім'я'
  />
</div>
<div className='registration-form__password'>
  <label
    className='registration-form__password-label'
    htmlFor='registration-password'
  >
    Пароль:
  </label>
  <input
    className='registration-form__password-input'
    id='registration-password'

```

```

        value={password}
        onChange={(event:
FormEvent<HTMLInputElement>) =>

        dispatch(changePasswordValue(event.currentTarget.value))
        }
        type='password'
        placeholder='Введіть свій пароль'
      />
    </div>
    <div className='registration-form__sign-in'>
      <button
        className='registration-form__sign-in-button'
        onClick={() =>
          dispatch(registration(email, username,
password))
        }
      >
        Зареєструватись
      </button>
    </div>
  </form>
);
}

```

Файл AboutArea.tsx

```

import { FormEvent, useState } from 'react';
import { Link } from 'react-router-dom';
import { useAppDispatch } from '../hooks/redux';
import '../scss/AboutArea.scss';
import {

```

```

    addAreaSquare,
    addCadastralNumber
  } from '../store/reducers/areaSlice';

export function AboutArea() {
  const [cadastral, setCadastral] = useState("");
  const [square, setSquare] = useState("");

  const dispatch = useAppDispatch();

  const addInfoAboutArea = () => {
    dispatch(addCadastralNumber(Number(cadastral)));
    dispatch(addAreaSquare(Number(square)));
    setCadastral("");
    setSquare("");
  };

  return (
    <div className='about-area'>
      <div className='about-area__inner'>
        <h2 className='about-area__title'>
          Введіть дані про вашу ділянку
        </h2>
        <div className='about-area__cadastral'>
          <label htmlFor='cadastral'>Введіть кадастровий
номер</label>
          <input
            type='number'
            value={cadastral}

```

```

                                onChange={{(event:
FormEvent<HTMLInputElement>) =>
                                setCadastral(event.currentTarget.value)
                                }
                                name='cadastral'
                                id='cadastral'

placeholder='XXXXXXXXXX:XX:XXX:XXXX'
                                />
                                </div>
                                <div className='about-area__square'>
                                <label htmlFor='square'>Введіть площу вашої
ділянки</label>
                                <input
                                type='number'
                                value={square}
                                onChange={{(event:
FormEvent<HTMLInputElement>) =>
                                setSquare(event.currentTarget.value)
                                }
                                name='square'
                                id='square'
                                placeholder='2 Га'
                                />
                                </div>
                                <div className='about-area__btns'>
                                <div className='about-area__add-info'>
                                <button
onClick={addInfoAboutArea}>Додати дані</button>
                                </div>

```

```

        <div className='about-area__next-page'>
            <Link to='/home/handbook/create-
plan/confirm-plan'>
                Наступний крок
            </Link>
        </div>
    </div>
</div>
</div>
);
}

```

Файл ConfirmPlan.tsx

```

import { Link } from 'react-router-dom';
import { useAppDispatch, useAppSelector } from '../hooks/redux';

import '../scss/FinalPlan.scss';
import {
    calculateCornPrice,
    calculateSunflowerAndSoyPrice,
    calculateWheatPrice,
    showCornDetails,
    showSunflowerAndSoyDetails,
    showWheatDetails
} from '../store/reducers/calculationsSlice';
import { Plan } from './Plan';

export function ConfirmPlan() {
    const { agriculture, squareSize } = useAppSelector(
        state => state.areaReducer
    );
}

```

```

const dispatch = useAppDispatch();

const calculatePlan = () => {
  if (agriculture === 'Пшениця') {
    dispatch(calculateWheatPrice(squareSize));
    dispatch(showWheatDetails(squareSize));
  }
  if (agriculture === 'Кукурудза') {
    dispatch(calculateCornPrice(squareSize));
    dispatch(showCornDetails(squareSize));
  }
  if (agriculture === 'Соняшник') {
    dispatch(calculateSunflowerAndSoyPrice(squareSize));
    dispatch(showSunflowerAndSoyDetails(squareSize));
  }
  if (agriculture === 'Соя') {
    dispatch(calculateSunflowerAndSoyPrice(squareSize));
    dispatch(showSunflowerAndSoyDetails(squareSize));
  }
};

return (
  <div className='final-plan'>
    <div className='final-plan__inner'>
      <h2 className='final-plan__title'>
        Створити план з наступними даними
      </h2>
      <Plan />
      <Link
        to='/home/handbook/create-plan/final-plan'

```

```

        className='final-plan__create-plan'
        onClick={calculatePlan}
      >
        Створити план
      </Link>
    </div>
  </div>
);
}

```

Файл FinalPlan.tsx

```

import axios from 'axios';
import { useState } from 'react';
import { useAppSelector } from '../hooks/redux';
import { Plan } from './Plan';

export function FinalPlan() {
  const [saved, setSaved] = useState(false);
  const totalPriceData = useAppSelector(
    state => state.calculationsReducer.totalPrice
  );
  const planInfo = useAppSelector(state => state.areaReducer);
  const planCalculateData = useAppSelector(
    state => state.calculationsReducer.totalPrice
  );
  const userId = useAppSelector(state => state.userReducer.currentUser.id);

  const savePlan = async () => {
    await axios.post(`http://localhost:5000/api/area-plan`, {
      ...planInfo,

```

```

        ...planCalculateData,
        _id: userId
    });
    setSaved(true);
};

return (
    <div className='final-plan'>
        {!saved ? (
            <div className='final-plan__inner'>
                <div className='final-plan__title title'>План</div>
                <Plan />
                <h2 className='final-plan__title'>Деталі</h2>
                <div className='final-plan__details'>
                    <div className='final-plan__detail'>
                        Загальна вартість:
{totalPriceData.totalPrice} грн.
                    </div>
                    <div className='final-plan__detail'>
                        Витрати пального: {totalPriceData.DT}
грн.
                    </div>
                    <div className='final-plan__detail'>
                        Насіння: {totalPriceData.seed} грн.
                    </div>
                    <div className='final-plan__detail'>
                        Міңдобрива: {totalPriceData.fertilizers}
грн.
                    </div>
                    <div className='final-plan__detail'>

```

Сульфат амонію:

{totalPriceData.sulfate} грн.

</div>

<div className='final-plan__detail'>

Селітра аміачна:

{totalPriceData.saltpeter} грн.

</div>

<div className='final-plan__detail'>

Гербіцид: {totalPriceData.herbicide}

грн.

</div>

<div className='final-plan__detail'>

Фунгіцид: {totalPriceData.fungicide}

грн.

</div>

<div className='final-plan__detail'>

Мікро мінерали:

{totalPriceData.microMinerals} грн.

</div>

<div className='final-plan__detail'>

Інсектицид: {totalPriceData.insecticide}

грн.

</div>

<div className='final-plan__detail'>

Карбамід: {totalPriceData.urea} грн.

</div>

<div className='final-plan__detail'>

Цинк: {totalPriceData.zinc} грн.

</div>

</div>

```

        <button className='final-plan__save'
onClick={savePlan}>
            Зберегти план
        </button>
    </div>
    ): (
        <div className='final-plan__saved'>
            Ваш план збережено! <br /> Щоб переглянути,
перейдіть у
            розділ "Ваші плани"
        </div>
    )}
</div>
);
}

```

Файл Plan.tsx

```

import { Fragment } from 'react';
import { useAppSelector } from '../hooks/redux';

export function Plan() {
    const { region, agriculture, squareSize, cadastralNumber } = useAppSelector(
        state => state.areaReducer
    );

    const firstTenNumbers = cadastralNumber.toString().slice(0, 10);
    const secondTwoNumbers = cadastralNumber.toString().slice(10, 12);
    const thirdTreeNumbers = cadastralNumber.toString().slice(12, 15);
    const fourthFourNumbers = cadastralNumber.toString().slice(15, 19);

```

```

return (
  <Fragment>
    <div className='final-plan__region'>Регіон: {region}</div>
    <div className='final-plan__agriculture'>
      Агрокультура: {agriculture}
    </div>
    <div className='final-plan__square'>
      Площа ділянки: {squareSize} Га
    </div>
    <div className='final-plan__cadastral'>
      <span className='final-plan__cadastral-title'>
        Кадастровий номер:
      </span>
      `{${firstTenNumbers} : ${secondTwoNumbers} :
${thirdTreeNumbers} : ${fourthFourNumbers}}`
    </div>
  </Fragment>
);
}

```

Файл Region.tsx

```

import { useEffect, FormEvent, useState } from 'react';
import { fetchRegions } from '../asyncActions/fetchRegions';
import { useAppDispatch, useAppSelector } from '../hooks/redux';

import {
  searchRegion,
  showAllRegionsAfterClearInput
} from '../store/reducers/regionsSlice';
import { addRegion } from '../store/reducers/areaSlice';

```

```

import '../scss/CreatePlan.scss';
import { Link } from 'react-router-dom';

export function Region() {
  const dispatch = useAppDispatch();
  const regions = useAppSelector(state => state.regionsReducer.regions);
  const searchedRegion = useAppSelector(
    state => state.regionsReducer.searchedRegion
  );
  const searchActive = useAppSelector(
    state => state.regionsReducer.searchActive
  );
  const region = useAppSelector(state => state.areaReducer.region);

  useEffect(() => {
    dispatch(fetchRegions());
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  const [search, setSearch] = useState("");
  const searchInputRegion = () => {
    if (!search) return;
    const searchedRegion = regions.filter(
      region => region.regionName === search
    );
    dispatch(searchRegion(searchedRegion));
  };

  useEffect(() => {

```

```

    if (!search) {
      dispatch(showAllRegionsAfterClearInput());
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [search]);

const addRegionToList = (event: any) => {
  const regionTitle = event.currentTarget.textContent;
  dispatch(addRegion(regionTitle));
};

return (
  <div className='region'>
    <h2 className='region__title'>
      Виберіть регіон, де знаходиться ваша земельна ділянка
    </h2>
    <div className='region__search-field'>
      <input
        type='text'
        placeholder='Пошук'
        value={search}
        onChange={(event:
FormEvent<HTMLInputElement>) =>
          setSearch(event.currentTarget.value)
        }
      />
      <button className='region__button'
onClick={searchInputRegion}>
        Знайти
      </button>
    </div>
  </div>
);

```

```

</div>
<div className='region__items'>
  {!searchActive
    ? regions.map(region => (
      <div
        className='region__item'
        key={region._id}
        onClick={addRegionToList}
      >
        {region.regionName}
      </div>
    ))
    : searchedRegion.map(region => (
      <div
        className='region__item'
        key={region._id}
        onClick={addRegionToList}
      >
        {region.regionName}
      </div>
    ))}
</div>
{region ? (
  <div className='info next-page'>
    `{${region} додана до списку`}
    <Link to='/home/handbook/create-plan/agriculture'>
      Наступний крок
    </Link>
  </div>
) : null}

```

```

        </div>

    );
}

```

Файл SelectAgroculture.tsx

```

import { Link } from 'react-router-dom';
import { useAppDispatch } from '../hooks/redux';
import { addAgriculture } from '../store/reducers/areaSlice';

import '../scss/SelectAgriculture.scss';

export function SelectAgriculture() {
    const dispatch = useAppDispatch();

    const getSelectedItem = (event: any) => {
        dispatch(addAgriculture(event.currentTarget.textContent));
    };

    return (
        <div className='select-agriculture'>
            <div className='select-agriculture__inner'>
                <h2 className='select-agriculture__title'>
                    Оберіть агрокультуру
                </h2>
                <div className='select-agriculture__options'>
                    <div className='select-agriculture__option'>
                        <label htmlFor='wheat'
onClick={getSelectedItem}>
                            Пшениця
                        </label>

```

```

        <span>
            <input type='radio' name='agriculture'
id='wheat' />
            <span className='checkmark'></span>
        </span>
    </div>
    <div className='select-agriculture__option'>
        <label htmlFor='corn'
onClick={getSelectedItem}>
            Кукурудза
        </label>
        <span>
            <input type='radio' name='agriculture'
id='corn' />
            <span className='checkmark'></span>
        </span>
    </div>
    <div className='select-agriculture__option'>
        <label htmlFor='sunflower'
onClick={getSelectedItem}>
            Соняшник
        </label>
        <span>
            <input
                type='radio'
                name='agriculture'
                id='sunflower'
            />
            <span className='checkmark'></span>
        </span>
    </div>

```

```

        </div>
        <div className='select-agriculture__option'>
            <label htmlFor='soy'
onClick={getSelectedItem}>
                Соя
            </label>
            <span>
                <input type='radio' name='agriculture'
id='soy' />
                <span className='checkmark'></span>
            </span>
        </div>
    </div>
    <div className='select-agriculture__next-page'>
        <Link to='/home/handbook/create-plan/about-area'>
            Наступний крок
        </Link>
    </div>
    <div className='select-agriculture__info'>
        Примітка: в залежності від вибору агрокультури,
загальна
        вартість обробітку буде відрізнятись
    </div>
</div>
</div>
</div>
    );
}

```

Файл Agriculture.tsx

```
import { Fragment } from 'react';
```

```

import { Link } from 'react-router-dom';
import { useAppDispatch, useAppSelector } from '../hooks/redux';
import { getArticle } from '../store/reducers/articlesSlice';
import { PlantIcon } from '../svg/PlantIcon';

export function Agriculture() {
  const dispatch = useAppDispatch();
  const articles = useAppSelector(state => state.articlesReducer.articles);
  return (
    <Fragment>
      <h3 className='handbook__list-title'>Агрокультура</h3>
      <ul className='handbook__list'>
        {articles.map(title => (
          <li className='handbook__list-item' key={title._id}>
            <Link
              to='/home/handbook/article'
              // TODO: find valid type to event
              onClick={(event: any) =>
                dispatch(
                  getArticle(event.currentTarget.textContent)
                )
              }
            >
              <PlantIcon />
              {title.title}
            </Link>
          </li>
        ))}
      </ul>
    )
  )
}

```

```

        </Fragment>
    );
}

```

Файл Article.tsx

```

import { useAppSelector } from '../hooks/redux';

export function Article() {
    const articles = useAppSelector(state => state.articlesReducer.articles);
    const currentArticle = useAppSelector(state => state.articlesReducer.title);
    const article = articles.find(obj => obj.title === currentArticle);
    return (
        <div className='article'>
            <h3 className='article__title'>{article?.title}</h3>
            <p className='article__text'>{article?.body}</p>
        </div>
    );
}

```

Файл Handbook.tsx

```

import { useEffect } from 'react';
import { useAppDispatch } from '../hooks/redux';
import { fetchArticles } from '../asyncActions/fetchArticles';
import { Agriculture } from './Agriculture';
import { WorkTypes } from './WorkTypes';
import '../scss/Handbook.scss';

export function Handbook() {
    const dispatch = useAppDispatch();

```

```

useEffect(() => {
    dispatch(fetchArticles());
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

return (
    <div className='handbook'>
        <h2 className='handbook__title'>Актуальний список
статей</h2>
        <div className='handbook__navigation'>
            <Agriculture />
            <WorkTypes />
        </div>
    </div>
);
}

```

Файл WorkTypes.tsx

```

import { Link } from 'react-router-dom';
import { WorkTypesIcon } from '../svg/WorkTypesIcon';

export function WorkTypes() {
    return (
        <ul className='handbook__list'>
            <h3 className='handbook__list-title'>Види робіт</h3>
            <li className='handbook__list-item'>
                <Link to='/home/handbook/article'>
                    <WorkTypesIcon />

```

```

        Підготовка ґрунту
        </Link>
      </li>
      <li className='handbook__list-item'>
        <Link to='/home/handbook/article'>
          <WorkTypesIcon />
          Посів
        </Link>
      </li>
      <li className='handbook__list-item'>
        <Link to='/home/handbook/article'>
          <WorkTypesIcon />
          Захист рослини
        </Link>
      </li>
      <li className='handbook__list-item'>
        <Link to='/home/handbook/article'>
          <WorkTypesIcon />
          Збирання врожаю
        </Link>
      </li>
    </ul>
  );
}

```

Файл Slider.tsx

```

import { Navigation, A11y, Pagination } from 'swiper';

import { Swiper, SwiperSlide } from 'swiper/react';

```

```

import 'swiper/css';
import '../scss/Slider.scss';

const article = require('../assets/Articles.jpg');
const region = require('../assets/Region.jpg');
const regionSearch = require('../assets/RegionSearch.jpg');
const agriculture = require('../assets/Agriculture.jpg');
const confirmPlan = require('../assets/ConfirmPlan.jpg');
const savePlan = require('../assets/SavePlan.jpg');
const cadastral = require('../assets/Cadastral.jpg');
const userPlans = require('../assets/UserPlans.jpg');

export function Slider() {
  return (
    <Swiper
      modules={[Navigation, Pagination, A11y]}
      spaceBetween={50}
      slidesPerView={1}
      pagination={{ clickable: true }}
      onSwiper={swiper => console.log(swiper)}
      onSlideChange={() => console.log('slide change')}
    >
    <SwiperSlide>
      <h2>Сторінка з статтями</h2>
      <div>
        На сторінці з статтями, ви можете знайти
        інформацію, яка
        може знадобитись при обробці вашої земельної
        ділянки.
      </div>
    </SwiperSlide>
  )
}

```

```

        <img src={article} alt='Articles' />
    </SwiperSlide>
    <SwiperSlide>
        <h2>Початок роботи зі створення плану</h2>
        <div>
            При переході до створення плану, ви потрапите на
сторінку,

            де необхідно вибрати область, де знаходиться
ваша ділянка

        </div>
        <img src={region} alt='Region' />
    </SwiperSlide>
    <SwiperSlide>
        <h2>Пошук свого регіону</h2>
        <div>
            Також є можливість ввести назву своєї області,
щоб

            пришвидшити процес

        </div>
        <img src={regionSearch} alt='Search Region' />
    </SwiperSlide>
    <SwiperSlide>
        <h2>Вибір агрокультури</h2>
        <div>
            Далі необхідно обрати агрокультуру, яку ви
будете

            вирощувати.

        </div>
        <img src={agriculture} alt='Select Agriculture' />
    </SwiperSlide>

```

```

<SwiperSlide>
  <h2>Введення даних про ділянку</h2>
  <div>
    На цій сторінці необхідно ввести кадастровий
номер та площу
    вашої земельної ділянки
  </div>
  <img src={cadastral} alt='Cadastral' />
</SwiperSlide>
<SwiperSlide>
  <h2>Підтвердження плану</h2>
  <div>
    Вам буде наведено перелік даних для створення
плану робіт на
    вашій земельній ділянці. Необхідно перевірити
дані та
    підтвердити створення, якщо все правильно
  </div>
  <img src={confirmPlan} alt='Confirm Plan' />
</SwiperSlide>
<SwiperSlide>
  <h2>Збереження плану</h2>
  <div>
    Вам буде представлено план робіт для з
детальними цінами
  </div>
  <img src={savePlan} alt='Save Plan' />
</SwiperSlide>
<SwiperSlide>
  <h2>Ваші плани</h2>

```

```

        <div>
            На сторінці представлений список створених вами
планів
        </div>
        <img src={userPlans} alt='User plans' />
    </SwiperSlide>
</Swiper>
);
}

```

Файл App.tsx

```

import { useEffect } from 'react';
import { Route, Routes, useNavigate } from 'react-router-dom';

import { useAppDispatch, useAppSelector } from '../hooks/redux';

import { auth } from '../asyncActions/auth';

import { Auth } from '../pages/Auth';
import { Home } from '../pages/Home';

import '../scss/App.scss';

export function App(): JSX.Element {
    const isAuth = useAppSelector(state => state.userReducer.isAuth);
    const navigate = useNavigate();
    const dispatch = useAppDispatch();

    useEffect(() => {
        dispatch(auth());
    });
}

```

```

        // eslint-disable-next-line react-hooks/exhaustive-deps
    }, []);

    useEffect(() => {
        !isAuth
            ? navigate('auth/login', { replace: true })
            : navigate('home/guide', { replace: true });
        // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [isAuth]);

    return (
        <div className='container'>
            <Routes>
                <Route path='/'>
                    {!isAuth ? (
                        <Route path='auth/*' element={ <Auth /> } />
                    ) : (
                        <Route path='home/*' element={ <Home /> } />
                    )}
                </Route>
            </Routes>
        </div>
    );
}

```

Файл DesktopMenu.tsx

```

import { NavLink } from 'react-router-dom';

import { useDispatch } from '../hooks/redux';

```

```

import { logoutUser } from '../store/reducers/userSlice';

import '../scss/Home.scss';

export const DesktopMenu = () => {
  const dispatch = useAppDispatch();
  return (
    <nav className='header__navigation'>
      <ul className='header__navigation-list'>
        <NavLink to='/home/guide'
className='header__navigation-item'>
          Гайд
        </NavLink>
        <NavLink
          to='/home/handbook'
          className='header__navigation-item'
        >
          Довідник
        </NavLink>
        <NavLink
          to='/home/handbook/create-plan/region'
          className='header__navigation-item'
        >
          Створити план
        </NavLink>
        <NavLink to='/home/plans'
className='header__navigation-item'>
          Ваші плани
        </NavLink>
        <button

```

```

        className='header__navigation-item'
        onClick={() => dispatch(logoutUser())}
      >
        Вийти
      </button>
    </ul>
  </nav>
);
};

```

Файл Header.tsx

```

import { useAppDispatch, useAppSelector } from '../hooks/redux';

import { DesktopMenu } from './DesktopMenu';

import { handleModal } from '../store/reducers/headerBurgerSlice';
import { ResponsiveMenu } from './ResponsiveMenu';

export function Header() {
  const burgerClassName = useAppSelector(
    state => state.headerBurgerReducer.burgerClassName
  );
  const openModal = useAppSelector(
    state => state.headerBurgerReducer.openModal
  );
  const dispatch = useAppDispatch();
  return (
    <header className='header'>
      <div className='header__container'>
        <div className='header__logo'>Agro App</div>
        <DesktopMenu />

```

```

        <div
            className={burgerClassName}
            onClick={() => dispatch(handleModal())}
        >
            <span></span>
        </div>
    </div>
    {openModal ? <ResponsiveMenu /> : null}
</header>
);
}

```

Файл ResponsiveMenu.tsx

```

import { NavLink } from 'react-router-dom';
import { useAppDispatch } from '../hooks/redux';
import { handleModal } from '../store/reducers/headerBurgerSlice';
import { logoutUser } from '../store/reducers/userSlice';

import '../scss/Home.scss';

export function ResponsiveMenu() {
    const dispatch = useAppDispatch();
    return (
        <nav className='menu__navigation'>
            <ul className='menu__navigation-list'>
                <NavLink
                    to='/home/guide'
                    className='menu__navigation-item'
                    onClick={() => dispatch(handleModal())}
                >

```

Гайд

</NavLink>

<NavLink

to='/home/handbook'

className='menu__navigation-item'

onClick={() => dispatch(handleModal())}

>

Довідник

</NavLink>

<NavLink

to='/home/handbook/create-plan/region'

className='menu__navigation-item'

onClick={() => dispatch(handleModal())}

>

Створити план

</NavLink>

<NavLink

to='/home/plans'

className='menu__navigation-item'

onClick={() => dispatch(handleModal())}

>

Ваші плани

</NavLink>

<button

className='menu__navigation-item'

onClick={() => dispatch(logoutUser())}

>

Вийти

</button>


```

<div
    className='user-plans__plan'
    key={plan.planData._id}
>
    <div className='user-plans__plan-title'>
        {`${index + 1}. `}
        {`Кадастровий номер: ${plan.planData.cadastralNumber}`}
        <div className='info'>
            <h3 className='info__title'>Інформація</h3>
            <div className='info__region'>{`Регіон: ${plan.planData.region}`}</div>
            <div className='info__agriculture'>{`Агрокультура:
            ${plan.planData.agriculture}`}</div>
        </div>
        <div className='user-plans__details'>
            <div className='user-plans__detail'>
                </div>
                <div className='user-plans__detail'>
                    {`Насіння: ${plan.planData.seed} грн.`}
                </div>
                <div className='user-plans__detail'>
                    {`Міндобрива: ${plan.planData.fertilizers} грн.`}
                </div>
            <div className='user-plans__detail'>
                {`Сульфат амонію: ${plan.planData.sulfate} грн.`}
            </div>
            <div className='user-plans__detail'>
                {`Селітра аміачна: ${plan.planData.saltpeter} грн.`}
            </div>
            <div className='user-plans__detail'>
                {`Гербіцид: ${plan.planData.herbicide} грн.`}

```

```

</div>
<div className='user-plans__detail'>
    {`Мікро
    мінерали:
    ${plan.planData.microMinerals} грн.`}
</div>
<div className='user-
plans__detail'>
    {`Інсектицид:
    ${plan.planData.insecticide} грн.`}
</div>
<div className='user-
plans__detail'>
    {`Карбамід:
    ${plan.planData.urea} грн.`}
</div>
<div className='user-
plans__detail'>
    {`Цинк
    ${plan.planData.zinc} грн.`}
</div>
</div>
</div>
</div>
</div>
    )}
</div>
</div>
</div>
    );
}

```

Файл redux.ts

```
import { TypedUseSelectorHook, useDispatch, useSelector } from 'react-redux';
```

```
import { AppDispatch, RootState } from '../store/store';
```

```
export const useAppDispatch = () => useDispatch<AppDispatch>();
```

```
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
```

Файл Auth.tsx

```
import { NavLink, Route, Routes } from 'react-router-dom';
```

```
import { Registration } from '../components/Authorization/Registration';
```

```
import { Login } from '../components/Authorization/Login';
```

```
import '../scss/Auth.scss';
```

```
export function Auth() {
```

```
  return (
```

```
    <div className='auth-page'>
```

```
      <div className='auth-page__container'>
```

```
        <div className='auth-page__links'>
```

```
          <NavLink
```

```
            className='auth-page__registration'
```

```
            to='registration'
```

```
          >
```

```
            Реєстрація
```

```
          </NavLink>
```

```
          <NavLink className='auth-page__login' to='login'>
```

```
            Авторизація
```

```

        </NavLink>
      </div>
    <Routes>
      <Route path="/">
        <Route path='registration'
element={<Registration />} />
        <Route path='login' element={<Login />} />
      </Route>
    </Routes>
  </div>
</div>
);
}

```

Файл Home.tsx

```

import { Route, Routes } from 'react-router-dom';
import { Handbook } from '../components/Handbook/Handbook';
import { Article } from '../components/Handbook/Article';
import { Region } from '../components/CreatePlan/Region';
import { Header } from '../components/Static/Header';

import '../scss/Home.scss';
import { SelectAgriculture } from '../components/CreatePlan/SelectAgriculture';
import { AboutArea } from '../components/CreatePlan/AboutArea';
import { ConfirmPlan } from '../components/CreatePlan/ConfirmPlan';
import { FinalPlan } from '../components/CreatePlan/FinalPlan';
import { UserPlans } from '../components/UserPlans/UserPlans';
import { Slider } from '../components/Slider/Slider';

export function Home() {
  return (

```

```

<div className='home-page'>
  <Header />
  <Routes>
    <Route path='handbook' element={<Handbook />} />
    <Route path='handbook/article' element={<Article />} />
    <Route
      path='handbook/create-plan/region'
      element={<Region />}
    />
    <Route
      path='handbook/create-plan/agriculture'
      element={<SelectAgriculture />}
    />
    <Route
      path='handbook/create-plan/about-area'
      element={<AboutArea />}
    />
    <Route
      path='handbook/create-plan/confirm-plan'
      element={<ConfirmPlan />}
    />
    <Route
      path='handbook/create-plan/final-plan'
      element={<FinalPlan />}
    />
    <Route path='plans' element={<UserPlans />} />
    <Route path='guide' element={<Slider />} />
  </Routes>
</div>
);

```

```
}
```

Файл _vars.scss

```
//colors
```

```
$white: #EAEBED;
```

```
$blue: #98DAD9;
```

```
$middle-blue: #5B8291;
```

```
$dark-blue: #2E424D;
```

```
//fonts
```

```
$main-font: 'Poppins', sans-serif;
```

Файл AboutArea.scss

```
@import 'vars';
```

```
.about-area {
```

```
    display: flex;
```

```
    flex-direction: column;
```

```
    justify-content: center;
```

```
    align-items: center;
```

```
    &__inner {
```

```
        display: flex;
```

```
        flex-direction: column;
```

```
        justify-content: center;
```

```
        align-items: center;
```

```
        border: 1px solid rgba(0, 0, 0, 0.02);
```

```
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
```

```
                    rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
```

```
        border-radius: 5px;
```

```
        padding: 50px 100px;
```

```

}
height: 92vh;
&__title {
    font-weight: 400;
    letter-spacing: 1px;
    color: $middle-blue;
    padding: 5px 0 15px 0;
}
&__cadastral,
&__square {
    display: flex;
    flex-direction: column;
    align-items: center;
    label {
        font-size: 18px;
        color: $middle-blue;
        letter-spacing: 1px;
        padding-bottom: 5px;
    }
    input {
        padding: 12px 100px 12px 20px;
        font-size: 16px;
        letter-spacing: 1px;
        outline: none;
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
        border-radius: 5px;
        color: $dark-blue;
        margin-bottom: 20px;
    }
}

```

```

        &::-webkit-outer-spin-button,
        &::-webkit-inner-spin-button {
            -webkit-appearance: none;
            margin: 0;
        }
        &::placeholder {
            color: $middle-blue;
        }
    }
}

&__btns {
    display: flex;
    align-items: center;
}

&__add-info button {
    border: 1px solid $middle-blue;
    padding: 6px 30px;
    border-radius: 5px;
    margin-left: 10px;
    font-size: 18px;
    color: $middle-blue;
    &:hover {
        cursor: pointer;
        background-color: $middle-blue;
        color: $white;
        transition: all 0.3s;
    }
}

&__next-page a {
    border: 1px solid $middle-blue;

```

```
padding: 7px 30px;
border-radius: 5px;
margin-left: 10px;
font-size: 18px;
color: $middle-blue;
text-decoration: none;
```

```
&:hover {
    cursor: pointer;
    background-color: $middle-blue;
    color: $white;
    transition: all 0.3s;
}
```

```
}
```

```
}
```

Файл App.scss

```
@import 'vars';
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: $main-font;
}
```

```
#root, .container {
    min-height: 100vh;
}
```

Файл Auth.scss

```
@import 'vars';
```

```

.auth-page {
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background: $white;

  &__container {
    max-width: 1600px;
    margin: 0 auto;
    background: $white;
    padding: 50px 50px;
    border-radius: 25px;
    box-shadow: rgba(0, 0, 0, 0.16) 0 10px 36px 0, rgba(0, 0, 0, 0.06) 0 0 0 1px;
  }

  &__links {
    text-align: center;
    padding: 10px 15px;
    margin-bottom: 15px;

    a {
      display: inline-block;
      padding: 10px 30px;
      border: 2px solid $middle-blue;
      text-decoration: none;
      color: $white;
      background: $middle-blue;
      border-radius: 5px;
    }
  }
}

```

```
font-size: 18px;
text-transform: uppercase;
letter-spacing: 1px;
box-shadow: 3px 3px 10px $middle-blue;

&:hover {
    box-shadow: 3px 3px 10px $blue;
    background: $white;
    color: $dark-blue;
    transition: .5s;
}
}
}

&__registration {
    margin-right: 2.5px;
}

&__login {
    margin-left: 2.5px;
}

.active {
    background: $white;
    color: $dark-blue;
}

.login-form, .registration-form {
    display: flex;
```

```
flex-direction: column;
align-items: center;
color: $dark-blue;
```

```
&__email, &__password, &__username {
  display: flex;
  flex-direction: column;
}
```

```
&__email-input, &__password-input, &__username-input {
  padding: 12px 100px 12px 20px;
  font-size: 16px;
  letter-spacing: 1px;
  outline: none;
  border: 2px solid $middle-blue;
  border-radius: 5px;
  box-shadow: 3px 3px 5px $middle-blue;
  color: $dark-blue;
  margin-bottom: 20px;
```

```
&::placeholder {
  color: $middle-blue;
}
```

```
&:focus {
  box-shadow: 3px 3px 10px $middle-blue;
}
}
```

```
&__email-label, &__password-label, &__username-label {
```

```

padding: 0 0 5px 5px;
}

&__sign-in-button {
  font-size: 18px;
  padding: 10px 50px;
  letter-spacing: 2px;
  box-shadow: 3px 3px 10px $middle-blue;
  background: $middle-blue;
  border: 2px solid $middle-blue;
  border-radius: 5px;
  color: $white;
  margin-top: 10px;

  &:hover {
    cursor: pointer;
    background: $white;
    color: $dark-blue;
    transition: .3s;
  }
}
}

}

@media (max-width: 530px) {
  .auth-page__links {
    display: flex;
    flex-direction: column;
    justify-content: center;
  }
}

```

```

    a {
      margin: 7.5px 0;
      padding: 8px 15px;
    }
  }
}

```

Файл CreatePlan.scss

```

@import 'vars';

.region {
  display: flex;
  flex-direction: column;
  align-items: center;
  &__title {
    color: $middle-blue;
    font-weight: 400;
    padding-top: 50px;
  }
  &__search-field {
    padding: 20px;
    margin-top: 20px;
    display: flex;
    align-items: center;
    input {
      padding: 10px 20px;
      color: $middle-blue;
      border: 1px solid rgba(0, 0, 0, 0.02);
      box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
      border-radius: 5px;
    }
  }
}

```

```

        outline: none;
        &::placeholder {
            color: $middle-blue;
            font-size: 16px;
        }
    }
}

&__button {
    border: 1px solid $middle-blue;
    padding: 6px 30px;
    border-radius: 5px;
    margin-left: 10px;
    font-size: 18px;
    color: $middle-blue;
    &:hover {
        cursor: pointer;
        background-color: $middle-blue;
        color: $white;
        transition: all 0.3s;
    }
}

&__items {
    display: flex;
    flex-direction: column;
    align-items: center;
    min-width: 500px;
    height: 500px;
    overflow: scroll;
    overflow-x: hidden;
    border: 1px solid rgba(0, 0, 0, 0.02);

```

```

    box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
                rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
    border-radius: 5px;
    &::-webkit-scrollbar {
        width: 5px;
    }
    &::-webkit-scrollbar-track {
        background-color: #eee;
    }
    &::-webkit-scrollbar-thumb {
        box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
        border-radius: 5px;
    }
}

&__item {
    border: 1px solid rgba(0, 0, 0, 0.02);
    box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
                rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
    border-radius: 5px;
    padding: 5px;
    min-width: 100%;
    text-align: center;
    margin: 3px;
    color: $middle-blue;
    font-size: 16px;
    letter-spacing: 1px;
    &.active {
        background-color: $middle-blue;
        color: $white;
    }
}

```

```

        &:hover {
            cursor: pointer;
        }
    }
    .info {
        color: $middle-blue;
        font-size: 16px;
        padding: 10px;
        letter-spacing: 1px;
        margin-top: 10px;
    }
    .next-page a {
        border: 1px solid $middle-blue;
        padding: 6px 30px;
        border-radius: 5px;
        margin-left: 10px;
        font-size: 18px;
        color: $middle-blue;
        text-decoration: none;
        &:hover {
            cursor: pointer;
            background-color: $middle-blue;
            color: $white;
            transition: all 0.3s;
        }
    }
}

@media (max-width: 640px) {
    .region {

```

```

        &__title {
            width: 400px;
            text-align: center;
        }
        .info {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
    }
}

@media (max-width: 520px) {
    .region {
        &__items {
            min-width: 400px;
        }
    }
}

@media (max-width: 420px) {
    .region {
        &__items {
            min-width: 300px;
        }
    }
}

```

Файл FinalPlan.scss

```
@import 'vars';
```

```

.final-plan {
    display: flex;

```

```

justify-content: center;
align-items: center;
height: 93vh;
&__inner {
    border: 1px solid rgba(0, 0, 0, 0.02);
    box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
    border-radius: 5px;
    padding: 30px 50px;
    display: flex;
    flex-direction: column;
    align-items: center;
}
&__title {
    color: $middle-blue;
    letter-spacing: 1px;
    padding-bottom: 15px;
    font-weight: 400;
    margin-top: 10px;
}
&__title.title {
    font-size: 24px;
    margin-top: 0;
}
&__region,
&__agriculture,
&__square,
&__cadastral {
    border: 1px solid rgba(0, 0, 0, 0.02);
    box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,

```

```

        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
border-radius: 5px;
padding: 10px 0 10px 10px;
margin: 5px 0;
color: $middle-blue;
font-size: 18px;
width: 421px;
span {
    margin-right: 5px;
}
}
&__create-plan {
    border: 1px solid $middle-blue;
padding: 6px 30px;
border-radius: 5px;
margin-left: 10px;
font-size: 18px;
color: $middle-blue;
margin-top: 10px;
text-decoration: none;
&:hover {
    cursor: pointer;
background-color: $middle-blue;
color: $white;
transition: all 0.3s;
}
}
&__details {
display: grid;
grid-template-columns: 300px 300px 300px;

```

```

        justify-items: center;
        grid-gap: 10px;
    }
    &__detail {
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
        border-radius: 5px;
        padding: 5px 10px;
        width: 300px;
        color: $middle-blue;
        font-size: 16px;
        letter-spacing: 1px;
    }
    &__save {
        border: 1px solid $middle-blue;
        padding: 6px 30px;
        border-radius: 5px;
        margin-left: 10px;
        font-size: 18px;
        color: $middle-blue;
        margin-top: 20px;
        &:hover {
            cursor: pointer;
            background-color: $middle-blue;
            color: $white;
            transition: all 0.3s;
        }
    }
    &__saved {

```

```

        font-size: 30px;
        color: $middle-blue;
        letter-spacing: 1px;
        text-align: center;
    }
}

@media (max-width: 1040px) {
    .final-plan__details {
        grid-template-columns: 400px 400px;
    }
}

```

Файл Handbook.scss

```

@import 'vars';

.handbook {
    min-height: calc(100vh - 63px);
    display: flex;
    align-items: center;
    flex-direction: column;
    &__title {
        color: $middle-blue;
        padding: 50px 0;
        font-size: 30px;
        font-weight: 400;
    }
    &__navigation {
        padding: 15px 30px;
        box-shadow: 3px 3px 10px $middle-blue;
        border-radius: 5px;
    }
}

```

```

display: flex;
flex-direction: column;
width: 1000px;
}
&__list {
list-style: none;
&-title {
font-size: 22px;
color: $middle-blue;
text-align: center;
padding: 5px;
margin-bottom: 5px;
font-weight: 400;
}
&-item {
a {
color: $middle-blue;
font-size: 18px;
padding: 5px 0;
margin-bottom: 5px;
display: flex;
align-items: center;
text-decoration: none;
border: 1px solid rgba(0, 0, 0, 0.02);
box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
border-radius: 5px;
&:hover {
cursor: pointer;
background-color: $middle-blue;

```

```

        color: $white;
        transition: all 0.3s;
        border-radius: 5px;
        fill: $white;
    }
}

}

}

}

&__btns {
    display: flex;
    flex-direction: column;
}

&__btn {
    margin: 5px 0;
    border: none;
    border-radius: 5px;
    padding: 5px 5px;
    color: $middle-blue;
    text-transform: uppercase;
    font-size: 16px;
    outline: none;
    &:hover {
        color: $white;
        cursor: pointer;
        background-color: $middle-blue;
        transition: all 0.3s;
    }
}

}

```

```

.article-area {
    max-width: 50%;
}

.handbook-icon {
    height: 25px;
    display: inline-block;
    padding-right: 5px;
    &:hover {
        fill: $white;
    }
}

.article {
    display: flex;
    flex-direction: column;
    align-items: center;
    max-width: 968px;
    margin: 25px auto;
    border: 1px solid rgba(0, 0, 0, 0.02);
    box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
    border-radius: 5px;
    padding: 10px 30px;
    &__title {
        font-size: 30px;
        color: $middle-blue;
        font-weight: 400;
    }
    &__text {
        text-align: center;
    }
}

```

```

        font-size: 18px;
        color: $dark-blue;
        letter-spacing: 1px;
    }
}
@media (max-width: 1050px) {
    .handbook {
        &__title {
            font-size: 24px;
            padding: 20px 0;
        }
        &__navigation {
            width: 500px;
        }
    }
    .handbook__list-item {
        span {
            display: none;
        }
    }
    .article {
        max-width: 768px;
    }
}
@media (max-width: 800px) {
    .article {
        max-width: 568px;
    }
}
@media (max-width: 620px) {

```

```

.handbook {
    &__navigation {
        width: 300px;
    }
}

.article {
    max-width: 380px;
}
}

@media (max-width: 520px) {
    .handbook {
        &__navigation {
            width: 300px;
        }
    }
}

```

Файл Home.scss

```

@import 'vars';

.home-page {
    background: $white;
    min-height: 100vh;

    .header {
        box-shadow: 1px 1px 3px $middle-blue;
        padding: 5px 0;
        background: $middle-blue;
        &__logo {
            font-size: 24px;
            color: $white;
        }
    }
}

```

```
        font-style: italic;
        padding: 0 10px;
    }

    &__container {
        display: flex;
        justify-content: space-between;
        align-items: center;
        max-width: 1600px;
        margin: 0 auto;
    }

    &__navigation {
        &-list {
            margin: 5px 0;
        }

        &-item {
            text-decoration: none;
            color: $white;
            background: $middle-blue;
            font-size: 22px;
            margin: 0 10px;
            border: none;
            padding: 5px 10px;

            &:hover {
                background: $white;
                color: $middle-blue;
                transition: 0.3s;
            }
        }
    }
}
```

```

        cursor: pointer;
        border-radius: 5px;
    }
}
}

&__burger {
    position: absolute;
}
}
}

```

```

.menu__navigation {
    min-height: 100vh;

    &-list {
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
    }

    &-item {
        text-decoration: none;
        color: $white;
        font-size: 18px;
        margin: 10px 0;
        border: none;
        padding: 10px 0;
        background: $middle-blue;
    }
}

```

```

    }
}

@media (max-width: 800px) {
    .header {
        &__burger {
            display: block;
            width: 22px;
            height: 22px;
            cursor: pointer;
            margin-right: 15px;
            right: 0;

            &.active:before {
                transform: rotate(45deg);
                top: 10px;
            }

            &.active:after {
                transform: rotate(-45deg);
                bottom: 10px;
            }

            &.active span {
                transform: scale(0);
            }

            &:before {
                content: "";
                position: absolute;

```

```
        top: 0;
        width: 22px;
        height: 2px;
        background: $white;
        transition: 0.3s;
    }

    &:after {
        content: "";
        position: absolute;
        bottom: 0;
        width: 22px;
        height: 2px;
        background: $white;
        transition: 0.3s;
    }

    span {
        position: absolute;
        top: 10px;
        width: 22px;
        height: 2px;
        background: $white;
        transition: 0.3s;
    }
}

&__navigation {
    display: none;
}
```

```

    }
    .menu__navigation-item {
        font-size: 20px;
    }
}

```

Файл **SelectAgriculture.scss**

```

@import 'vars';

.select-agriculture {
    height: calc(100vh - 63px);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    &__inner {
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
        border-radius: 5px;
        padding: 20px 10px;
        margin: 0 10px;
    }
    &__title {
        color: $middle-blue;
        letter-spacing: 1px;
        padding: 0 0 10px 0;
    }
}

```

```

        font-weight: 400;
    }
    &__options {
        width: 400px;
    }
    &__option {
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
        border-radius: 5px;
        padding: 10px;
        display: flex;
        justify-content: space-between;
        align-items: center;
        margin: 10px 0;
        label {
            font-size: 18px;
            color: $middle-blue;
            letter-spacing: 1px;
            width: 300px;
        }
        span {
            border: 1px solid $middle-blue;
            border-radius: 50%;
            height: 21px;
            width: 21px;
        }
        .checkmark {
            width: 15px;
            height: 15px;

```

```

        background-color: $middle-blue;
        border-radius: 50%;
        display: none;
        position: relative;
        left: 1.5px;
        bottom: 1.5px;
    }
    input {
        display: none;
    }
    & input:checked + .checkmark {
        display: inline-block;
    }
}

&__next-page {
    margin: 15px 0 20px 0;
    a {
        border: 1px solid $middle-blue;
        padding: 6px 30px;
        border-radius: 5px;
        margin-left: 10px;
        font-size: 18px;
        color: $middle-blue;
        text-decoration: none;
        &:hover {
            cursor: pointer;
            background-color: $middle-blue;
            color: $white;
            transition: all 0.3s;
        }
    }
}

```

```

        }
    }
    &__info {
        font-size: 16px;
        letter-spacing: 1px;
        color: $middle-blue;
    }
}

```

```

@media (max-width: 820px) {
    .select-agriculture__info {
        text-align: center;
    }
}

```

Файл Slider.scss

```

@import 'vars';

.swiper-slide {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    height: 90vh;
    img {
        padding: 5px;
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
            rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
        border-radius: 5px;
    }
}

```

```

h2 {
    color: $middle-blue;
    font-weight: 400;
    padding: 10px;
}
div {
    color: $middle-blue;
    font-size: 18px;
    margin-bottom: 10px;
}
}

```

Файл UserPlans.scss

```

@import 'vars';

.user-plans {
    &__title {
        text-align: center;
        color: $middle-blue;
        font-weight: 400;
        padding: 15px 0 10px 0;
    }
    &__inner {
        display: flex;
        flex-direction: column;
        align-items: center;
    }

    &__detail {
        border: 1px solid rgba(0, 0, 0, 0.02);
        box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,

```

```

        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
border-radius: 5px;
padding: 5px;
}
&__plan {
border: 1px solid rgba(0, 0, 0, 0.02);
box-shadow: rgba(0, 0, 0, 0.02) 0px 1px 3px 0px,
        rgba(27, 31, 35, 0.15) 0px 0px 0px 1px;
border-radius: 5px;
padding: 20px 40px;
margin: 5px;
}
&__details {
display: grid;
grid-template-columns: 1fr 1fr 1fr;
}
&__plan-title {
color: $middle-blue;
font-size: 16px;
letter-spacing: 1px;
}
}

```

Файл areaSlice.ts

```

import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { IAreaState } from '../types/IArea';

const initialState: IAreaState = {
  region: "",
  agriculture: "",

```

```

    squareSize: 0,
    cadastralNumber: 0
  };

export const areaSlice = createSlice({
  name: 'area',
  initialState,
  reducers: {
    addRegion(state, action: PayloadAction<string>) {
      state.region = action.payload;
    },
    addAgriculture(state, action: PayloadAction<string>) {
      state.agriculture = action.payload;
    },
    addCadastralNumber(state, action: PayloadAction<number>) {
      state.cadastralNumber = action.payload;
    },
    addAreaSquare(state, action: PayloadAction<number>) {
      state.squareSize = action.payload;
    }
  }
});

```

```
export const areaReducer = areaSlice.reducer;
```

```
export const { addRegion, addAgriculture, addCadastralNumber, addAreaSquare } =
  areaSlice.actions;
```

Файл articlesSlice.ts

```

import { IArticles, IArticlesState } from '../types/IArticles';
import { createSlice, PayloadAction } from '@reduxjs/toolkit';

```

```

const initialState: IArticlesState = {
  articles: [],
  title: ''
};

export const articlesSlice = createSlice({
  name: 'articles',
  initialState,
  reducers: {
    getAllArticles(state, action: PayloadAction<IArticles[]>) {
      state.articles = action.payload;
    },
    getArticle(state, action: PayloadAction<string>) {
      state.title = action.payload;
    }
  }
});

```

```

export const articlesReducer = articlesSlice.reducer;

```

```

export const { getAllArticles, getArticle } = articlesSlice.actions;

```

Файл calculationsSlice.ts

```

import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { ICalculateState } from '../types/ICalculate';

```

```

const initialState: ICalculateState = {
  wheat: {
    t: 70,
    n1: 10,

```

```
p: 20,  
s: 20,  
c: 28,  
g: 1000,  
f: 800,  
m: 350,  
i: 1000,  
k: 34  
,  
corn: {  
  t: 70,  
  n1: 5000,  
  p: 20,  
  s: 20,  
  c: 28,  
  g: 1000,  
  f: 800,  
  m: 350,  
  i: 1000,  
  k: 34,  
  x: 400  
,  
sunflower: {  
  t: 70,  
  n1: 3000,  
  p: 20,  
  s: 20,  
  c: 28,  
  g: 1000,  
  f: 800,
```

```

        m: 350,
        i: 1000,
        k: 34
    },
    totalPrice: {
        totalPrice: 0,
        DT: 0,
        seed: 0,
        fertilizers: 0,
        sulfate: 0,
        saltpeter: 0,
        herbicide: 0,
        fungicide: 0,
        microMinerals: 0,
        insecticide: 0,
        urea: 0,
        zinc: 0
    }
};

export const calculationsSlice = createSlice({
  name: 'calculation',
  initialState,
  reducers: {
    calculateWheatPrice(state, action: PayloadAction<number>) {
      state.totalPrice.totalPrice =
        action.payload *
        (20 * state.wheat.t +
          5 * state.wheat.t +
          3 * state.wheat.t +

```

```

(200 * state.wheat.n1 + 150 * state.wheat.p) +
100 * state.wheat.s +
150 * state.wheat.c +
(0.5 * state.wheat.g +
    0.15 * state.wheat.i +
    0.2 * state.wheat.m +
    state.wheat.t) +
(0.5 * state.wheat.f +
    0.15 * state.wheat.i +
    state.wheat.t) +
(0.5 * state.wheat.f + 12 * state.wheat.k +
state.wheat.t) +
    12 * state.wheat.t);
},
calculateCornPrice(state, action: PayloadAction<number>) {
    state.totalPrice.totalPrice =
        action.payload *
        (150 * state.corn.p +
            state.corn.t +
            20 * state.corn.t +
            (200 * state.corn.p + state.corn.t) +
            4 * state.corn.t +
            5 * state.corn.t +
            (3 * state.corn.t + state.corn.n1) +
            (1.2 * state.corn.g + 0.2 * state.corn.m + state.corn.t)
+
            (0.5 * state.corn.x + 0.15 * state.corn.i + state.corn.t)
+
            12 * state.corn.t);
},

```

```

calculateSunflowerAndSoyPrice(state, action: PayloadAction<number>)
{
    state.totalPrice.totalPrice =
        action.payload *
        (150 * state.corn.p +
            20 * state.corn.t +
            (200 * state.corn.p + state.corn.t) +
            4 * state.corn.t +
            5 * state.corn.t +
            (3 * state.corn.t + state.corn.n1) +
            (1.2 * state.corn.g + 0.2 * state.corn.m + state.corn.t)
+
            (0.5 * state.corn.x + 0.15 * state.corn.i + state.corn.t)
+
            12 * state.corn.t);
},
showWheatDetails(state, action: PayloadAction<number>) {
    state.totalPrice.DT =
        action.payload *
        (20 * state.wheat.t +
            5 * state.wheat.t +
            3 * state.wheat.t +
            state.wheat.t +
            state.wheat.t +
            state.wheat.t);
    state.totalPrice.seed = action.payload * (200 * state.wheat.n1);
    state.totalPrice.fertilizers =
        action.payload * (150 * state.wheat.p);
    state.totalPrice.sulfate = action.payload * (100 * state.wheat.s);
    state.totalPrice.saltpeter = action.payload * (150 * state.wheat.c);
}

```

```

state.totalPrice.herbicide = action.payload * (0.5 * state.wheat.g);
state.totalPrice.fungicide =
    action.payload * (0.5 * state.wheat.f + 0.5 * state.wheat.f);
state.totalPrice.microMinerals =
    action.payload * (0.2 * state.wheat.m);
state.totalPrice.insecticide =
    action.payload * (0.15 * state.wheat.i + 0.15 *
state.wheat.i);

state.totalPrice.urea = action.payload * (12 * state.wheat.k);
},
showCornDetails(state, action: PayloadAction<number>) {
    state.totalPrice.DT =
        action.payload *
        (state.corn.t +
            20 * state.corn.t +
            state.corn.t +
            4 * state.corn.t +
            5 * state.corn.t +
            3 * state.corn.t +
            state.corn.t +
            state.corn.t +
            12 * state.corn.t);
state.totalPrice.seed = action.payload * state.corn.n1;
state.totalPrice.fertilizers =
    action.payload * (150 * state.corn.p + 200 * state.corn.p);
state.totalPrice.herbicide = action.payload * (1.2 * state.corn.g);
state.totalPrice.microMinerals =
    action.payload * (1.2 * state.corn.m);
state.totalPrice.insecticide =
    action.payload * (0.15 * state.corn.i);

```

```

        state.totalPrice.zinc = action.payload * (0.5 * state.corn.x);
    },
    showSunflowerAndSoyDetails(state, action: PayloadAction<number>) {
        state.totalPrice.DT =
            action.payload *
            (state.corn.t +
                20 * state.corn.t +
                state.corn.t +
                4 * state.corn.t +
                5 * state.corn.t +
                3 * state.corn.t +
                state.corn.t +
                state.corn.t +
                12 * state.corn.t);
        state.totalPrice.seed = action.payload * state.corn.n1;
        state.totalPrice.fertilizers =
            action.payload * (150 * state.corn.p + 200 * state.corn.p);
        state.totalPrice.herbicide = action.payload * (1.2 * state.corn.g);
        state.totalPrice.microMinerals =
            action.payload * (1.2 * state.corn.m);
        state.totalPrice.insecticide =
            action.payload * (0.15 * state.corn.i);
        state.totalPrice.zinc = action.payload * (0.5 * state.corn.x);
    }
}

});

export const calculationsReducer = calculationsSlice.reducer;

export const {

```

```

    calculateWheatPrice,
    calculateCornPrice,
    calculateSunflowerAndSoyPrice,
    showWheatDetails,
    showCornDetails,
    showSunflowerAndSoyDetails
  } = calculationsSlice.actions;

```

Файл headerBurgerSlice.ts

```

import { createSlice } from '@reduxjs/toolkit';

import { IBurgerModal } from '../types/IBurgerModal';

const initialState: IBurgerModal = {
  openModal: false,
  burgerClassName: 'header__burger',
  active: false
};

export const headerBurgerSlice = createSlice({
  name: 'burgerModalWindow',
  initialState,
  reducers: {
    handleModal(state) {
      state.openModal = !state.openModal;
      state.active = !state.active;
      state.burgerClassName =
        ? (state.burgerClassName = 'header__burger active')
        : (state.burgerClassName = 'header__burger');
    }
  }
});

```

```
});
```

```
export const headerBurgerReducer = headerBurgerSlice.reducer;
```

```
export const { handleModal } = headerBurgerSlice.actions;
```

Файл loginFormSlice.ts

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
```

```
import { ILogin } from '../types/ILogin';
```

```
const initialState: ILogin = {
```

```
  email: "",
```

```
  password: ""
```

```
};
```

```
export const loginFormSlice = createSlice({
```

```
  name: 'login',
```

```
  initialState,
```

```
  reducers: {
```

```
    changeEmailValue(state, action: PayloadAction<string>) {
```

```
      state.email = action.payload;
```

```
    },
```

```
    changePasswordValue(state, action: PayloadAction<string>) {
```

```
      state.password = action.payload;
```

```
    },
```

```
    clearFields(state) {
```

```
      state.email = "";
```

```
      state.password = "";
```

```
    }
```

```
  }
```

```
});
```

```
export const loginFormReducer = loginFormSlice.reducer;
```

```
export const { changeEmailValue, changePasswordValue, clearFields } =  
  loginFormSlice.actions;
```

Файл regionsSlice.ts

```
import { IRegionState, IRegions } from '../types/IRegions';
```

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
```

```
const initialState: IRegionState = {  
  regions: [],  
  searchedRegion: [],  
  searchActive: false  
};
```

```
export const regionsSlice = createSlice({  
  name: 'regions',  
  initialState,  
  reducers: {  
    getAllRegions(state, action: PayloadAction<IRegions[]>) {  
      state.regions = action.payload;  
    },  
    searchRegion(state, action: PayloadAction<IRegions[]>) {  
      state.searchActive = true;  
      state.searchedRegion = action.payload;  
    },  
    showAllRegionsAfterClearInput(state) {
```

```

        state.searchActive = false;
    }
}
});

```

```
export const regionsReducer = regionsSlice.reducer;
```

```
export const { getAllRegions, searchRegion, showAllRegionsAfterClearInput } =
    regionsSlice.actions;
```

Файл registrationFormSlice.ts

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
```

```
import { IRegistrationState } from '../types/IRegistration';
```

```
const initialState: IRegistrationState = {
    email: "",
    username: "",
    password: ""
};
```

```
export const registrationFormSlice = createSlice({
    name: 'registration',
    initialState,
    reducers: {
        changeEmailValue(state, action: PayloadAction<string>) {
            state.email = action.payload;
        },
        changeUsernameValue(state, action: PayloadAction<string>) {
            state.username = action.payload;
        },
    },
});
```

```

        changePasswordValue(state, action: PayloadAction<string>) {
            state.password = action.payload;
        },
        clearFields(state) {
            state.email = "";
            state.username = "";
            state.password = "";
        }
    }
});

```

```
export const registrationFormReducer = registrationFormSlice.reducer;
```

```

export const {
    changeEmailValue,
    changeUsernameValue,
    changePasswordValue,
    clearFields
} = registrationFormSlice.actions;

```

Файл UserSlice.ts

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
```

```
import { IUser, IUserState } from '../types/IUser';
```

```

const initialState: IUserState = {
    currentUser: {
        id: "",
        email: "",
        username: "",
        areas: []
    }
}

```

```

    },
    isAuth: false,
    isLoading: false,
    error: ""
  };

export const userSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    authUser(state, action: PayloadAction<IUser>) {
      state.currentUser = action.payload;
      state.isAuth = true;
    },
    logoutUser(state) {
      localStorage.removeItem('token');
      state.currentUser.id = "";
      state.currentUser.email = "";
      state.currentUser.username = "";
      state.isAuth = false;
    },
    onLoading(state) {
      state.isLoading = true;
    },
    offLoading(state) {
      state.isLoading = false;
    }
  }
});

```

```
export const userReducer = userSlice.reducer;
export const { authUser, onLoading, offLoading, logoutUser } =
  userSlice.actions;
```

Файл store.ts

```
import { combineReducers, configureStore } from '@reduxjs/toolkit';

import { userReducer } from './reducers/userSlice';
import { registrationFormReducer } from './reducers/registrationFormSlice';
import { loginFormReducer } from './reducers/loginFormSlice';
import { headerBurgerReducer } from './reducers/headerBurgerSlice';
import { articlesReducer } from './reducers/articlesSlice';
import { regionsReducer } from './reducers/regionsSlice';
import { areaReducer } from './reducers/areaSlice';
import { calculationsReducer } from './reducers/calculationsSlice';

const rootReducer = combineReducers({
  userReducer,
  registrationFormReducer,
  loginFormReducer,
  headerBurgerReducer,
  articlesReducer,
  regionsReducer,
  areaReducer,
  calculationsReducer
});

export const setupStore = () => {
  return configureStore({
    reducer: rootReducer
```

```
    });  
  };  
};
```

```
export type RootState = ReturnType<typeof rootReducer>;  
export type AppStore = ReturnType<typeof setupStore>;  
export type AppDispatch = AppStore['dispatch'];
```

Файл index.ts

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
  
import { BrowserRouter } from 'react-router-dom';  
import { Provider } from 'react-redux';  
  
import { setupStore } from './store/store';  
  
import { App } from './components/Static/App';  
  
const store = setupStore();  
  
const root = ReactDOM.createRoot(  
  document.getElementById('root') as HTMLElement  
)  
;  
  
root.render(  
  <Provider store={store}>  
    <BrowserRouter>  
      <App />  
    </BrowserRouter>  
  </Provider>
```

```
);
```

Файл auth.middleware.js

```
const jwt = require('jsonwebtoken');
const config = require('config');

module.exports = (req, res, next) => {
  if (req.method === 'OPTIONS') {
    return next();
  }

  try {
    const token = req.headers.authorization.split(' ')[1];
    if (!token) {
      return res.status(401).json({ message: 'Auth error' });
    }
    const decoded = jwt.verify(token, 'agro-secret'
  ));
    req.user = decoded;
    next();
  } catch (e) {
    return res.status(401).json({ message: 'Auth error' });
  }
};
```

Файл cors.middleware.js

```
function cors(req, res, next) {
```

```

    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Methods', 'GET, PUT, PATCH, POST,
DELETE');
    res.header('Access-Control-Allow-Headers', 'Content-Type, Authorization');
    next();
}

```

```
module.exports = cors;
```

Файл Articles.js

```
const { Schema, model } = require('mongoose');
```

```

const Articles = new Schema({
  title: { type: String, required: true },
  body: { type: String }
});

```

```
module.exports = model('Article', Articles);
```

Файл Regions.js

```
const { Schema, model } = require('mongoose');
```

```

const Regions = new Schema({
  regionName: { type: String, required: true }
});

```

```
module.exports = model('Regions', Regions);
```

Файл User.js

```
const { Schema, model, ObjectId } = require('mongoose');
```

```

const User = new Schema({
  email: { type: String, required: true, unique: true },

```

```

    username: { type: String, required: true },
    password: { type: String, required: true },
    areas: []
  });

```

```

module.exports = model('User', User);

```

Файл areaPlan.route.js

```

const Router = require('express');
const { db } = require('../models/User');

const User = require('../models/User');

const router = new Router();

router.post('/area-plan', async (req, res) => {
  try {
    const planData = req.body;
    const user = await User.findOne({ _id: planData._id });
    await user.areas.push({ planData });
    console.log(user.areas);
    await user.save();
    return res.json({
      message: 'План додано!'
    });
  } catch (e) {
    console.log(e);
    res.send({ message: 'Server error' });
  }
});

```

```
module.exports = router;
```

Файл articles.route.js

```
const Router = require('express');
```

```
const Articles = require('../models/Articles');
```

```
const router = new Router();
```

```
router.get('/articles', async (req, res) => {
  try {
    const articles = await Articles.find();
    return res.json({
      articles
    });
  } catch (event) {
    console.log(event);
    res.send({ message: 'Server error' });
  }
});
```

```
module.exports = router;
```

Файл auth.route.js

```
const Router = require('express');
```

```
const config = require('config');
```

```
const jwt = require('jsonwebtoken');
```

```
const User = require('../models/User');
```

```
const authMiddleware = require('../middleware/auth.middleware');
```

```
const router = new Router();
```

```

router.get('/auth', authMiddleware, async (req, res) => {
  try {
    const user = await User.findOne({ _id: req.user.id });
    const token = jwt.sign({ id: user.id }, config.get('secretKey'), {
      expiresIn: '1h'
    });
    return res.json({
      token,
      user: {
        id: user.id,
        email: user.email,
        areas: user.areas
      }
    });
  } catch (e) {
    console.log(e);
    res.send({ message: 'Server error' });
  }
});

```

```
module.exports = router;
```

Файл login.route.js

```

const Router = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const config = require('config');

const User = require('../models/User');

```

```

const router = new Router();

router.post('/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(404).json({ message: 'User not found' });
    }
    const isPasswordValid = bcrypt.compareSync(password, user.password);
    if (!isPasswordValid) {
      return res.status(400).json({ message: 'Invalid password' });
    }
    const token = jwt.sign({ id: user.id }, 'agro-secret'
{
    expiresIn: '1h'
  });
    return res.json({
      token,
      user: {
        id: user.id,
        email: user.email,
        username: user.username,
        areas: user.areas
      }
    });
  } catch (event) {
    console.log(event);
    res.send({ message: 'Server error' });
  }
}

```

```
});
```

```
module.exports = router;
```

Файл regions.route.js

```
const Router = require('express');
```

```
const Regions = require('../models/Regions');
```

```
const router = new Router();
```

```
router.get('/regions', async (req, res) => {
  try {
    const regions = await Regions.find();
    return res.json({
      regions
    });
  } catch (event) {
    console.log(event);
    res.send({ message: 'Server error' });
  }
});
```

```
module.exports = router;
```

Файл registration.route.js

```
const Router = require('express');
```

```
const bcrypt = require('bcryptjs');
```

```
const { check, validationResult } = require('express-validator');
```

```
const User = require('../models/User');
```

```

const router = new Router();

router.post(
  '/registration',
  [
    check('email', 'Incorrect email').isEmail(),
    check(
      'username',
      'Password must be longer than 2 and shorter than 10'
    ).isLength({ min: 2, max: 10 }),
    check(
      'password',
      'Password must be longer than 3 and shorter than 12'
    ).isLength({ min: 3, max: 12 })
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        return res
          .status(400)
          .json({ message: 'Incorrect request', errors });
      }

      const { email, username, password } = req.body;
      const candidate = await User.findOne({ email });
      if (candidate) {
        return res.status(400).json({
          message: `User with email ${email} already exist`
        });
      }
    }
  }
);

```

```

    }
    const hashPassword = await bcrypt.hash(password, 8);

    const user = new User({ email, username, password:
hashPassword });
    await user.save();

    return res.json({
        message:
            'Ви успішно зареєструвались. Пройдіть
авторизацію з цими даними!'
    });
  } catch (e) {
    console.log(e);
    res.send({ message: 'Server error' });
  }
}
);

```

```
module.exports = router;
```

Файл index.js

```

const express = require('express');
const mongoose = require('mongoose');
const config = require('config');

const registrationRouter = require('./src/routes/registration.route');
const loginRouter = require('./src/routes/login.route');
const authRouter = require('./src/routes/auth.route');
const articlesRouter = require('./src/routes/articles.route');

```

```

const regionsRouter = require('./src/routes/regions.route');
const areaPlanRouter = require('./src/routes/areaPlan.route');
const corsMiddleware = require('./src/middleware/cors.middleware');

const app = express();

app.use(express.json());

app.use(corsMiddleware);

app.use('/api/auth', registrationRouter);
app.use('/api/auth', loginRouter);
app.use('/api/auth', authRouter);
app.use('/api/', articlesRouter);
app.use('/api/', regionsRouter);
app.use('/api/', areaPlanRouter);

const runServer = async () => {
  try {
    await
mongoose.connect('mongodb+srv://solodok00:3OEKWxdwy7gEmjZB@diploma.ux
w9goz.mongodb.net/'
);
    app.listen(5000, () => {
      console.log('Server has been started on port',5000);
    });
  } catch (event) {}
};

runServer();

```