МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АГРАРНИХ ПІДПРИЄМСТВ

Здобувач освіти гр. ІН – 82 Євгенія ПАВЛЕНКО

Науковий керівник,
кандидат фізико-математичних наук,
доцент Олена ПРОЦЕНКО

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-82 спеціальності «122 – Комп'ютерні науки» денної форми навчання Павленко Євгенії Сергіївни.

Тема: «ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АГРАРНИХ ПІДПРИЄМСТВ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) методика вирішення поставлених задач; 3) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____Олена ПРОЦЕНКО
Завдання прийняв до виконання _____ Євгенія ПАВЛЕНКО

РЕФЕРАТ

Записка: 45 стор., 22 рис., 1 додаток, 15 джерел.

Об'єкт дослідження — веб-додатки.

Мета роботи — дослідження і розробка методів створення веб-додатків. Розробка додатку для обчислення вартості обробки земельної ділянки різних розмірів.

Результати — розроблено веб-додаток, за допомогою React, NodeJS, Express, MongoDB та TypeScript. Додаток виконує обчислення вартості обробітку земельних ділянок в залежності від вибраної агрокультури та розміру ділянки.

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АГРАРНИХ ПІДПРИЄМСТВ

ЗМІСТ

ВСТУП

Щодня, щогодини, щохвилини у світі розвиваються технології, швидкість розвитку настільки велика що ми це навіть не помічаємо. Поки ти спиш в одному куточку Землі в іншому хтось вже розробляє щось нове. Такими стали реалії сучасного світу.

Оскільки в нашому житті все частіше й частіше на зміну людській праці приходять комп'ютери, то ця тенденція не оминула аграріїв. Функції, які виконували кілька людей, може замінити веб-додаток. Такий підхід заощаджує не тільки час а й гроші.

Технології які начебто вирішують одну проблему насправді мають безліч корисних функцій та прорахунків. Наприклад існує безліч веб помічників в аграрній галузі, але щоб повністю володіти знаннями по обробітку землі необхідно отримати професійну освіту/пройти курси чи довго читати статті.

Веб-додаток - це будь-яка комп'ютерна програма, яка виконує певну функцію, використовуючи веб-браузер у якості свого клієнта. Додаток може бути таким же простим, як дошка оголошень або контактна форма на веб-сайті, або настільки ж складний, як текстовий процесор або багатокористувацький мобільний ігровий додаток, який ви завантажуєте на телефон.

Веб-додатки існують з тих пір, поки всесвітня павутина не набула популярності. Наприклад, Ларрі Уолл розробив Perl, популярну мову сценаріїв на стороні сервера, у 1987 році. Це було за сім років до того, як Інтернет дійсно почав набирати популярність поза академічними та технологічними колами.

Перші основні веб-додатки були відносно простими, але наприкінці 90-х років відбувся поштовх до більш складних веб-додатків. У наш час мільйони американців використовують веб-додаток для виконання багатьох завдань в Інтернеті.

Актуальність роботи полягає в тому, що аграрні підприємства та фермери новачки потребують додаток, котрий буде висококваліфікованим помічником, довідником й нотатком. Створений додаток фіксуватиме дані по виконаних роботах, витрачених коштам та використаних матеріалів. Важливою функцією буде прорахунок майбутніх витрат та можливий прибуток, з урахуванням витрачених матеріалів та ринку збуту.

Метою роботи є розробка веб-додатку розрахунку та підтримки повного циклу обробітку земельної ділянки з обчисленням площі ділянки, визначенням регіону й агрокультури, детальними інструкціями плану робіт, вирахуванням прибутку та рекомендаціями.

1. ЛІТЕРАТУРНИЙ ОГЛЯД ЗА ОБРАНОЮ ТЕМАТИКОЮ РОБОТИ

1.1. Інструменти для розробки веб-додатків

Для побудови веб-додатків можна використовувати різноманітній стек технологій, який залежить від конкретної задачі та може бути адаптований під конкретну задачу.

Наразі є можливість створювати веб додатки, використовуючи мови програмування, окрім JavaScript. Найпершим прикладом можна назвати Python. Але все ж, поки що сформувались інші фаворити.

При розробці клієнтської частини додатку, перевагу надають 3 інструментам – React.js, Angular, Vue.js. Vue та Angular є фреймворками, React, у свою чергу, бібліотека. Різниця полягає у тому, що більшість додаткових інструментів вже додана у фреймворки «із коробки», а у випадках з бібліотеками, додаткові модулі, наприклад, для побудови маршрутів у додатку, потрібно додатково встановлювати за допомогою npm або yarn.

При розробці серверної частини додатку, є можливість використовувати більш широкий спектр технологій. Наприклад, використати Java або C#, бо ці інструменти дуже давно на ринку, але максимально актуальні і зараз. Також, можна використати інструменти на базі JavaScript, щоб і клієнтська і серверна частина була написана однією мовою. У цьому випадку, найчастіше використовується Node.js та його легкий фреймворк Express. Також популярність набрав Nest, який написаний на базі TypeScript, завдяки чому, розробка, для команди, яка має досвід з цим інструментом дуже полегшується.

Також перед розробкою кожного продукту постає питання про базу даних, яка буде використана. Є декілька видів БД, але найуживанішим нині варіантом є реляційні БД, наприклад MySQL, Oracle DB, PostgreSQL. Також набирають популярність бази даних, які побудовані на основі документів, а не таблиць. Такі бази краще використовувати у менш навантажених системах.

1.2 Методи розробки клієнтської частини додатку зі сторони розробника

Якщо дивитися зі сторони користувача, то основним завданням клієнтської частини додатку є якісне відображення даних, які будуть надаватись від серверної частини. Але якщо дивитися, зі сторони команді розробки, то головне завдання – розробити додаток таким чином, щоб у майбутньому його було легко підтримувати.

У сучасній Frontend-розробці, є декілька шляхів для виконання цієї умови.

Перше, та, мабуть, основне – це розділення структури відображення контенту та структуру обробки інформації. В цілому, основним завдання компонентів у сучасних фреймворках та бібліотеках є відображення розмітки. І це єдине, чим вони і повинні займатись у ідеальній ситуації. Звичайно, так виходить не завжди, але потрібно до цього прагнути.

Для того, щоб відділити інформацію та керування логікою від компонентів, використовують спеціальні утиліли для управління станом додатку. Найвідомішим прикладом є Redux. Його аналогами слугують MobX, Effector. Більш просунутою версією є Redux Toolkit.

Суть таких утиліт полягає у тому, що вони дозволять повність відділити логіки по обробці даних та передавати результат обробки тільки у ті компоненти додатку, де вони необхідні.

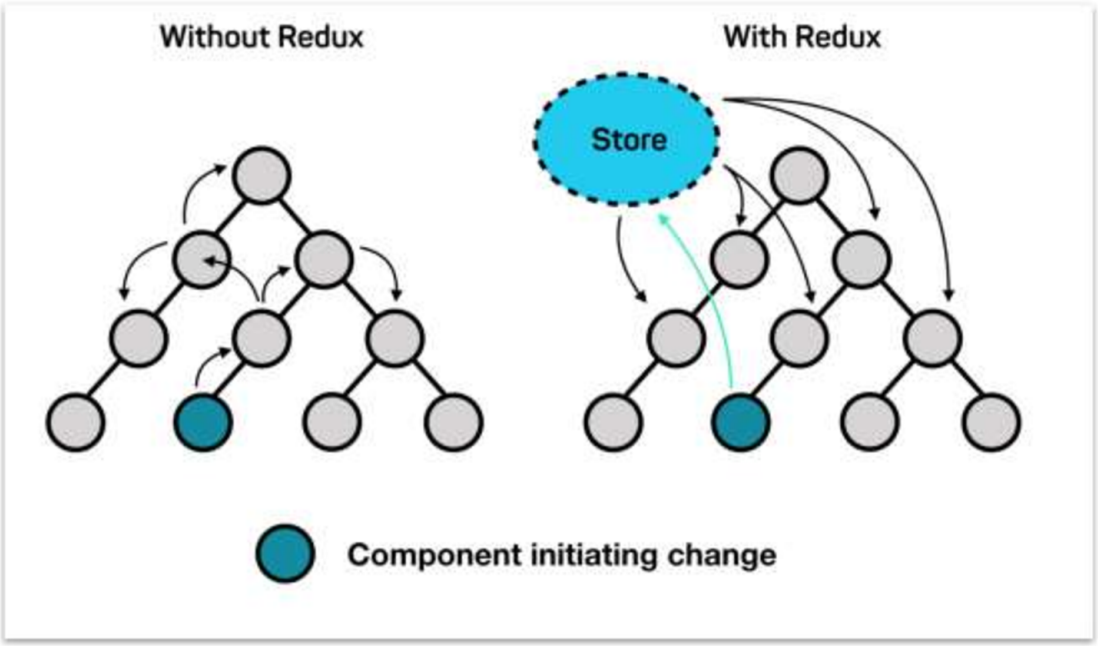

Рисунок 1.1 — Додаток з Redux та без нього

На рисунку 1.1 наглядно продемонстровані основні плюси. Приблизно так працюють і аналоги. [1]

Ще одним інструментом для досягнення цілі є TypeScript. За замовчуванням, JavaScript має динамічну типізацію, що несе за собою ряд основних недоліків мови.

Наприклад, коли розробник надсилає запит на сервер, відповідь, яка йому приходить можна назвати «чорною скринькою». Він не розуміє, там має бути об’єкт або масив, які повинні бути поля та типи даних. Відповідно, при спробі обробити відповідь, велика вірогідність отримати помилку та витратити багато часу для її вирішення. Цю проблему як раз і вирішує TypeScript.

Єдиним недоліком є лише те, що це по суті додатковий проміжок, який потім все одно компілюється у JavaScript. Але досвід багатьох додатків, каже про те, що краще витратити трохи часу на додаткову типізацію, ніж потім витратити ще більше часу на виправлення непередбачуваних помилок додатку. [2]

1.3 Проблеми інтегрування TypeScript.

За роки свого існування TypeScript встиг завоювати велику популярність. Зараз складно знайти більш-менш великий проект, команда якого використовує для розробки фреймворк без інтегрування типізації. Більше того, у фреймворку Angular, який розробляє та підтримує компанія Google, TS інтегрований вже з початку. Інші популярні фреймворки та бібліотеки дозволяють обрати, писати додаток на чистому JavaScript або додавати сувору типізацію.

Якщо брати за приклад бібліотеку React, то вона є прикладом коли, розробники, а саме, компанія Meta дозволяють вибирати самостійно. Раніше, до популярності TypeScript, у React була можливість додавати сувору типізацію за допомогою PropTypes. Проблема була у тому, що типізувати можна було лише так звані props. Якщо, казати просто, то props – це об'єкт, який можна передати від батьківського компонента до дочірнього. Іноді пропсів буває багато, тому виникала необхідність додавати типи, щоб увесь об'єкт не був так званим «чорним ящиком».

У цьому плані, TypeScript набагато гнучкіший. Ми можемо додавати типу у буквальному сенсі слова до всього, до чого тільки можливо. Є можливість типізувати пропси, більш зручнішим способом, написавши для них окремий interface, також можна типізувати функції, вказуючи, який тип вона приймає як параметр і що вона повертає. Додана можливість типізації сторонніх бібліотек, axios, react-router-dom та інші.

На сьогоднішній день, лише невелика TypeScript набирає все більшу і більшу популярність. Увесь час, який витрачається на типізацію, потім окупається під час підтримки додатку та інтегрування нових розробників у проект. Актуальність TS гарантує те, що його розробила та підтримує компанія Microsoft. Можна бути впевненим, що TypeScript затримається на ринку IT дуже надовго, тому що нині немає альтернатив, які можуть повноцінно і, головне, більш зручно замінити його. Тому зараз і простежується

одноосібна гегемонія TS, що, насправді, приносить за собою набагато більше плюсів, ніж мінусів.

1.4 Порівняння Express.js та Nest.js – найпопулярніших фреймворків для розробки серверної частини

Node або Node.js – програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення.

Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду.

Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel і espruino).

В основі Node.js лежить асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням. [3]

NestJS-це той фреймворк, створений для регулювання життя розробника, користуючись правильними архітектурними підходами та диктуючим свої правила. Тому NestJS — це не тільки фреймворк для бекенда, але й можливість переходу в світ передових концепцій, наприклад, таких, як DDD, пошук подій та мікросервісна архітектура. Все запаковано в простій і легкій формі, так що ви вибираєте, що використовувати: всю платформу або просто її компоненти. NestJS, фреймворк, який повністю написаний на TypeScript, він легко тестується та містить все необхідне. [4]

Express – це популярна, швидка, мінімальна та гнучка рамка Node.js Model-View-Controller (MVC), що пропонує потужну колекцію функцій для розробки веб та мобільних додатків. Це фактичний API для написання веб-

додатків поверх Node.js. Це набір бібліотек маршрутизації, що забезпечує тонкий шар основ-них функцій веб-додатків, що додають до чудових існуючих функцій Node.js. Він фокусується на високій продуктивності та підтримує надійну маршрутизацію та HTTP-помічники (перенаправлення, кешування тощо). Він оснащений системою перегляду, яка підтримує 14+ шаблонів, узгодження вмісту та виконувану програму для швидкої генерації програм. Крім того, Express постачається з безліччю простих у використанні утилітних методів, функцій та програмного забезпечення HTTP, що дозволяє розробникам легко та швидко писати надійні API. На Express створено кілька популярних фреймворків Node.js.

Отже, за завантаження із офіційних сайтів фреймворків, Express завантажили в десять раз більше ніж будь-який інший фреймворк. Це дає змогу стверджувати, що йому надають більше перевагу для розробки програмних засобів, ніж будь-якому іншому фрейморку. [5]

1.5 Огляд схожих додатків.

Український державний портал Дистанційного Зондування Землі запустив портал, який має широкий функціонал та працює у тестовому режимі.[6]

Функції, які підтримує додаток розміщені в таблиці 1.1:

Таблиця 1.1 – Функціонал додатку

LST Температура поверхні землі	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	Карта поверхневої температури Азово-Чорноморського басейну, створена за методикою Морського

Продовження таблиці 1.1

Азово-Чорноморського басейну	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

Функціонал, безумовно, вражаючий. Додаток дає великі змоги не тільки у сфері обробки земельних ділянок, а і у сфері науки. Робота над ним ще

триває, бо він знаходиться у тестовому режимі, але вже зараз зрозуміло, що можна туди додати і зробити його більш поширеним серед людей, які не займаються наукою:

- Можливість авторизації та реєстрації;
- Обрахунок вартості посіву на тій чи іншій земельній ділянці в залежності від області, розміру ділянки та курсу долара;
- Набір статей, які зможуть допомогти користувачу у питаннях, які у нього виникли;

Подібний функціонал дуже позитивно вплине на користувачів додатку та його можна буде викласти у вільний доступ. [6]

1.6 Постановка задачі

Розробити інформаційний веб-ресурс, який матиме візуальну та серверну частину з необхідним функціоналом Необхідно реалізувати наступні задачі:

1. Виконати огляд інструментів, які найбільш популярні на даний момент, провести їх короткий аналіз та вигоду для розробки та обрати оптимальні варіанти;
2. Розробити клієнтський інтерфейс, який зможе надати користувачу приємний користувацький досвід;
3. Спроектувати базу даних, для комфортного використання та оброблення даних на клієнтській частині;
4. Для комфортного користувацького досвіду, додати адаптивний інтерфейс під мобільні пристрої;
5. Розробити формули, які будуть використовуватись у додатку, для створення плану робіт.
6. Розробити та відобразити у клієнтській частині користувацьку інструкцію, яка дозволить швидко зрозуміти, яким чином потрібно використовувати додаток;
7. Провести тестування основного функціоналу додатку.

2 МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Кінцевий стек технологій, який використано для розробки

Клієнтська частина:

React – для розробки UI додатку, з ним використано ряд додаткових інструментів.

Redux – для керування актуальним станом додатку, щоб відділити бізнес логіку та UI-компоненти.

TypeScript – додаткова типізація коду клієнтської частини, щоб попередити ще на етапі розробки ряд потенційних помилок.

Додаткові бібліотеки для роботи з React:

React-router-dom – для побудови додатку по принципу SPA.

Axios – відправлення запитів на сервер з більш зручним синтаксисом.

React-redux – забезпечує зв'язок react-додатку з Redux store.

Node-sass – дозволяє використовувати синтаксис scss у react-додатку.

Swiper – надає ідеальний варіант слайдеру з максимально гнучкою системою кастомізації, в тому числі за допомогою scss. [4]

Серверна частина:

NodeJS – для створення серверної частини додатку на JavaScript.[10]

Express – легкий фреймворк для NodeJS, який дозволяє пришвидшити роботу та зробити код більш зрозумілим та простим.

MongoDB – документ-орієнтована база даних, яка більш підходить для розробки менш навантажених додатків.

Додаткові бібліотеки:

Mongoose – бібліотека, яка забезпечує взаємодію Node та MongoDB.

Config – дозволяє зберігати у форматі json деякі дані, які потрібні у різних модулях додатку та динамічно використовувати їх.

Express-validator – використовується для валідації даних на стороні серверу.

Jsonwebtoken – необхідний для збереження даних користувача у вигляді токену, для подальшої авторизації.

Bcryptjs – дозволяє захешувати пароль користувача при реєстрації, щоб не зберігати пароль у базі даних у тому вигляді, який задав користувач. [5]

2.2 Формули розрахунків для формування плану робіт

Розрахунок формули підрахунку витрат для культури «Пшениця»

1. Підготовка грунту 1 га
   - Оранка ділянки, ДТ=20л
   - Культивація, ДТ=5л
2. Посів ділянки
   - Посів, внесення необхідних добрив=ДТ3л +насіння 200кг + міндобрива npk150кг.
   - Внесення сульфат амонію 100кг на 1 га.
   - Внесення селітра аміачна 150кг на 1га.
3. Захист рослини
   - Листкове підживлення, шляхом внесення гербіциду 0.5л +фунгіциду 0.5л +мікро мінералів 0.2л +ДТ 1л
   - Листкове підживлення, шляхом внесення фунгіциду 0.5л +інсектицид 0.15л +ДТ 1л
   - Підживлення під час фази Колосіння, шляхом внесення фунгіциду 0.5л +карбаміду 12кг +ДТ 1л
4. Жнива
   - ДТ12л

Необхідні витрати для обробітку 1га пшениці $v=20t+5t+3t+(200n_1+150p)+100s+150c+(0.5g+0.15i+0.2m+t)+(0.5f+0.15i+t)+(0.5f+12k+t)+12t$

де вартість

t -затрати Дт 1л, t=70грн

n1 - насіння пшениці 1кг, n_1=10грн

p - міндобрива npk 1кг, p=20грн

s - сульфат амонію 1 кг, s=20грн

f - фунгіциду 1л, f=800грн

m - мікро мінералів 1л, m=350грн

i - інсектицид 1л, i=1000грн

k- карбаміду 1кг, k=34грн

Розрахунок формули підрахунку витрат для культури «Кукурудза»

1. Підготовка грунту 1 га

- Внесення NPK 150кг +ДТ 1л
- Оранка Дт20л
- Внесення КАС-32 200кг +ДТ 1л
- Боронування ДТ 4л

2. Посів ділянки

- Посів ДТ3л +1п.о. кукурудзи

3. Захист рослини

- Листкове підживлення, шляхом внесення гербіциду 1.2л +мікро мінералів 0.2л +ДТ 1л
- Листкове підживлення, шляхом внесення цинк 0.5л+ інсектицид 0.15л +ДТ 1л на 1га

4. Жнива

- ДТ12л

Необхідні витрати для обробітку 1га кукурудзи= =(150p+t+20t)+(200q+t)+4t+5t+(3t+n2)+(1.2g+0.2m+t)+(0.5x+0.15i+t)+12t

де вартість

t -затрати Дт 1л, t=

n1 - насіння кукурудзи 1п.о., n_1=5000грн

p - міндобрива npk 1кг, p=20грн

s - сульфат амонію 1 кг, s=20грн

c- селітра аміачна 1кг, c= 28грн

g - гербіциду 1л, g=1000грн

f - фунгіциду 1л, f=800грн

m - мікро мінералів 1л, m=350грн

i - інсектицид 1л, i=1000грн

k- карбаміду 1кг, k=34грн

x- цинк 1л, x=400грн

Розрахунок формули підрахунку витрат для культури «Соняшник»

1. Підготовка грунту 1 га
   - Внесення NPK 150кг +ДТ 1л
   - Оранка Дт20л
   - Внесення КАС-32 100кг +ДТ 1л
   - Боронування ДТ 4л
   - Культивація ДТ 5л
2. Посів ділянки
   - Посів ДТ3л +1п.о. соняшнику
3. Захист рослини
   - Листкове підживлення, шляхом внесення гербіциду 1.2л +мікро мінералів 0.2л +ДТ 1л

– Листкове підживлення, шляхом внесення фунгіциду 0.5л+ ДТ 1л

4. Жнива

– ДТ12л

Необхідні витрати для обробітку 1га соняшнику= =(150p+t+20t)+(200q+t)+4t+5t+(3t+n2)+(1.2g+0.2m+t)+(0.5x+0.15i+t)+12t

де вартість

t -затрати Дт 1л, t=

n1 - насіння соняшника 1п.о, n_1=3000

p - міндобрива npk 1кг, p=20грн

s - сульфат амонію 1 кг, s=20грн

c- селітра аміачна 1кг, c= 28грн

g - гербіциду 1л, g=1000грн

f - фунгіциду 1л, f=800грн

m - мікро мінералів 1л, m=350грн

i - інсектицид 1л, i=1000грн

k- карбаміду 1кг, k=34грн

2.3 Серверна частина веб-додатку

На серверну частину додатку покладено декілька задач.

По-перше, це зв'язок з базою даних. Всі операції проходять через БД. Статті із довідника, реєстрація та авторизація користувача, отримання планів робіт – усе це виконується через відправлення запитів на сервер, який бере з БД необхідні дані.

По-друге, валідація даних користувача. При введенні та відправленні даних, сервер перед записом даних у БД перевірить введені дані за критеріями відповідності. Для username та password це кількість символів, для email – присутність знаку @ у ньому.

По-третє, у реалізації серверу будуть описані endpoints. Саме сюди, користувач буде відправляти запити і контактувати з даними.

По-четверте, на серверній частині розроблені моделі для БД. З використанням цих моделей відбуваються пошук по базі даних, створення та видалення елементів.

Також, на сервері зберігаються middleware, які допомагають обійти політику захисту CORS та використовувати інші сценарії для покращення користувацького досвіду клієнтів додатку.

2.4 Огляд підходів до стилізації веб-додатку

Нині, є декілька популярних підходів для роботи зі стилями додатку.

1) CSS або Sass/SCSS
2) CSS-in-JS
3) CSS-бібліотеки.

Розглянемо кожен детальніше, та виділимо мінуси які і вплинули на рішення відмовитись від деяких підходів.

Суть підходу CSS-in-JS полягає у тому, щоб писати стилі у самих компонентах. Найвідомішим прикладом є styled-components.

Плюсом є те, що розробник має компонент і одразу у тому ж файлі бачить його стилі, і ці стилі не потрібно шукати у інших місцях додатку.

Мінусом є те, що така структура нагромаджує компонент та відходить від принципу ізоляції компонентів додатку. До того ж, це викликає додаткову вимогу у компілюванні цього коду у зрозумілий браузеру CSS.

Також, популярність набирають CSS-бібліотеки. Найвідомішим та найстарішим нині прикладом є Bootstrap. Але нині, частіше користуються Tailwind.css, Material UI або Ant Design.

Їх суть полягає у тому, що ці бібліотеки несуть із собою декілька CSS файлів у яких вже описані стилі. Тому шляхом підключення чітко визначених класів до розмітки, вони починають працювати.

Плюсом є швидкість розробки, але мінусом є те, що завдяки цьому підходу, кінцевий результат буде мати великий розмір за рахунок невикористаних стилів, бо не всі класи потрібні при розробці та перевантажені класи у розмітці, бо інколи один клас може досягати розмірів у декілька рядків коду, що для класу дуже не бажано.

Тому, було вирішено зупинитись на класичному підході написання стилів за допомогою SCSS.

Мінусом є те, що розробка займе трохи більше часу. Але плюсом є те, що розробник буде чітко знати, що і де він додає і за що відповідає конкретний клас. Також SCSS додає чудові можливості які дуже прискорюють роботу, наприклад можливість вкладати класи один в одного, використовувати змінні та структури, які називають міксінами – можливість підготувати деяку заготовку CSSта за допомогою include долучити до різних класів не дублюючи код.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

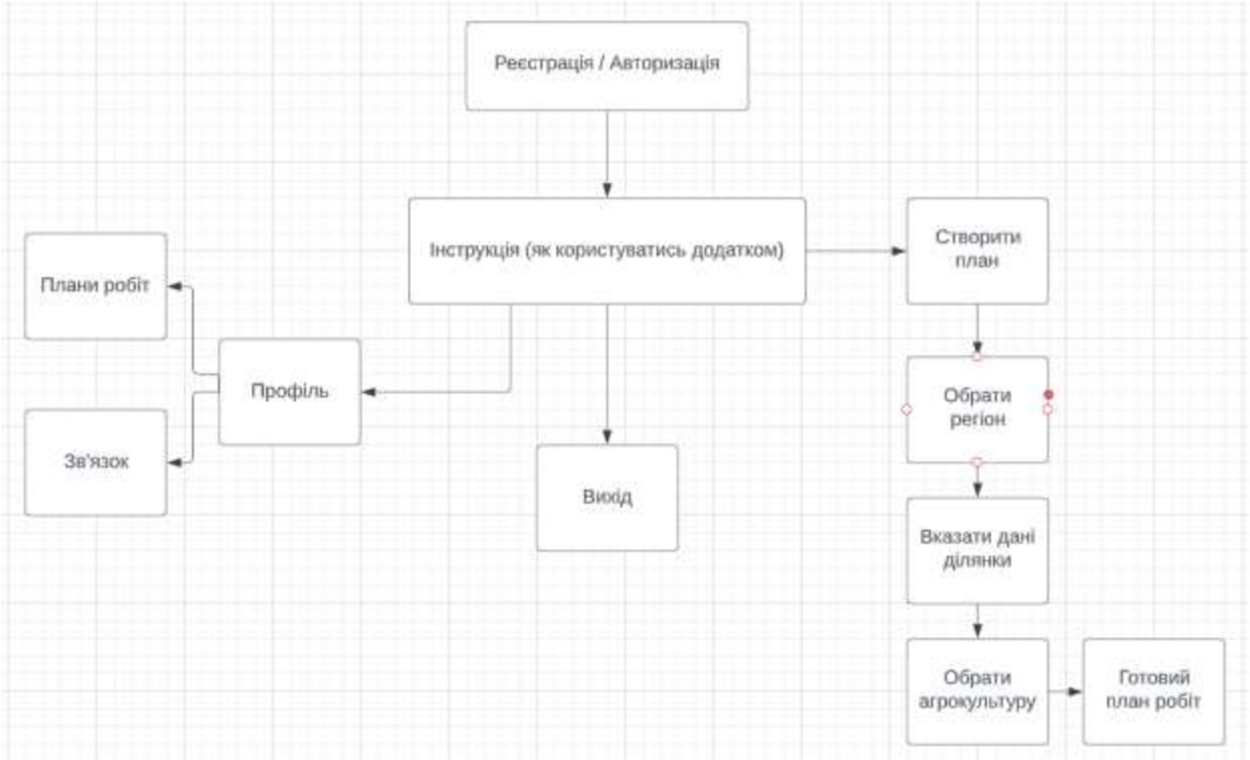

Рисунок 3.1 – Навігація клієнтської частини

На рисунку 3.1 представлено навігацію по додатку, яку буде використовувати клієнт.

Після реєстрації та авторизації у додатку, користувач перейде на сторінку, де буде показано як використовувати додаток ефективно та для чого він потрібен.

У панелі навігації можна буде перейти до Профілю або до Створення плану робіт.

Якщо почати створювати план робіт, необхідно вказати регіон земельної ділянки, дані про неї та агрокультуру, яка буде використовуватись. На основі цих даних буде сформований приблизний план робіт.

При переході на сторінку профілю, буде можливість переглянути плани робіт користувача та зв’язатись із автором.

У базі даних буде міститись декілька документів. До основних відносяться User, Articles, Areas.

Таблиця 3.1 – Модель User для роботи з БД

User	**Email:** Type: string Required: true Unique: true	**Username:** Type: string Required: true	**Password:** Type: string Required: true	Areas: []

Модель містить поля email, username та password. Поле з електронною поштою, є унікальним, бо можливість зареєструвати однакову пошту не дають самі поштові сервіси, тобто таким чином ми виключимо можливість існування двох акаунтів з однаковою поштою та уникнемо ряду помилок.

Таблиця 3.2 – Модель Areas для роботи з БД

Areas	**Region:** Type: string Required: true	**Price:** Type: string Required: true	**Agriculture:** Type: string Required: true	-

Таблиця 3.2 містить поля region для вказання, де саме знаходиться земельна ділянка, price – для збереження ціни на обробку, agriculture – для збереження конкретної культури, якою планується засів поля.

Кожен документ має свої поля та атрибути цих полів, тип даних, обов'язковість, унікальність. Також MongoDB додає до кожного елементу _id – унікальний ідентифікатор, яким можна користуватись, для того, щоб зв'язати документи між собою. У нашому випадку, це знадобиться при роботи документів User та Areas.

Таблиця 3.3 – Модель Articles для роботи з БД

Articles	**Title:** Type: string Required: true Unique: true	**Body:** Type: string Required: true	**-**	-

Таблиця 3.3 містить поля title та body. Перший слугує для назви статті, друге – для змісту. За допомогою поля title буде проводиться пошук у БД та відображення відповідної статті на клієнтській частині.

3.2 Програмна реалізація

Спочатку необхідно реалізувати реєстрацію та авторизацію користувача у додатку.

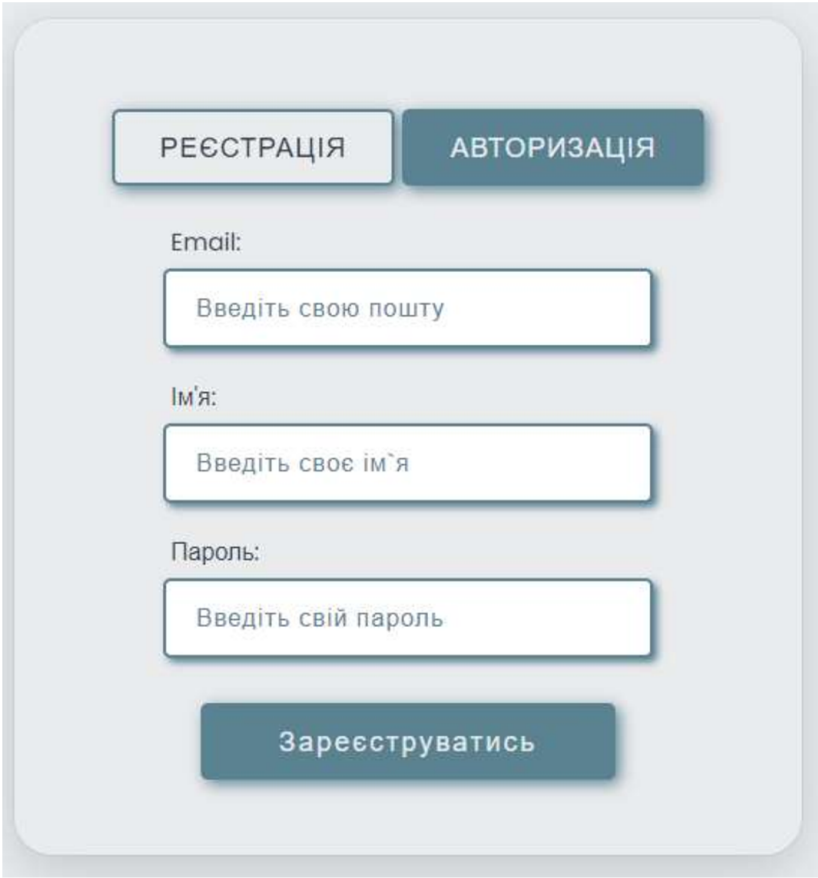

Рисунок 3.2—Форма реєстрації користувача у додатку

На рисунку 3.2 представлена форма для реєстрації користувача у додатку. Для реєстрації необхідно ввести пошту, ім'я та придумати пароль.

Валідація даних проходить на серверній частині додатку. За допомогою express-validator проходить перевірка кожного значення, яке надходить з клієнта. Необхідно ввести коректний email, ім'я, довжина якого повинна бути від 2 до 10 символів та пароль, від 3 до 12 символів.

```
[
            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 })
      ],
```

У прикладі коду express-validator повертає масив, у якому ми перевіряємо за допомогою функції check поля форми.

Рисунок 3.3—Форма авторизації користувача у додатку

На рисунку 3.3 представлена форма для авторизації користувача. Адреса електронної пошти та пароль – дані які користувачу необхідно ввести, щоб увійти у додаток. Ці дані були збережені у базі даних під час реєстрації користувача. Коли надходить запит на відповідний route на сервері, йде пошук клієнта по _id, який йому автоматично присвоїла MongoDB. На клієнтську частину сервер повертає пошту, ім'я, масив з планами користувача та його _id.

```
      const isPasswordValid = bcrypt.compareSync(password, user.password);
                  if (!isPasswordValid) {
                        return    res.status(400).json({    message:    'Invalid
password' });
                  }
                  const     token     =     jwt.sign({     id:     user.id     },
config.get('secretKey'), {
                        expiresIn: '1h'
                  });
                  return res.json({
                        token,
                        user: {
                              id: user.id,
                              email: user.email,
                              username: user.username,
                              areas: user.areas
                        }
                  });
```

У прикладі коду спочатку, необхідно перевірити пароль, що ввів користувач при авторизації. Для початку необхідно розшифрувати захешований пароль та порівняти його з паролем, що ввів користувач. Якщо паролі однакові, то повертаємо token користувача та його дані.

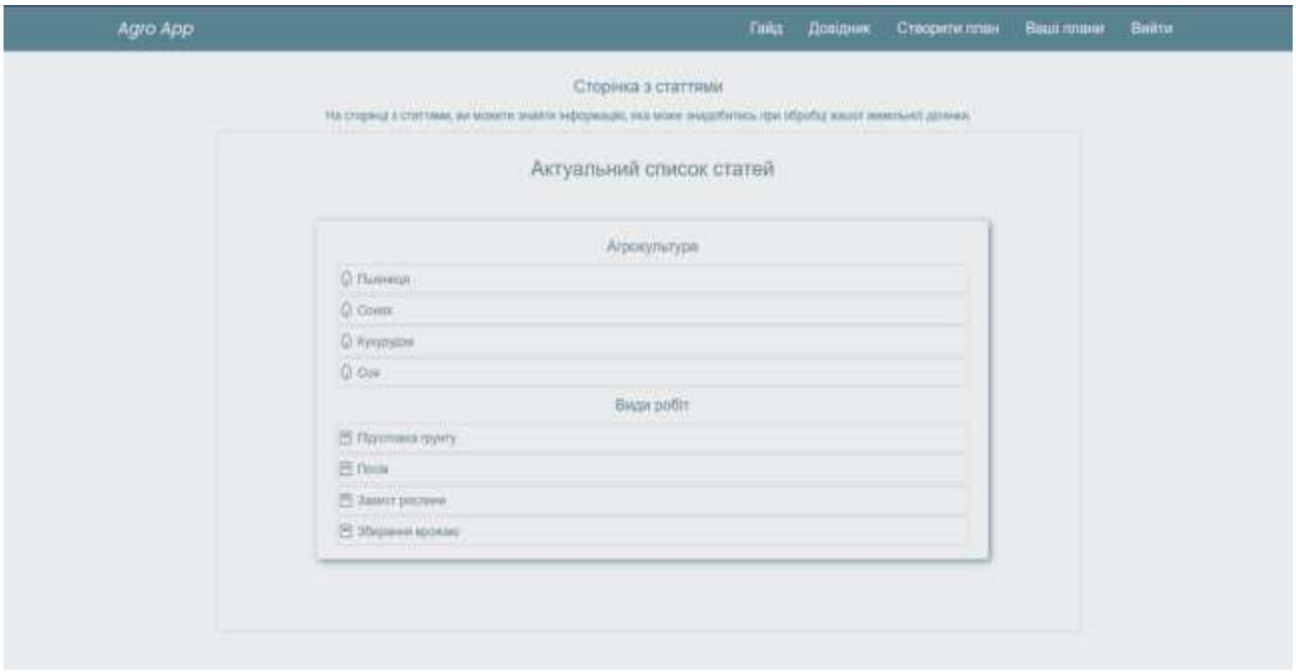

Рисунок 3.4—Початкова сторінка додатку

На рисунку 3.4 представлена початкова сторінка додатку. На ній представлений slider, за допомогою якого користувач зможе ознайомитись з основними функціями додатку. Для реалізації було використано Swiper. Який надає можливість швидко і якісно створити slider. Перемикання слайдів відбувається за допомогою жестів. Достатньо затримати ЛКМ на потягнути вліво або вправо для перемикання.

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

У прикладі коду modules відповідає за підключення модулів, за допомогою яких працює slider, sliderPreView – кількість слайдів на сторінці, onSwiper – функція зворотного виклику, яка спрацює при перемиканні слайду жестом, onSlideChange – функція зворотного виклику, яка спрацює при будь-якому перемиканні слайду.

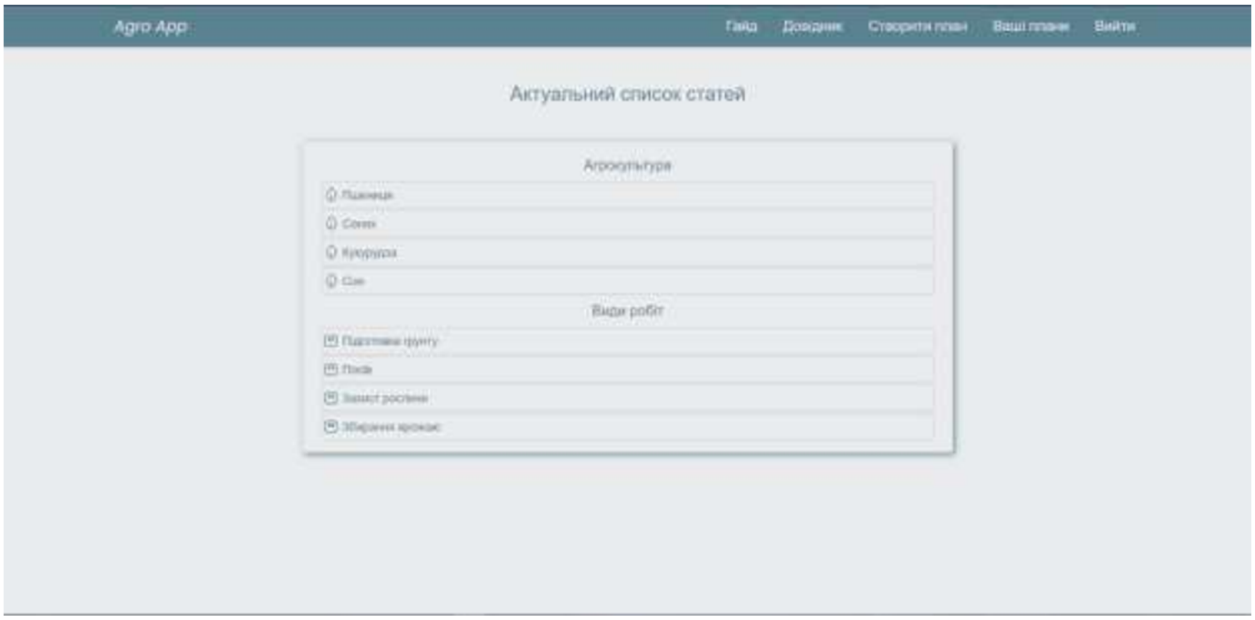

Рисунок 3.5—Сторінка з статтями

На рисунку 3.5 представлена сторінка з статтями. Користувач може обрати пункт у списку, який його цікавить та перейти до перегляду самого тексту. Пункти списку, як і самі статті зберігаються у базі даних.

```
{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>
                        ))}
```

У прикладі коду у articles ми отримали масив заголовків статей, за допомогою методу масиву map проходимо по кожному елементу та виводимо його на екран. При кліку на конкретний заголовок передаємо його у store.

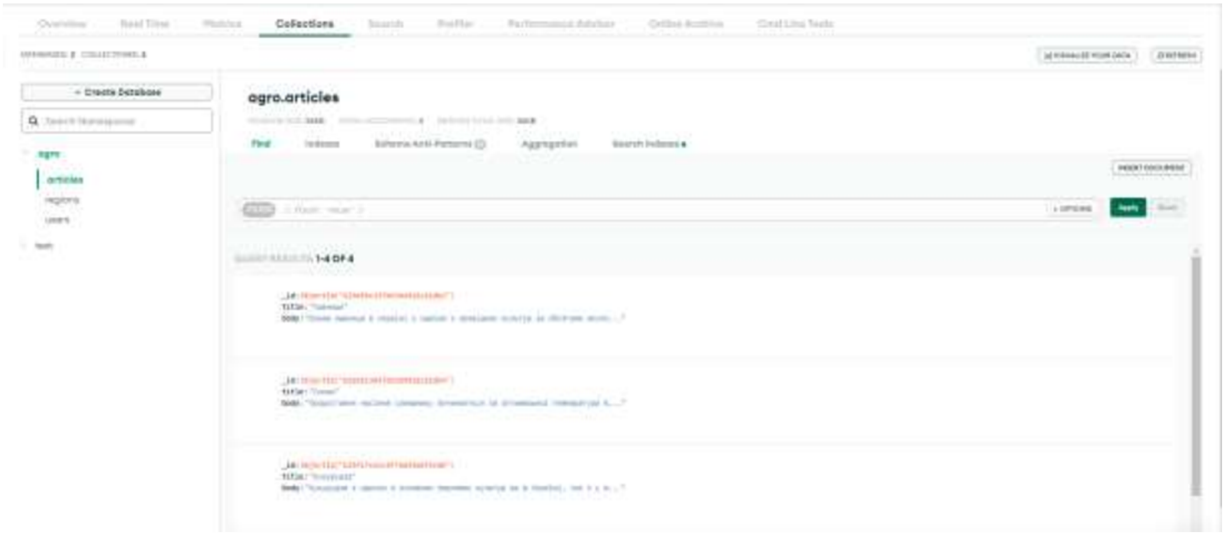

Рисунок 3.6—Документ бази даних для зберігання статей

На рисунку 3.6 представлено документ бази даних для зберігання статей. Дані зберігаються у форматі «масив об’єктів». У кожному об’єкті збігається _id окремої статті, який автоматично надала MongoDB, title – заголовок статті, який також використовувався при формуванні списку статей, body – текст статті, який відображається при переході у окремий пункт списку.

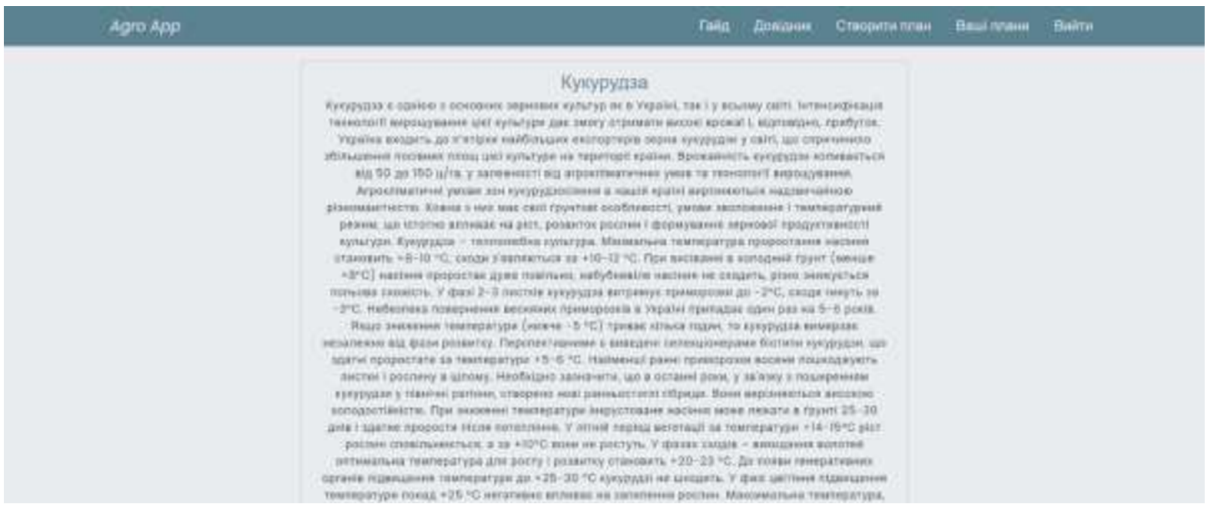

Рисунок 3.7—Документ бази даних для зберігання статей

На рисунку 3.7 представлена реалізація відображення окремої статті. Дані ми отримуємо з бази даних та зберігаємо у store, який надає Redux Toolkit. Після цього у компоненті для відображення статті, за допомогою хуків, які надає state-manager – useSelector() та UseDispatch() ми отримуємо дані у компоненті та відображаємо їх.

```
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>
      );
}
```

У прикладі коду зі store забираємо усі статті та обраний користувачем заголовок. Далі, за допомогою методу find знаходимо статтю, яка має такий заголовок та відображаємо її користувачу.

Рисунок 3.8 — Створення плану. Сторінка вибору ділянки

На рисунку 3.8 представлена сторінка для вибору регіону, в якому знаходиться ділянка користувача. На сторінці реалізована поле для пошуку області та, відповідно, список областей.

Рисунок 3.9 — Створення плану. Пошук області при введенні значення

На рисунку 3.9 представлена реалізацію пошук конкретного регіону. З бази даних приходить масив областей. Тому для того, щоб реалізувати пошук областей необхідно використати функцію метод масиву filter. Якщо значення,

яке введено в поле таке ж, яке знаходить у масиві, то це і буде єдиний елемент, який буде відображено. Для того, щоб при видаленні значення з поля вводу, знову відображались всі регіони було використано умовний render, за допомогою тернарного оператора.

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

У прикладі коду ми отримуємо дані зі локального state, у якому зберігається введене користувачем значення. Далі за допомогою методу масива filter фільтруємо масив з назвами областей та виводимо ту, яка підходить під умову.

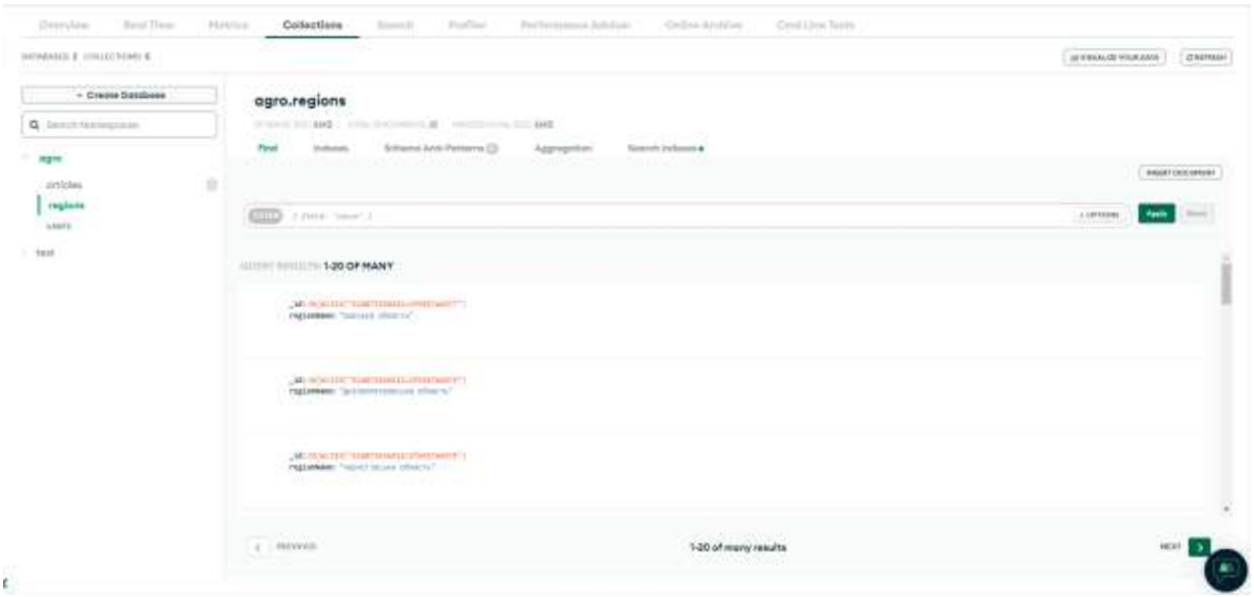

Рисунок 3.10 — Документ бази даних для збереження списку регіонів

На рисунку 3.10 представлено реалізація бази даних для збереження регіонів. Кожен елемент має поле _id, яке автоматично надає база даних та rigionName, який містить назву області.

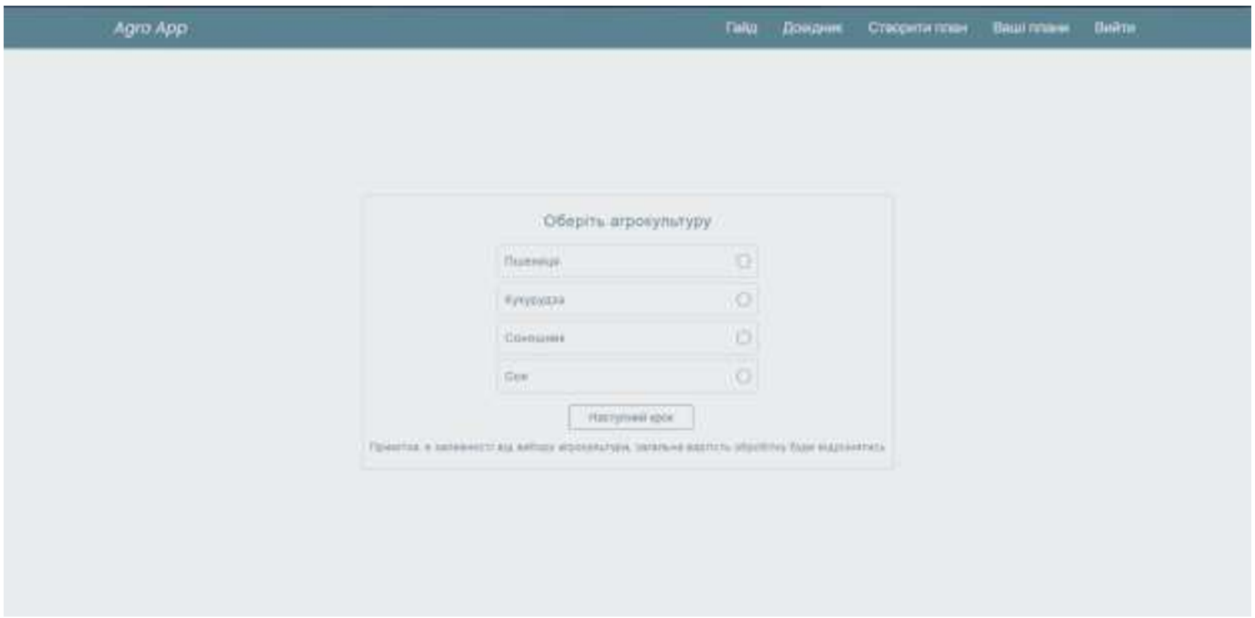

Рисунок 3.11 — Сторінка для вибору агрокультури

На рисунку 3.11 представлена реалізація сторінки для вибору агрокультури. В залежності від вибору культури, буде змінюватись загальна ціна на обробіток.

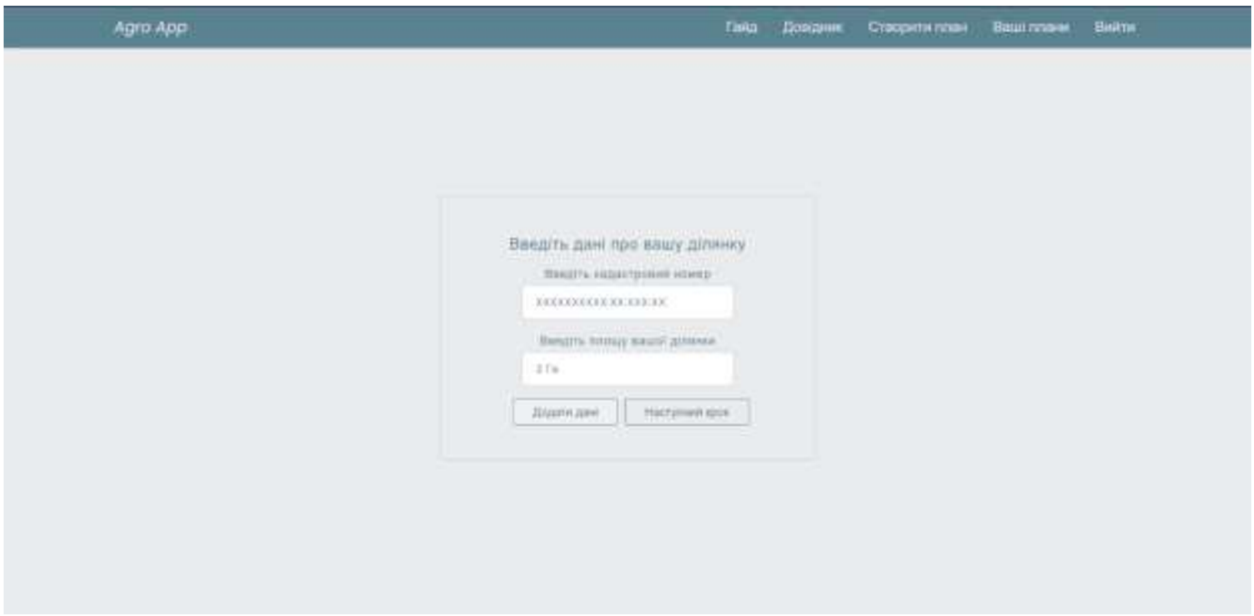

Рисунок 3.12 — Сторінка для збору даних про ділянку

На рисунку 3.12 представлена реалізація сторінки для збору кадастрового номеру у форматі XXXXXXXXXX : XX : XXX : XXXX та площі ділянки. Площа ділянки також безпосередньо впливає на формування ціни.

Після введення даних, необхідно натиснути кнопку “Додати дані”. Таким чином, додаються дані до загального store, який надає redux-toolkit.

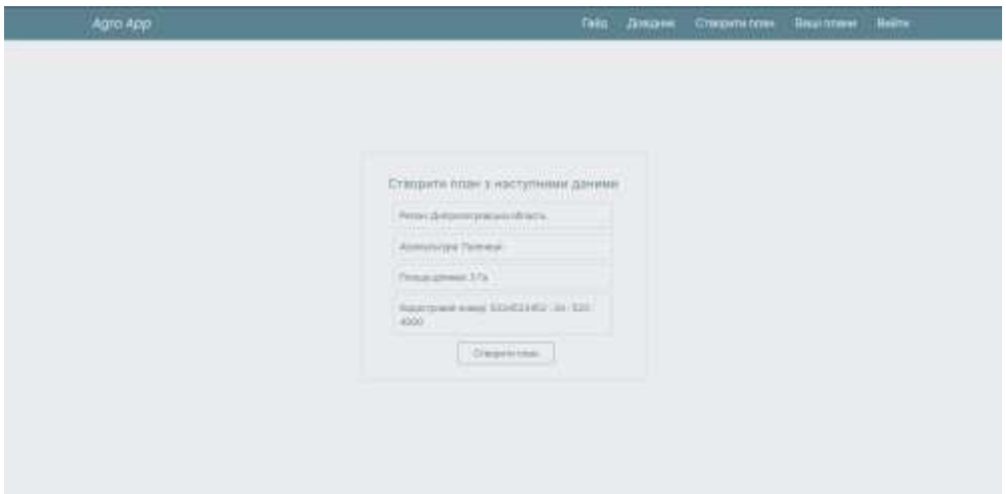

Рисунок 3.13 — Сторінка підтвердження плану

На рисунку 3.13 представлена реалізація сторінки, яка відповідає за підтвердження даних. Користувач перевіряє дані, які були введені та, якщо все влаштовує, натискає “Створити план”.

Після цього відбувається обчислення за допомогою формул, які зберігаються у redux-store.

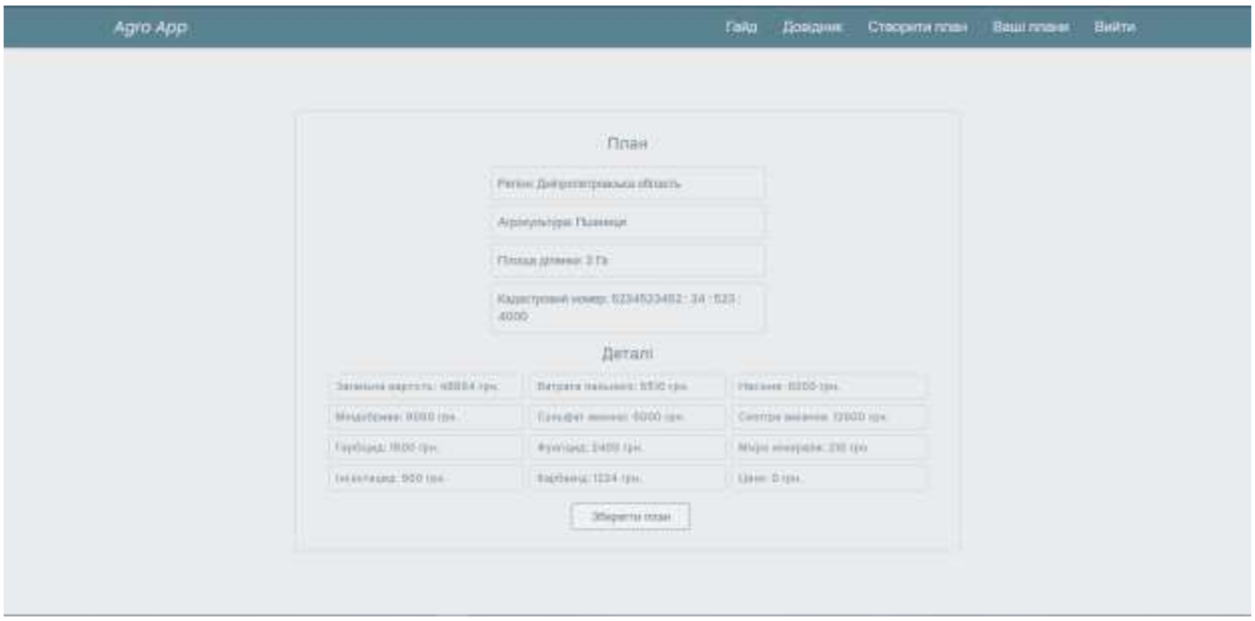

Рисунок 3.14 — Приклад кінцевого плану

На рисунку 3.14 представлено реалізацію виведення даних, які були обчислені. У першій частині таблиці показані дані про ділянку, а у нижній конкретні дані, які і потрібні користувачу. Наведено загальну вартість обробітку землі та приблизні витрати на кожний компонент. Таке відображення дозволить клієнту більш зручно оперувати грошима, які будуть виділені для обробітку.

Після цього необхідно зберегти дані. Після того, як користувач натисне кнопку “Зберегти план” буде сформовано об’єкт даних, у який входить дані про ділянку, обчислені одиниці та _id користувача. Цей об’єкт за допомогою post запита буде відправлено на сервер на відповідний route, у якому описана

логіка пошуку клієнта у базі даних за його _id, який було передано у об'єкті та додавання об'єкту до масиву areas.

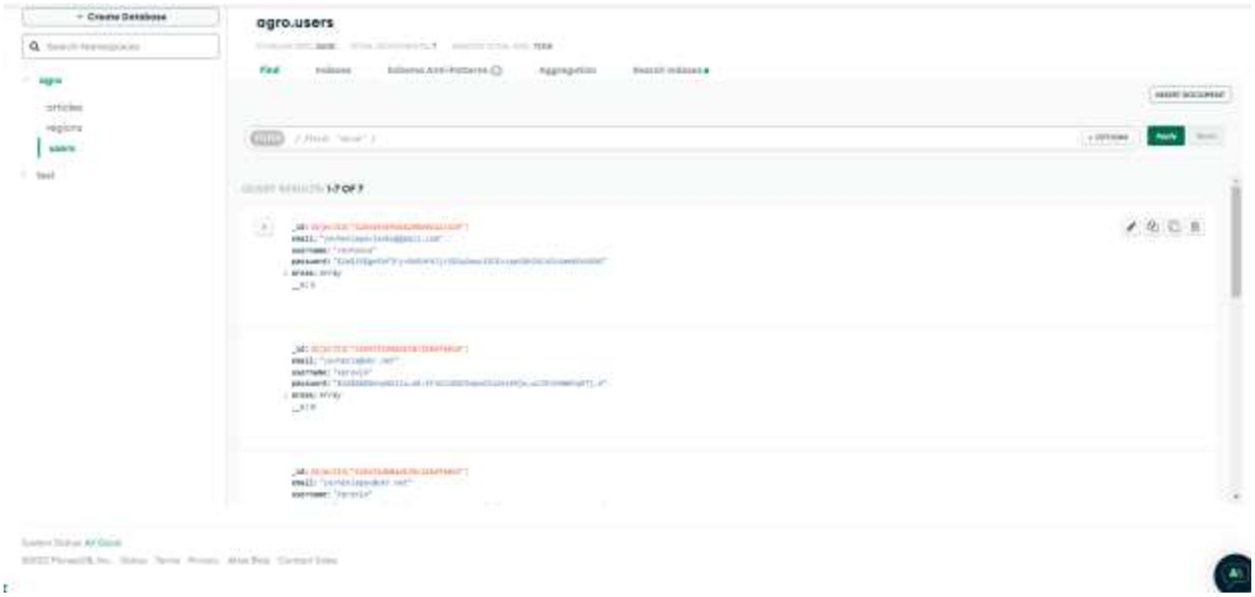

Рисунок 3.15 — Документ бази даних для профіля клієнта

На рисунку 3.15 представлена реалізація документа бази даних для користувача. Документ містить _id – унікальний ідентифікатор, який надає MongoDB, email – електронна пошта, яку користувач вводив при реєстрації, username – ім'я користувача, яка також було введено при реєстрації, password – захешований пароль, для більшої безпеки акаунту користувача, areas – масив з планами робіт, які створював користувач.

```
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');
        }
```

У прикладі коду представлено функцію для оновлення даних користувача при додаванні нового плану або при повному перезавантаженню сторінки. Викликати її необхідно за допомогою useEffect з пустим масивом залежностей.

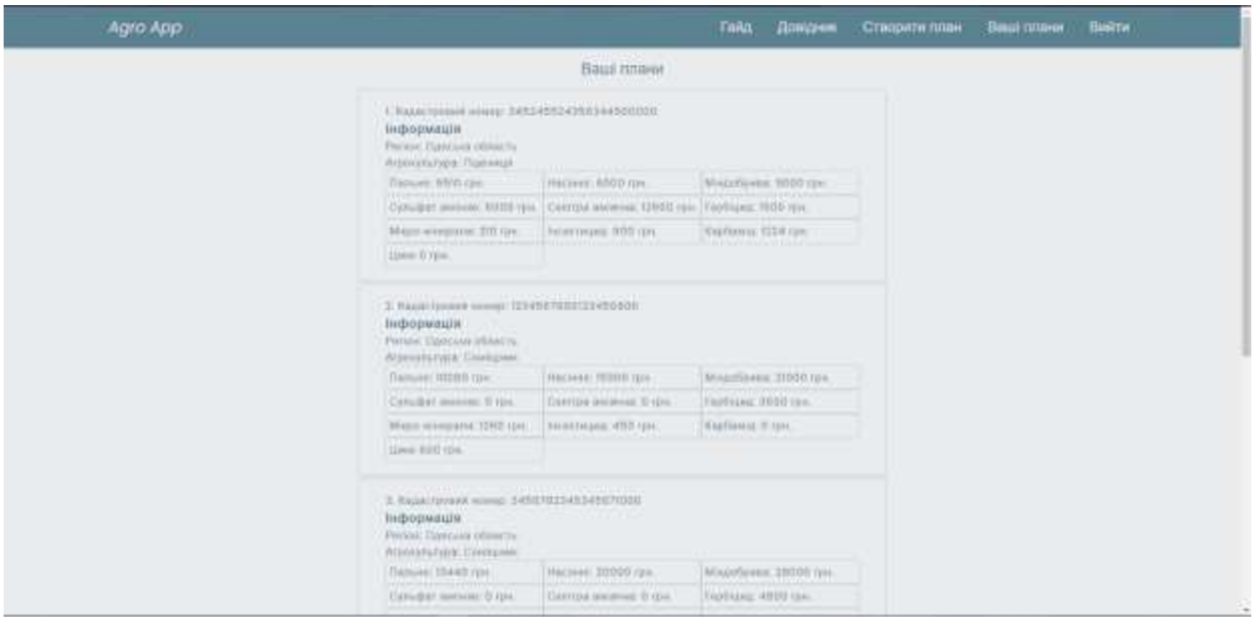

Рисунок 3.16 — Сторінка з усіма створеними планами

На рисунку 3.16 представлена реалізація виведення даних про всі плани, які створив користувач. Дані беруться з бази даних та виводяться на цій сторінці. При створенні користувачем нового плану, клієнтська частина не

отримує нових даних одразу, тому при першому переході на сторінку з планами на сервер відправляється get запит за допомогою якого користувач може отримати всі оновлені дані.

Також додаток адаптований під ноутбуки, планшети та мобільні пристрої.

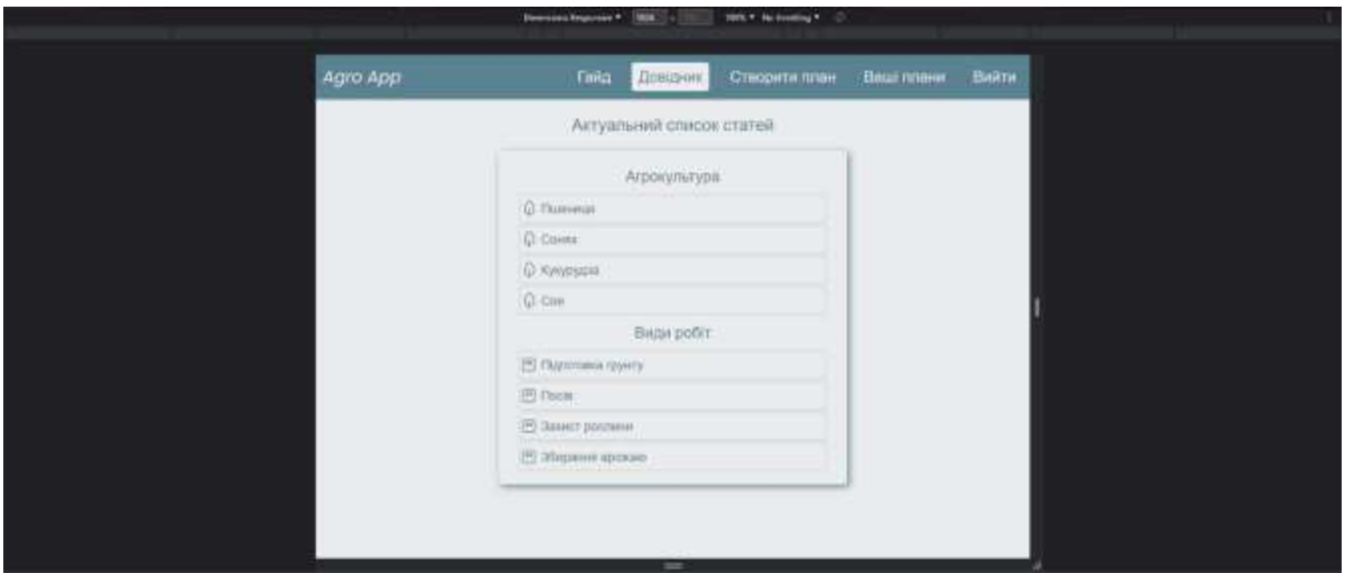

Рисунок 3.17 — Відображення додатку на діагоналі екрану ноутбука

На рисунку 3.17 представлена реалізація адаптивного інтерфейсу додатку під діагональ ноутбука. Реалізувати це вдалося завдяки зменшенню значення max-width при потрібній ширині екрану користувача.

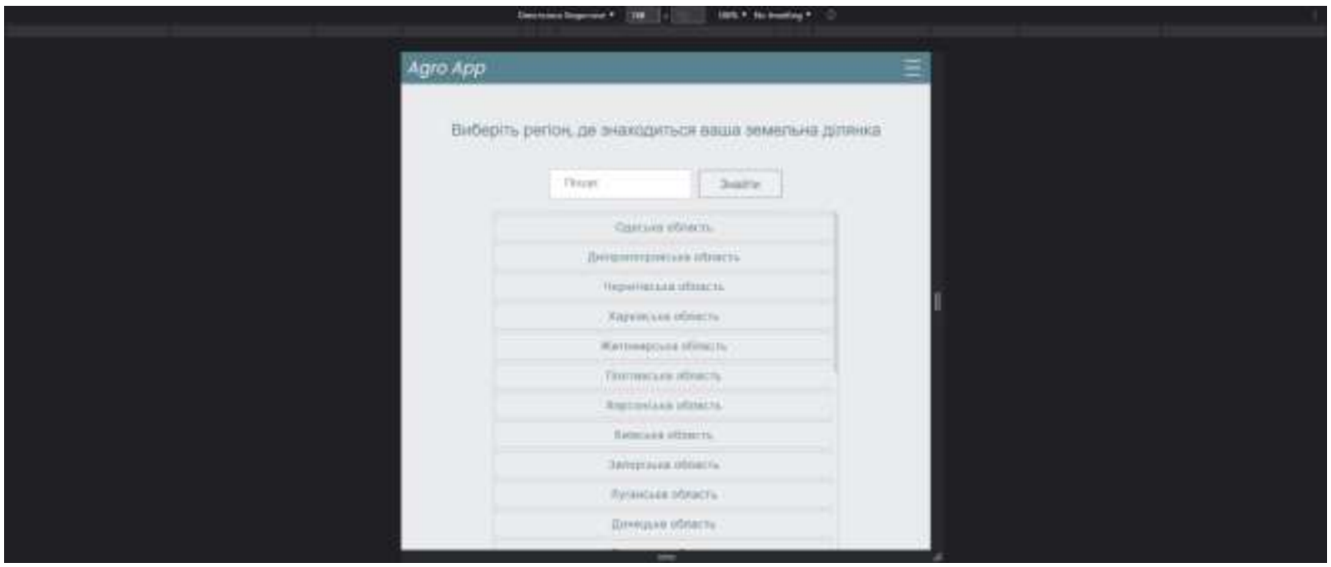

Рисунок 3.18 — Відображення додатку на діагоналі екрану планшета

На рисунку 3.18 представлена реалізація додатку на планшетній ширині екрану. У деяких місцях необхідно було змінити flex/grid сітку, для коректного відображення.

Рисунок 3.19 — Відображення адаптивного меню

При планшетному розмірі екрану з’явилась необхідність адаптувати головне меню додатку. На рисунку 3.19 показано реалізація адаптивності. Основне меню ховаємо за допомогою display: none і показуємо уже в новому

вигляді змінюючи flex-сітку. Також додано так зване «меню-бургер». При кліку на нього і з'являється адаптоване меню.

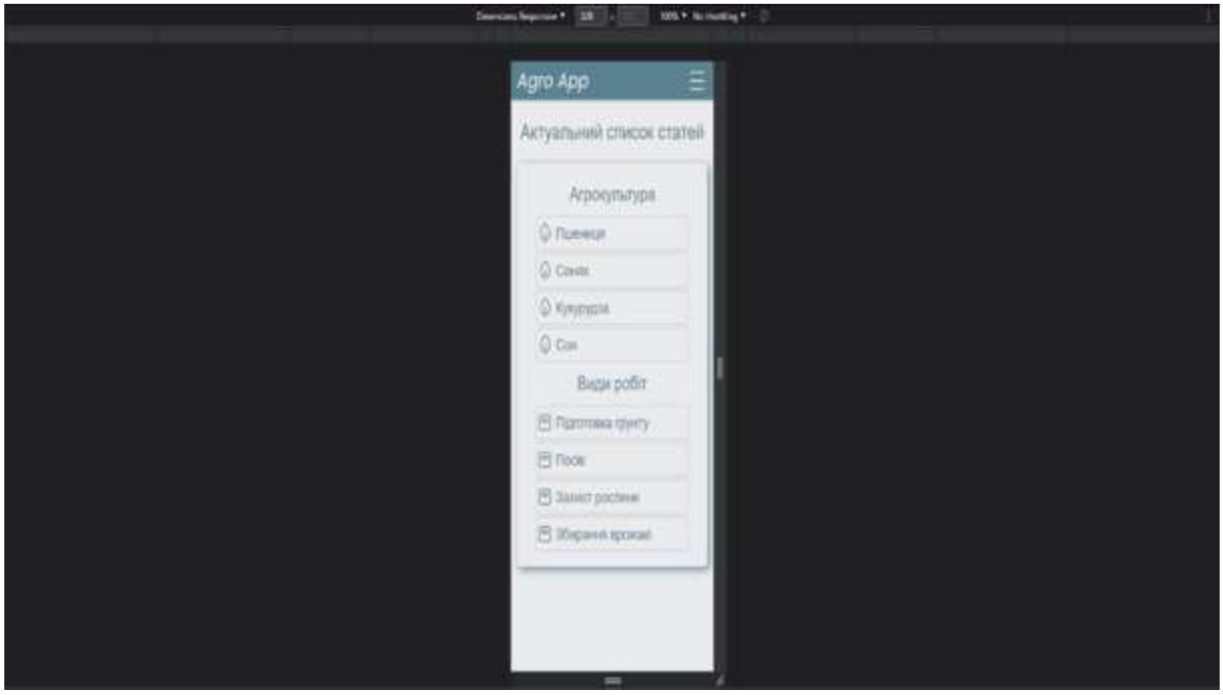

Рисунок 3.20 — Відображення адаптивного інтерфейсу на найменшому мобільному екрані

На рисунку 3.20 показана реалізація адаптивного інтерфейсу на мінімальній ширині екрану, яку є сенс підтримувати. Така реалізація вимагає використання усіх можливостей: зміна max-width, зменшення шрифтів, зміна felx/grid сітки та додавання адаптивного головного меню.

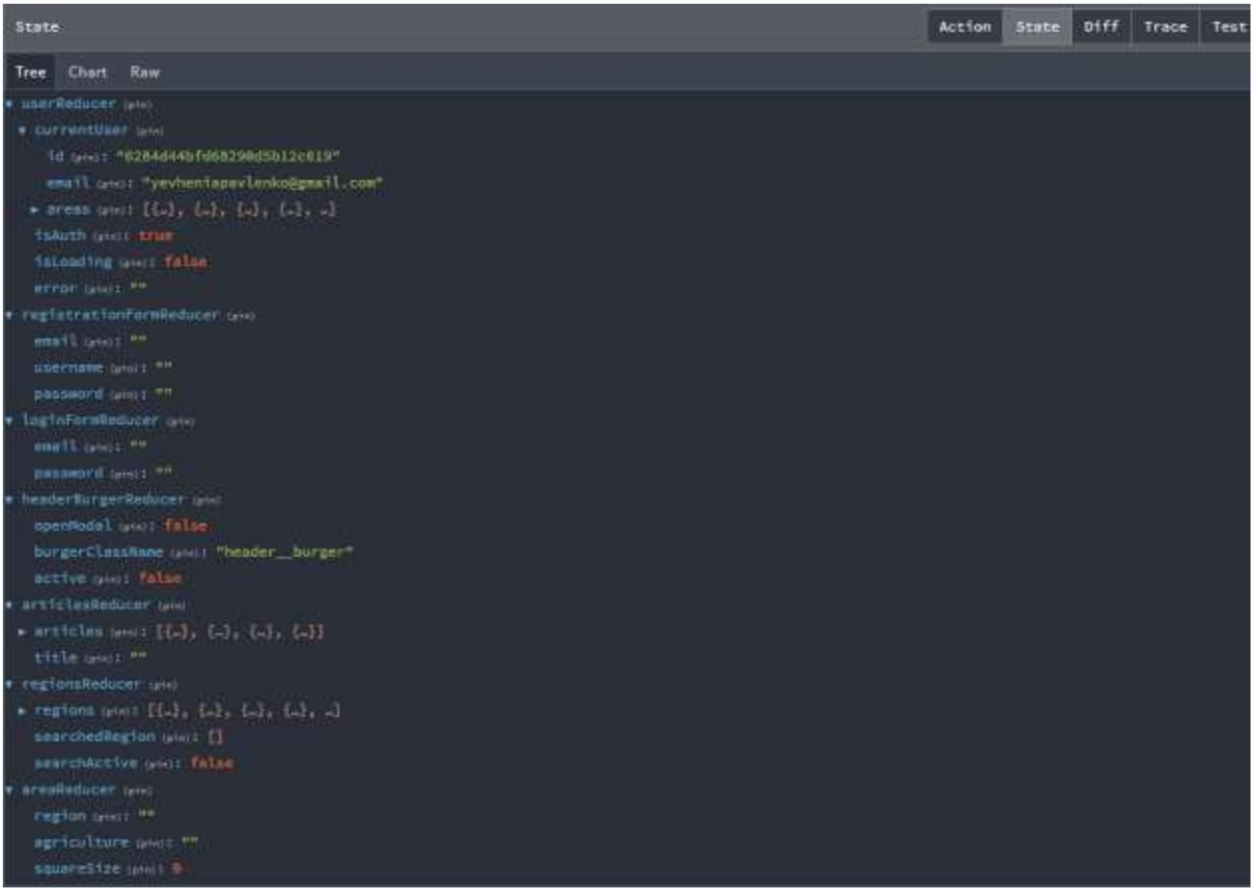

Рисунок 3.21 — Обробка даних на клієнтській частині за допомогою Redux Toolkit

На рисунку 3.21 представлено набір reducers, які використані у додатку. З їх допомогою, вдалося відокремити логіку по обробці даних від компонентів, які відповідають за відображення контенту для користувача.

При запитих на сервер, завжди використовувався dispatch, який у собі викликав необхідний action, який у свою чергу викликав необхідну логіку, яка була написана у reducers.

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

У прикладі коду представлено реалізація reducer для даних користувача. У полі authUser ми записуємо дані користувача при авторизації для майбутньої взаємодії з бекендом.

У полі logoutUser відбувається очищення даних користувача, коли він покине додаток.

OnLoading та offLoading керують відображенням поля “Loading…”, коли відбуваться асинхронні запити до сервера.

ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було виконано:

- Огляд сучасних інструментів для розроблення веб-додатків;
- Обрано оптимальні інструменти для розробки;
- Сформовані головні вимоги до додатку;
- Представлено план навігації по додатку;
- Сформована коротка модель основних документів бази даних;
- Представлені формули, за якими будуть проходити обрахування;
- Описано, який функціонал бере на себе серверна частина додатку;

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.

СПИСОК ЛІТЕРАТУРИ

1. Посібник: знайомство з React [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/tutorial/tutorial.html.
2. Посібник: знайомство з Redux [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/tutorials/essentials/part-1-overview-concepts
3. Дейли Б., Дейли Б, Дейли К. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN = Web Development with Node and Express.- Санкт-Петербург: «Диалектика-Вильямс», 2020.-656 с.
4. Стаття: знайомство з Nest.js [Електронний ресурс] - Режим доступу до ресурсу: https://habr.com/ru/post/439434/
5. Стаття: Порівняння фреймворків Express та Nest[Електронний ресурс] - Режим доступу до ресурсу: https://conf.ztu.edu.ua/wp-content/uploads/2021/01/3-2.pdf
6. Стаття: Безкоштовні агрододатки [Електронний ресурс] - Режим доступу до ресурсу: https://propozitsiya.com/ua/bezkoshtovni-agrododatky
7. Посібник: знайомство з TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/handbook/intro.html
8. Посібник: знайомство з npm [Електронний ресурс] - Режим доступу до ресурсу: https://docs.npmjs.com/
9. Посібник: знаймоство з MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/
10. Посібник: знайомство з NodeJS [Електронний ресурс] - Режим доступу до ресурсу: https://nodejs.org/uk/docs/
11. Посібник: знайомство з JS [Електронний ресурс] - Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web

12. Single page application. A complete guide [Електронний ресурс]– Режим доступу до ресурсу: https://appcheck-ng.com/single-page-applications#
13. SASS Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/documentation/
14. Redux-Toolkit Documentation [Електронний ресурс]– Режим доступу до ресурсу: https://redux-toolkit.js.org/usage/usage-guide
15. Express Documentation [Електронний ресурс]– Режим доступу до ресурсу: https://expressjs.com/en/guide/routing.html

ДОДАТОК

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
,700;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'
                        >
                              Пошта:
                        </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'>
                                    Міндобрива:
{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 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 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>
      );
}
```

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 грунту
                        </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>
                    <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 { useAppDispatch } 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>
                </ul>
            </nav>
        );
}
```

Файл UserPlans.tsx

```
import { useEffect } from 'react';
import { useDispatch } from 'react-redux';
import { auth } from '../../asyncActions/auth';
import { useAppSelector } from '../../hooks/redux';

import '../../scss/UserPlans.scss';

export function UserPlans() {
    const dispatch = useDispatch();

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

    const userPlans = useAppSelector(
        state => state.userReducer.currentUser.areas
    );
    return (
        <div className='user-plans'>
            <h2 className='user-plans__title'>Ваші плани</h2>
            <div className='user-plans__inner'>
                <div>
                    {userPlans.map((plan, index) => (
                        <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'>
                                                {`Пальне:
${plan.planData.DT} грн.`}
                                            </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>
      );
}
```

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.active
                        ? (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, config.get('secretKey'));
            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 }, config.get('secretKey'), {
                  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();
const PORT = config.get('serverPort');

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(config.get('dbUrl'));
            app.listen(PORT, () => {
                  console.log('Server has been started on port', PORT);
            });
      } catch (event) {}
};

runServer();
```