Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка web-застосунку порівняння прогнозів погоди»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1911»

	(підпис студента)	/Кирило ДОРОГОКУПЛЯ/ (Ім'я ПРІЗВИЩЕ)
Керівник:	(підпис)	/доц. Олександр ІВАНОВ/ (посада, Ім'я ПРІЗВИЩЕ)
Нормоконтролер:	(підпис)	/доц. Світлана ВОЛКОВА/ (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____ (підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Development of web application for comparing weather forecasts»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1911: /Kyrylo Dorohokuplia/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Дорогокуплі Кирилу Олекснадровичу
1. Тема роботи: «Розробка web-застосунку порівняння прогнозів погоди»
Керівник роботи: Іванов Олександр Петрович, доцент
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Вступ
Збір та аналіз вимог до веб додатку
Проектування веб додатку
Проектування прототипу веб додатку
Розробка прототипу веб додатку
Розробка веб додатку
Тестування веб додатку
Загальні висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
Презентація
Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 -18.02.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23
	Подання кваліфікаційної роботи до кафедри	07.06.23
	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	26.06.23

Студент _____ Кирило ДОРОГОКУПЛЯ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олександр ІВАНОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 7 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір та аналіз вимог до веб додатку – у цьому розділі ми розглядаємо та описуємо аналоги, аналізуємо їх оцінки доповнюючи це плюсами та мінусами та складаємо постановку подальшої роботи. Складається з 19 сторінок;

– проектування веб додатку – у цьому розділі описуються задачі проекту, його функціональні вимоги, вхідні та вихідні дані, описується вибір мови програмування та підходу розробки, проектується та розробляється прототип веб додатку який буде основою нашого веб додатку. Складається з 22 сторінок;

– розробка веб додатку – у цьому розділі, на основі прототипу, розробляється динаміка системи, інтерфейс веб додатку, система навігації та дизайн. Складається з 9 сторінок;

Тестування та налагодження веб додатку – у цьому розділі розроблений мобільний додаток тестується методами білої та чорної скриньки. Складається з 10 сторінок.

– загальні висновки – підсумки всієї роботи. Складається з 1 сторінки;

– список використаних джерел – включає в себе бібліографічний список використаної літератури. Складає 2 сторінки;

– додатки – містить технічне завдання код прототипу веб додатку та код робочого проекту. Кількість таблиць: 9 штук. Кількість рисунків: 54 штуки.

ЗМІСТ

ВСТУП

Ласкаво просимо до дипломної роботи на тему **"Development of web application for comparing weather forecasts"**. Сучасні технології та інтернет-ресурси дозволяють нам отримувати актуальну інформацію про погоду у будь-якій точці світу. Однак, як правило, дані про погоду збираються на різних метеостанціях і можуть відрізнятися за точністю та повнотою. У зв'язку з цим важливо мати можливість порівнювати дані з різних джерел для отримання більш точної та надійної інформації про погоду.

У даній роботі представлено розробку погодного застосунку, який дозволяє порівнювати дані про погоду з різних метеостанцій та візуалізувати їх у зручному та зрозумілому форматі. Розглянуто процес збирання, зберігання та обробки даних про погоду, а також реалізацію функціоналу програми, включаючи графічний інтерфейс, можливості налаштування та фільтрації даних, а також функції порівняння та аналізу.

Метою даної роботи є створення корисного та зручного інструменту для отримання та аналізу даних про погоду, який використовуватиметься як людьми, які цікавляться погодою у повсякденному житті, так і професійними метеорологами та дослідниками.

РОЗДІЛ 1. ЗБІР ТА АНАЛІЗ ВИМОГ ДО МОБІЛЬНОГО ДОДАТКУ

1.1. Опис аналогів

У сучасному світі інформаційних технологій, інтернет-ресурси надають багато можливостей для отримання актуальної інформації про погоду в будь-якій точці світу. Існує багато веб-сайтів та додатків, що забезпечують інформацію про погоду, проте не всі з них забезпечують точність та достовірність даних.

У цьому розділі розглянуто деякі аналоги за дпомогою пошукової системи Google, які вже існують на ринку, та порівняємо їх з нашим веб-застосунком для порівняння погодних ресурсів. Розглянтуо їхні переваги та недоліки, а також опишемо їхній функціонал, інтерфейс та можливості, що пропонують користувачам.

1. AccuWeather може відображати щотижневі, щоденні та щогодинні метеозведення. З них ви дізнаєтеся про температуру повітря в градусах Цельсія або Фаренгейта, вологість, опади, швидкість і напрям вітру, а також час сходу і заходу сонця. Також повідомлять про інтенсивність ультрафіолетового випромінювання, якщо ви зібралися засмагати, або про буру, що насувається.

Загальний опис функціоналу:

- можливість вибору прогнозу погоди:
  - поточний на день;
  - погодинний;
  - щоденний;
  - щохвилинний;
  - щомісячний;
- пошуку локації;
- можливість перегляду глобального режиму:
  - режим ураган (перелік активних і минувших ураганів);

- режим радар і мапи (відображення на мапі насуваючихся циклонів);

– можливість перегляду якості повітря;

– можливість перегляду здоров’я і заходи:

- алергія;
- здоров’я;
- вуличні заходи;
- поїздки та подорожі;
- сад і огород;
- шкідливі комахи.

Загальний опис інтерфейсу:

Веб додаток має 8 сторінок:

Перший екран – екран вибору операцій та поточний прогноз на день. На цьому екрані розташовані:

- навігаційна панель для пошуку необхідної локації;
- 7 кнопок сірого кольору для вибору необхідного прогнозу чи режиму;
- 6 зображень прогнозу погоди.

Вигляд інтерфейсу першого екрану представлений на рис. 1.1.

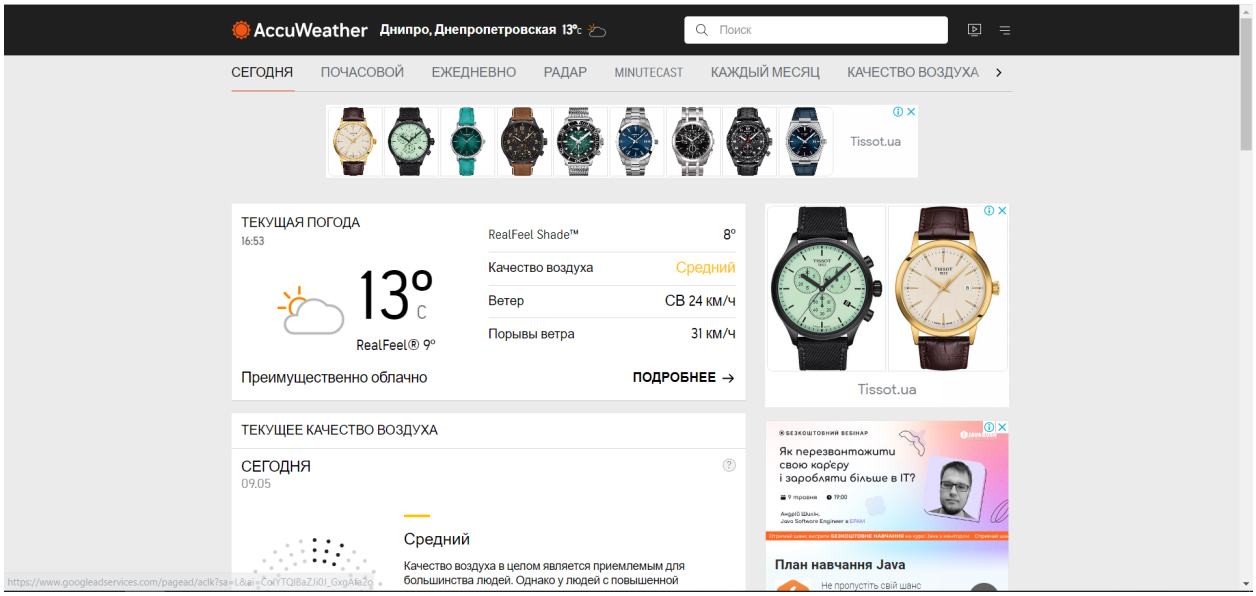

Рис. 1.1. Перший екран

Другий екран - екран погодинного прогнозу. На цьому екрані розташовані:

– (24 – n) зображень прогнозу для кожної години;
– різні погодні дані.

Вигляд інтерфейсу другого екрану представлений на рис. 1.2 та на рис. 1.3.

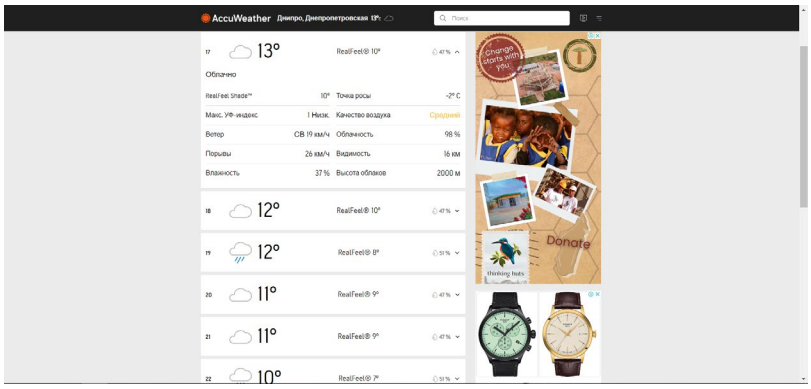

Рис. 1.2. Другий екран

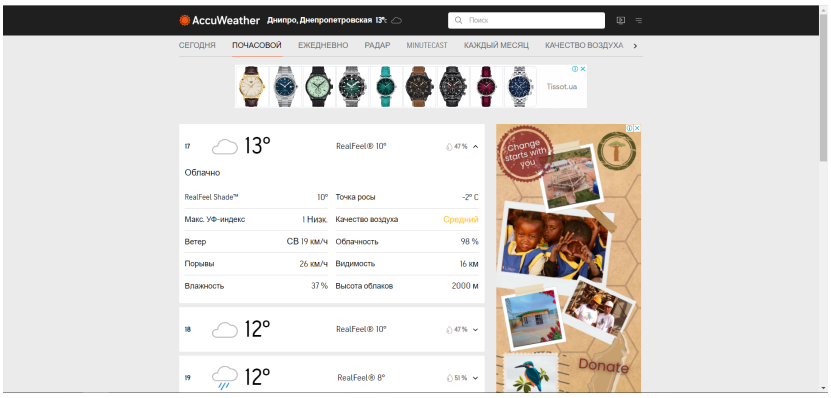

Рис. 1.3. Другий екран

Третій екран – щоденний прогноз. На цьому екрані розташовані:

– 7 елементів для прогнозу на кожень день протягом тижня.

Вигляд інтерфейсу третього екрану представлений на рис. 1.4

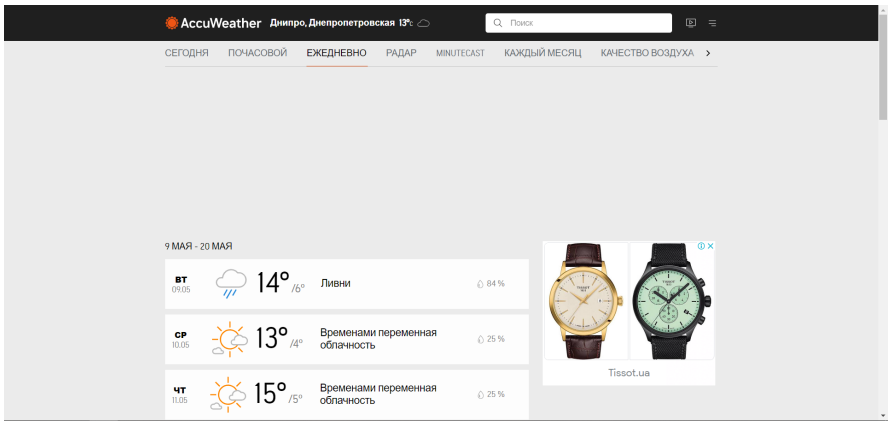

Рис. 1.4. Третій екран

Четвертий екран – сторінка-мапа «радар». На цьому екрані розташована:

– мапа, що відображає циклони, які наближаються.

Вигляд інтерфейсу четвертого екрану представлений на рис. 1.5

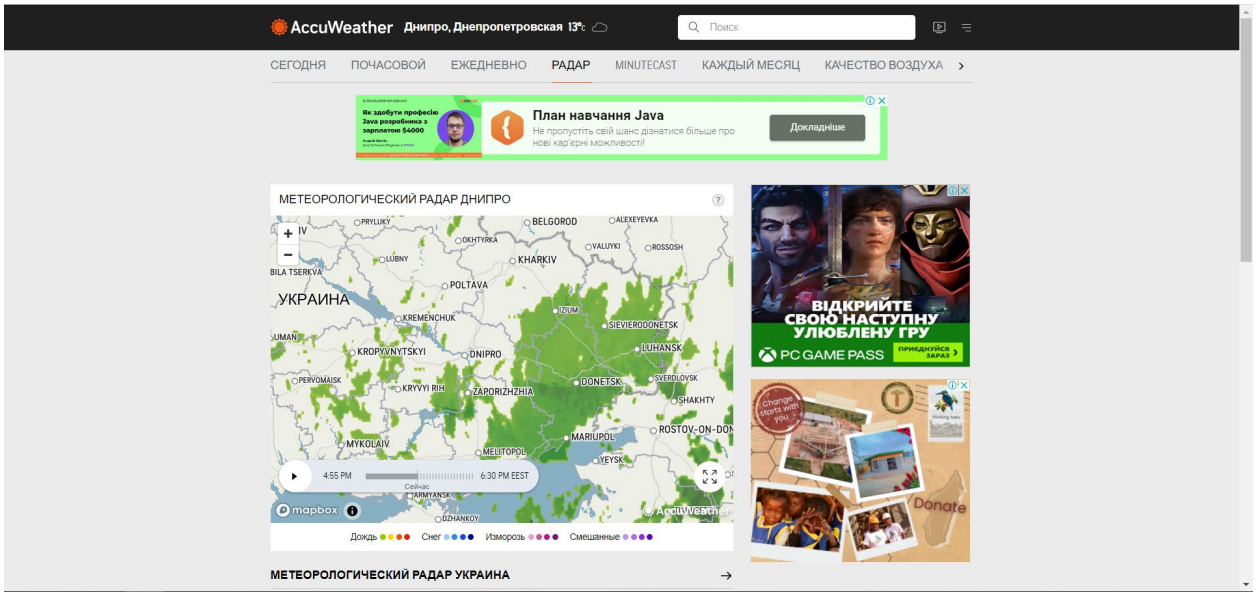

Рис. 1.5. Четвертий екран

П’тий екран – сторінка-графік щохивлинний прогнрз. На цьому екрані розташовано:

– графік, що відображає щохвилинний прогноз.

Вигляд інтерфейсу п’ятий екрану представлений на рис. 1.6

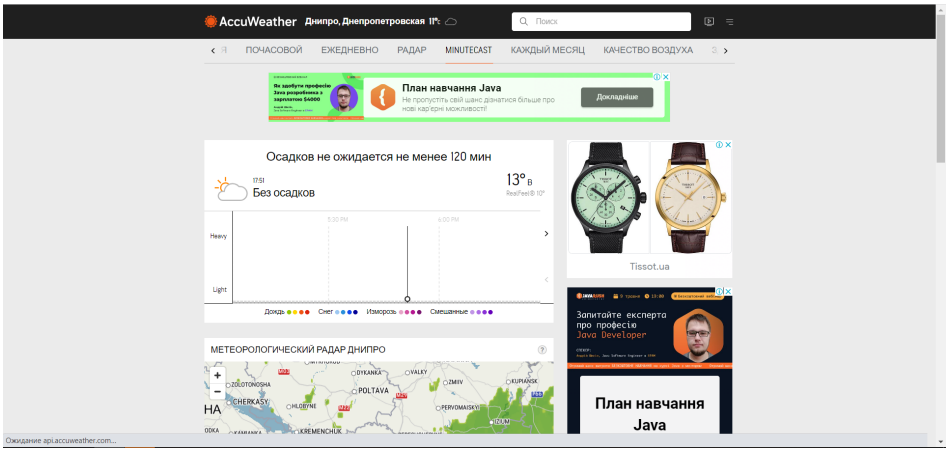

Рис. 1.6. П’ятий екран

Шостий екран – прогноз на місяць. На цьому екрані розташована:

– календар на місяць із прогнозом на день.

Вигляд інтерфейсу шостий екрану представлений на рис. 1.7

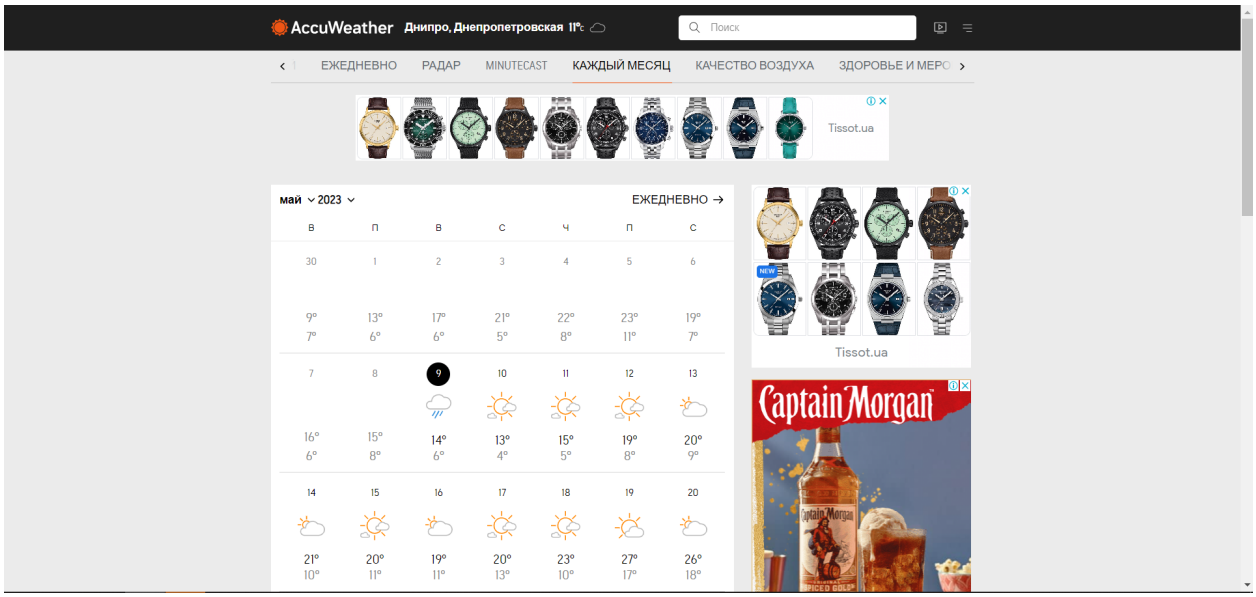

Рис. 1.7. Шостий екран

Сьомий екран – дані про якість повітря. На цьому екрані розташовані:

– поточна якість повітря;

– мапа якості повістря;

– прогноз якості повітря;

– дані про забруднюючі речовини.

Вигляд інтерфейсу сьомого екрану представлений на рис. 1.8

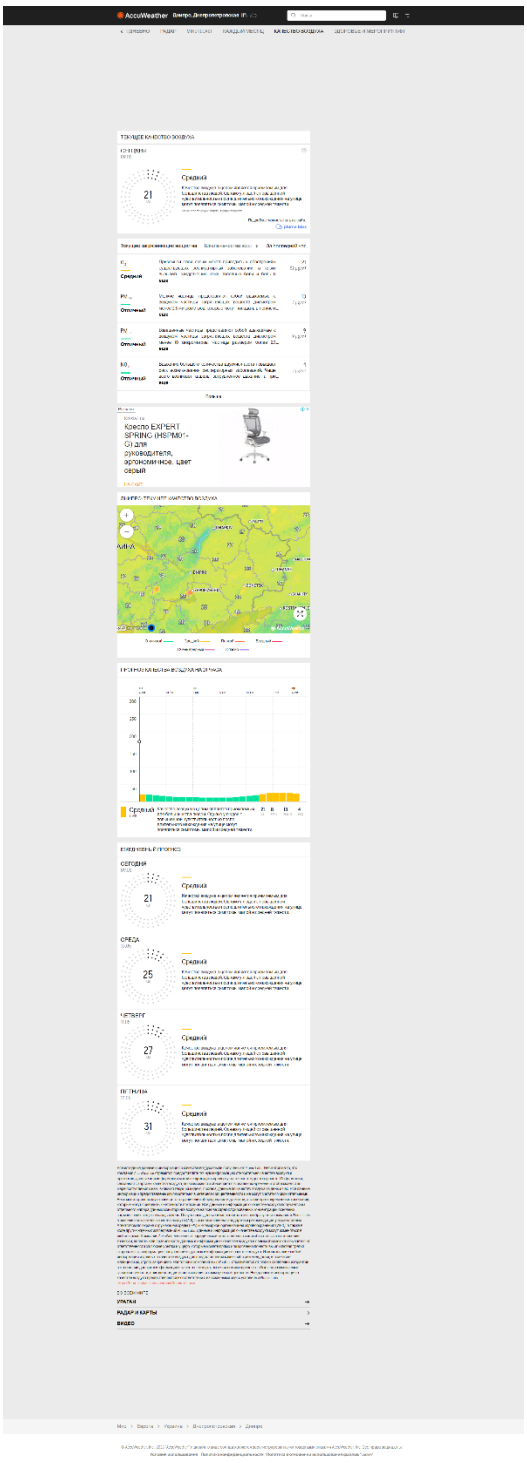

Рис. 1.8. Сьомий екран

Восьмий екран – дані про здоров’я і заходи. На цьому екрані розташована:

– дані для алергіків;
– дані для людей з хворобами;
– дані про вуличні заходи;

– дані для подорожей і поїздок;

– дані про дом і сад.

Вигляд інтерфейсу восьмого екрану представлений на рис. 1.9

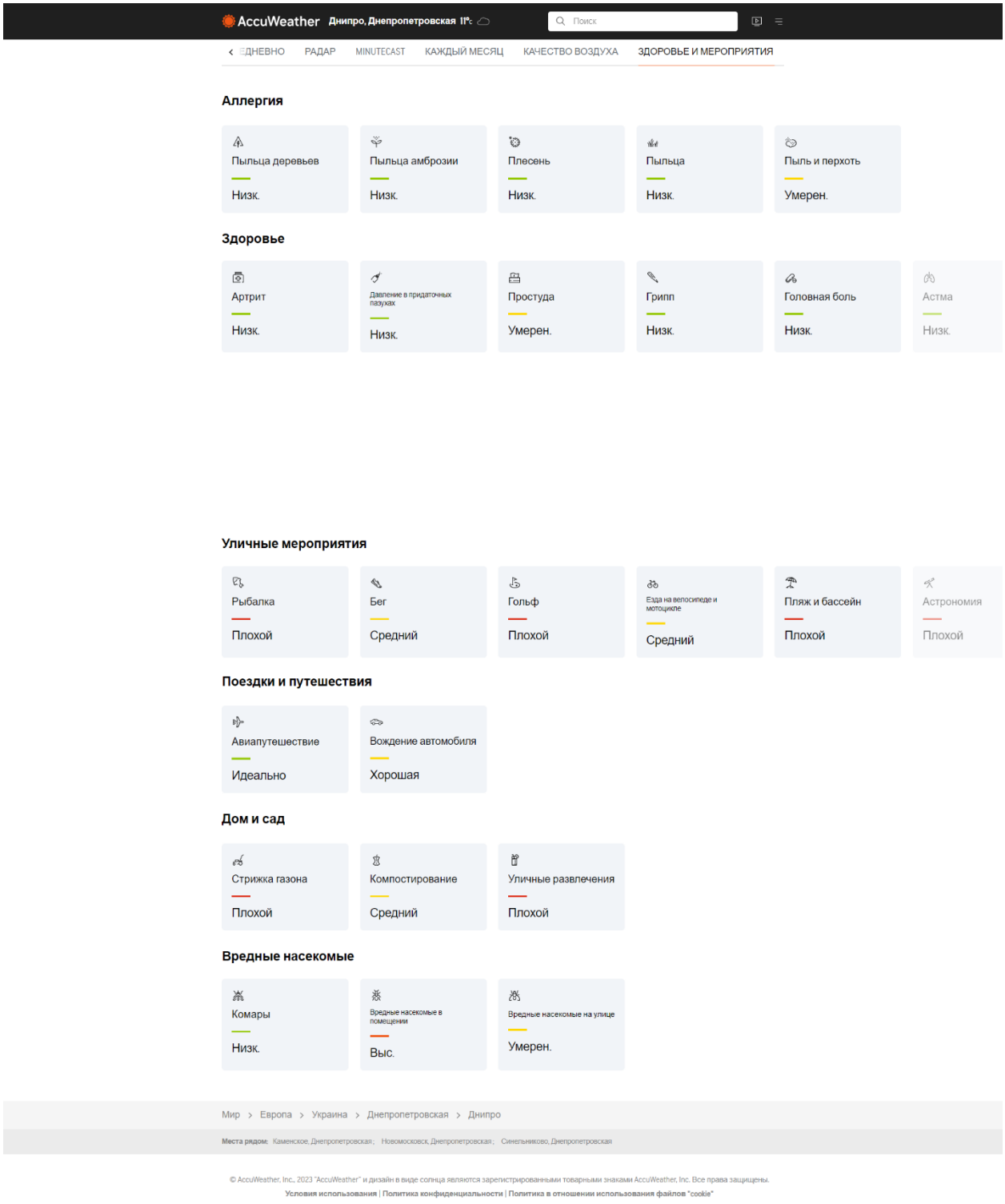

Рис. 1.8. Восьмий екран

2. The Weather Channel Одна з найточніших погодних програм. Воно неймовірно популярне за кордоном. Надає прогнози погоди на кожну годину, на 15 днів та на вихідні. Зі зведень можна дізнатися про реальну і відчутну температуру, швидкість вітру, опади та інтенсивність ультрафіолетового випромінювання.

Загальний опис функціоналу:

– можливість вибору мови інтерфейсу в залежності від країни за континентом;

– можливість вибору прогнозу:

- сьогодні;
- погодинно;
- на 10 днів;
- вихідні;
- щомісяця;
- відстеження алергенів;
- прогноз якості повітря;

– можливість вибору режиму радару.

Загальний опис інтерфейсу:

Мобільний додаток має 8 сторінок:

Перший екран – головна сторінка, яка відображає погоду на сьогодні. На цьому екрані розташовані:

- пошукова панель;
- 5 елементів для відображення погоди;
- 3 показника (якість погоди, здоров’я і активність, радар-мапа із локацією).

Вигляд інтерфейсу першого екрану представлений на рис. 1.8.

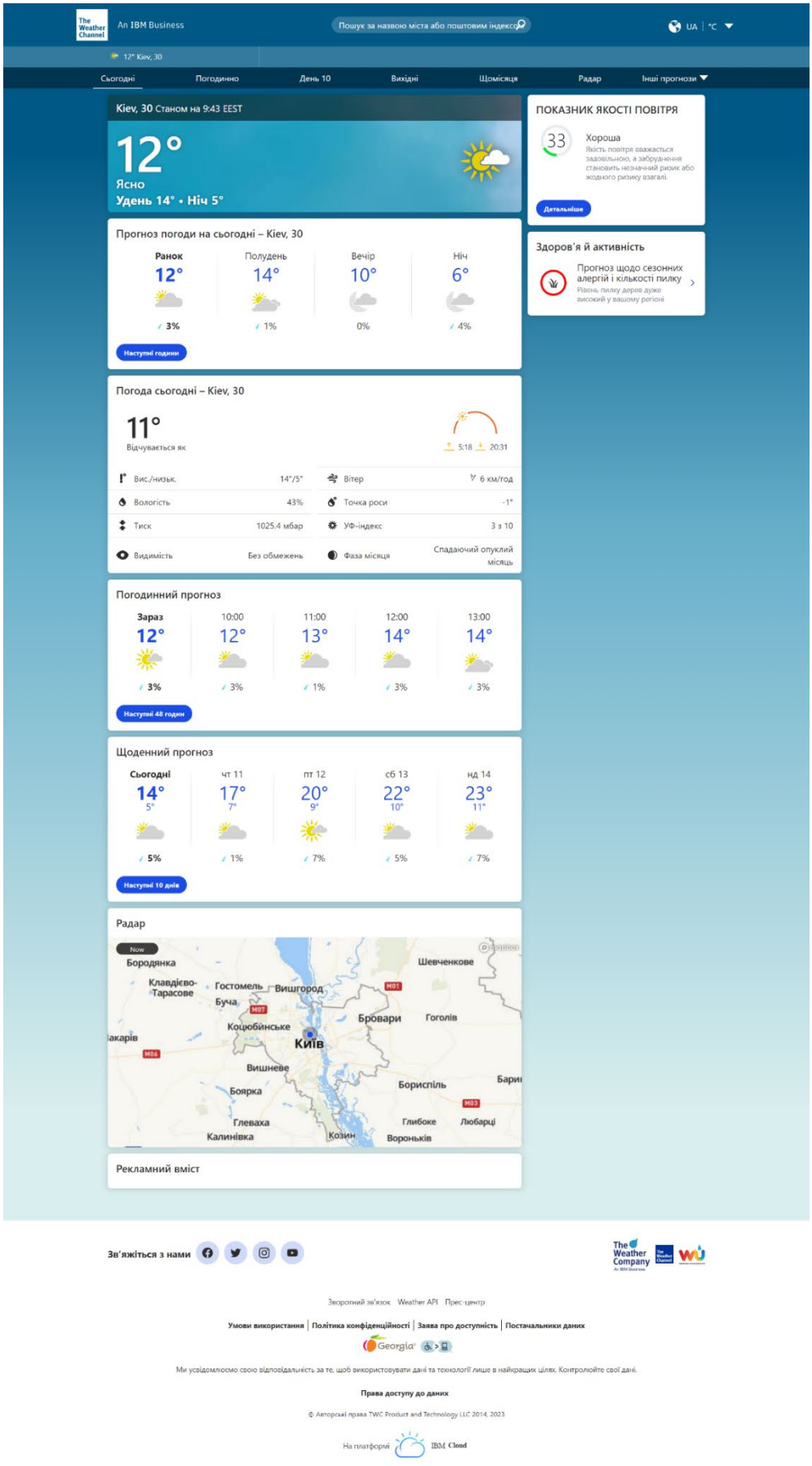

Рис. 1.8. Перший екран

Другий екран – сторінка щогоднинного прогнозу. На цьому екрані розташовані:

– елементи відображення погоди на кожну годину;
– радар з локаціює.

Вигляд інтерфейсу другого екрану представлений на рис. 1.9.

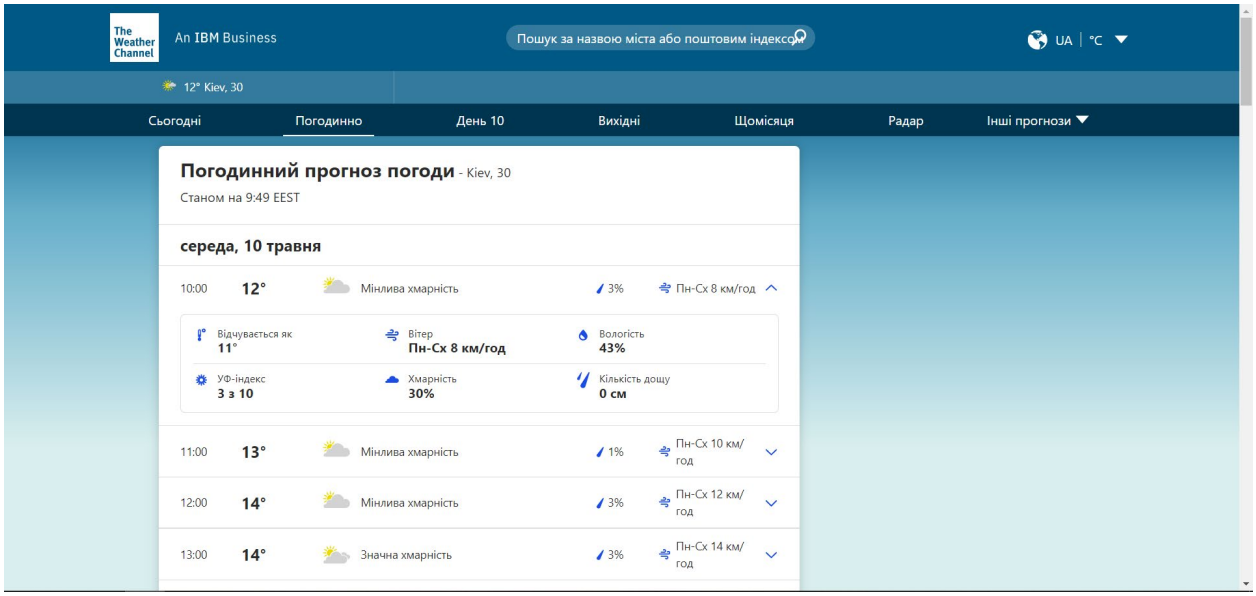

Рис. 1.9. Другий екран

Третій екран – сторінка прогнозу погоди на 10 днів. На цьому екрані розташовані:

– елементи відображення погодних даних і прогнозу погоди.

Вигляд інтерфейсу третього екрану представлений на рис. 1.10.

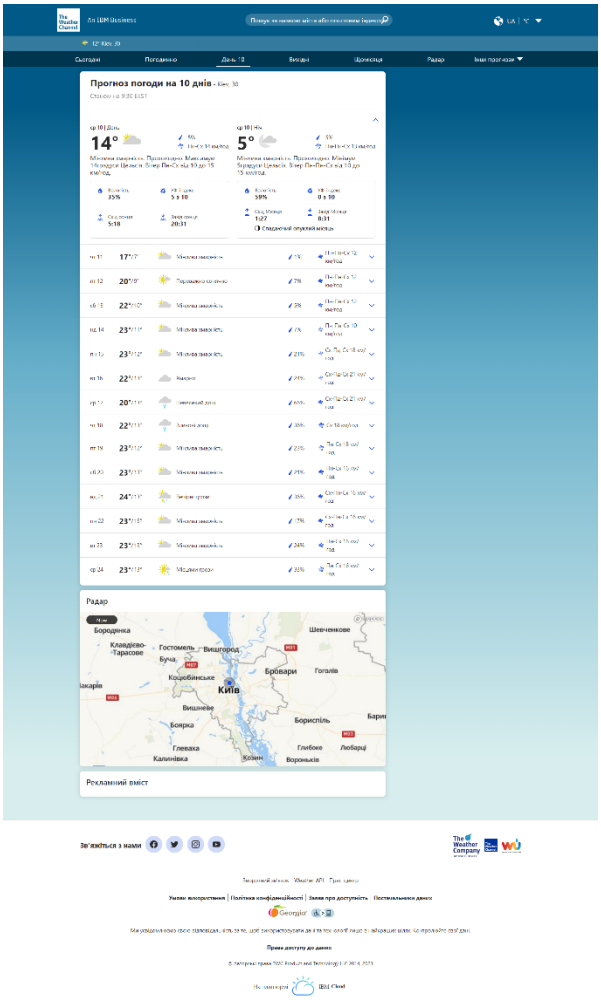

Рис. 1.10. Третій екран

Четвертий екран – сторінка прогнозу погоди вихідні. На цьому екрані розташовані:

– елементи відображення погодних даних і прогнозу погоди.

Вигляд інтерфейсу четвертого екрану представлений на рис. 1.11.

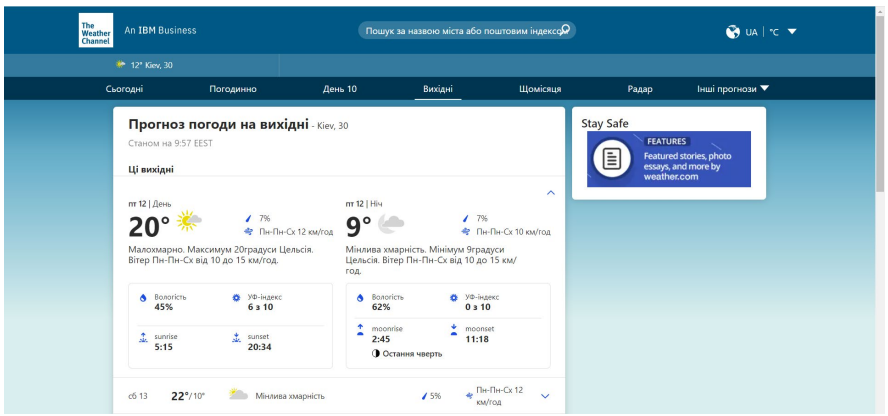

Рис. 1.11. Четвертий екран

П’ятий екран – сторінка прогнозу погоди на місяць. На цьому екрані розташовані:

– елементи відображення погодних даних і прогнозу погоди.

Вигляд інтерфейсу п’того екрану представлений на рис. 1.12.

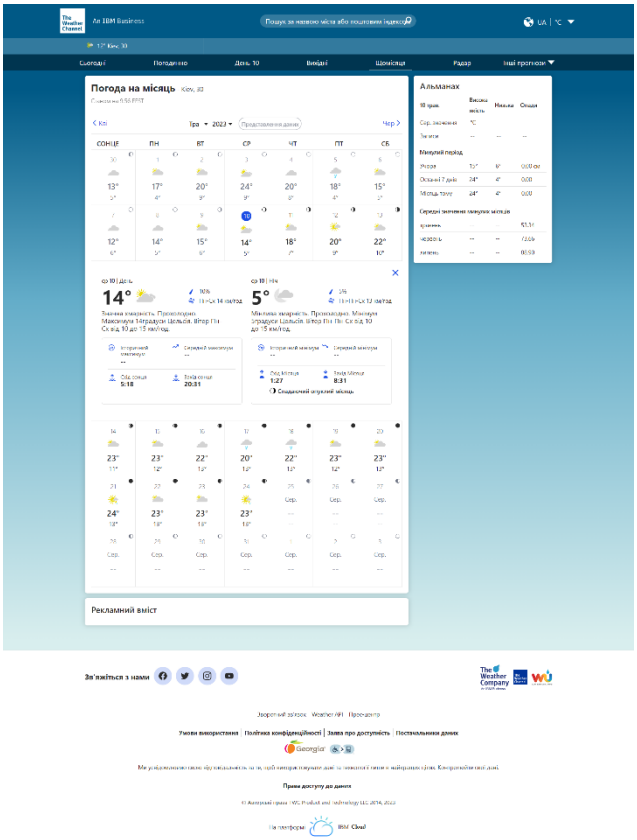

Рис. 1.12. П’ятий екран

Шостий екран – сторінка радару: мапи з відображенням поточної локації. На цьому екрані розташовані:

– мапа з даними про дощ, сніг.

Вигляд інтерфейсу шостого екрану представлений на рис. 1.13.

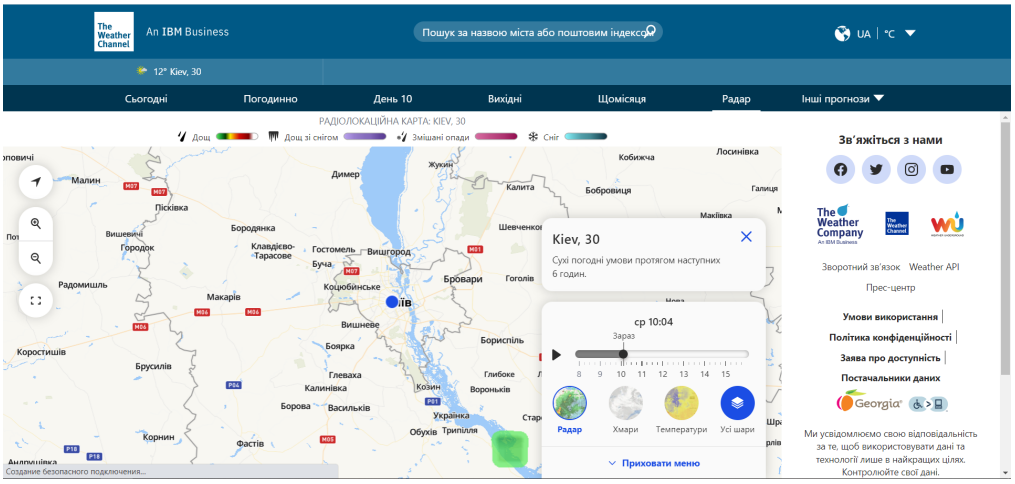

Рис. 1.13. Шостий екран

Сьомий екран – сторінка із даними для людей з алергією. На цьому екрані розташовані:

– елемент з відображенням даних про сезонні алергенні джерела.

Вигляд інтерфейсу сьомий екрану представлений на рис. 1.14.

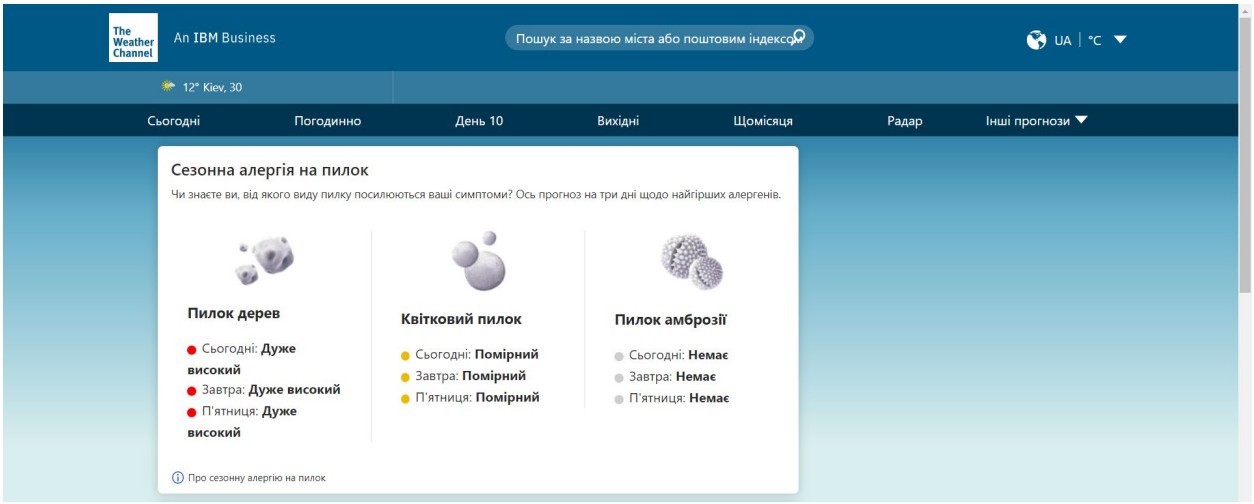

Рис. 1.14. Сьомий екран

Восьмий екран – сторінка якості повітря. На цьому екрані розташовані:

– елемент із відображенням якості повітря.

Вигляд інтерфейсу восього екрану представлений на рис. 1.15.

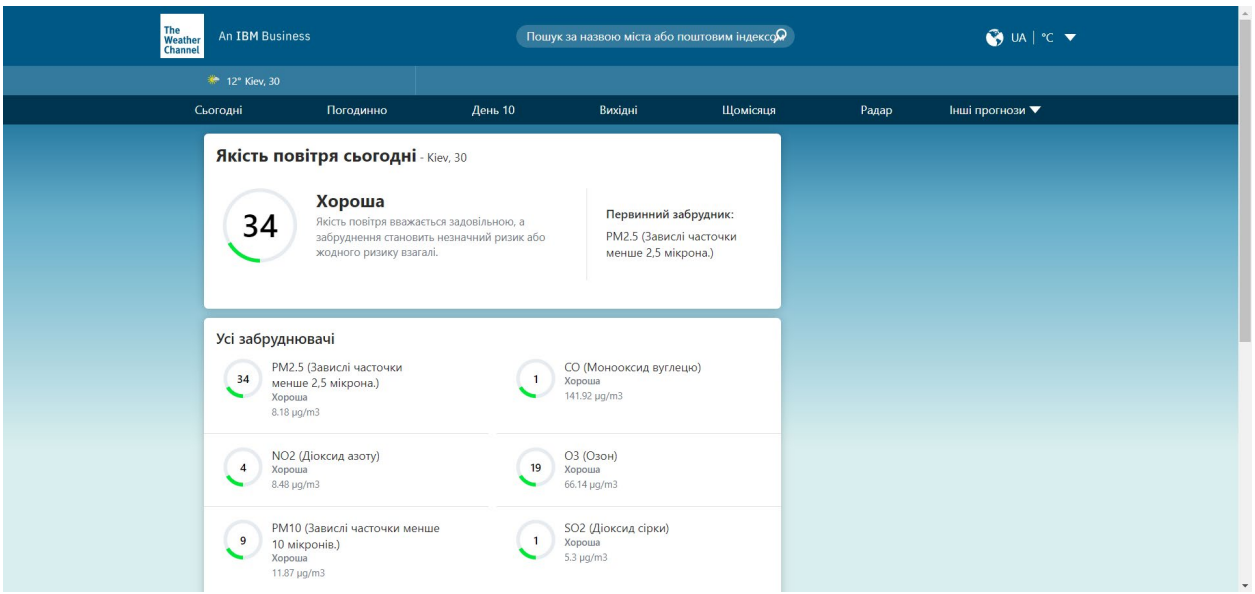

Рис. 1.15. Восьмий екран

3. Досить функціональна програма, в якій є все, чого можна очікувати від погодного інформера. У вашому розпорядженні прогнози з періодичністю від однієї години до 10 днів, радар, більше десятка інтерактивних погодних карт, а також дані про дорожню обстановку, попередження про стихійні лиха та зведення для алергіків.

Загальний опис функціоналу:

– можливість вибору одиниць виміру:
  – температура;
  – швидкіст вітру;
  – тиск;
– можливість вибору прогнозу:
  – сьогодні;
  – погодинно;
  – на 10 днів;
– можливість вибору режиму радара:
  – режим мапи;
  – режим мапи і загрози;
– можливість вибору режиму здоров’я.

Загальний опис інтерфейсу:

Мобільний додаток має 6 екрани:

Перший екран – прогноз на сьогодні. На цьому екрані розташовані:

– пошукова панель;

– кнопка налаштувань;

– 3 кнопки вибору прогнозу;

– 8 посилань на сторінки прогнозу, радару;

– елементи відображення погодних даних.

Вигляд інтерфейсу першого екрану представлений на рис. 1.16.

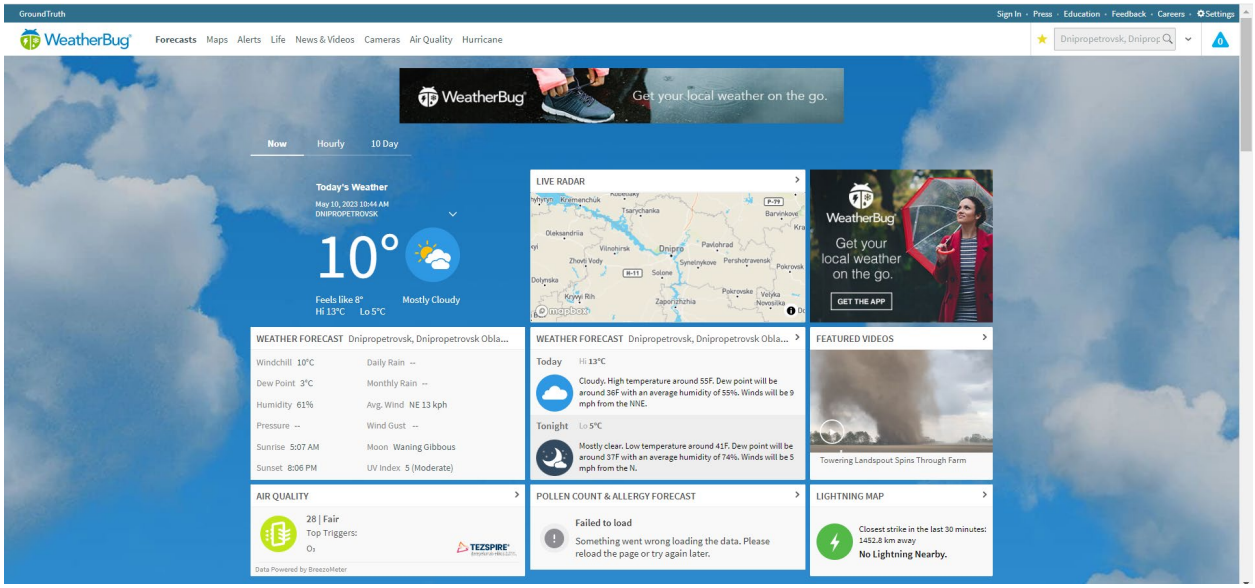

Рис. 1.16. Перший екран

Другий екран – сторінка прогнозу погодинно. На цьому екрані розташовані:

– елементи відобреження погоди на кожну годину.

Вигляд інтерфейсу другого екрану представлений на рис. 1.17.

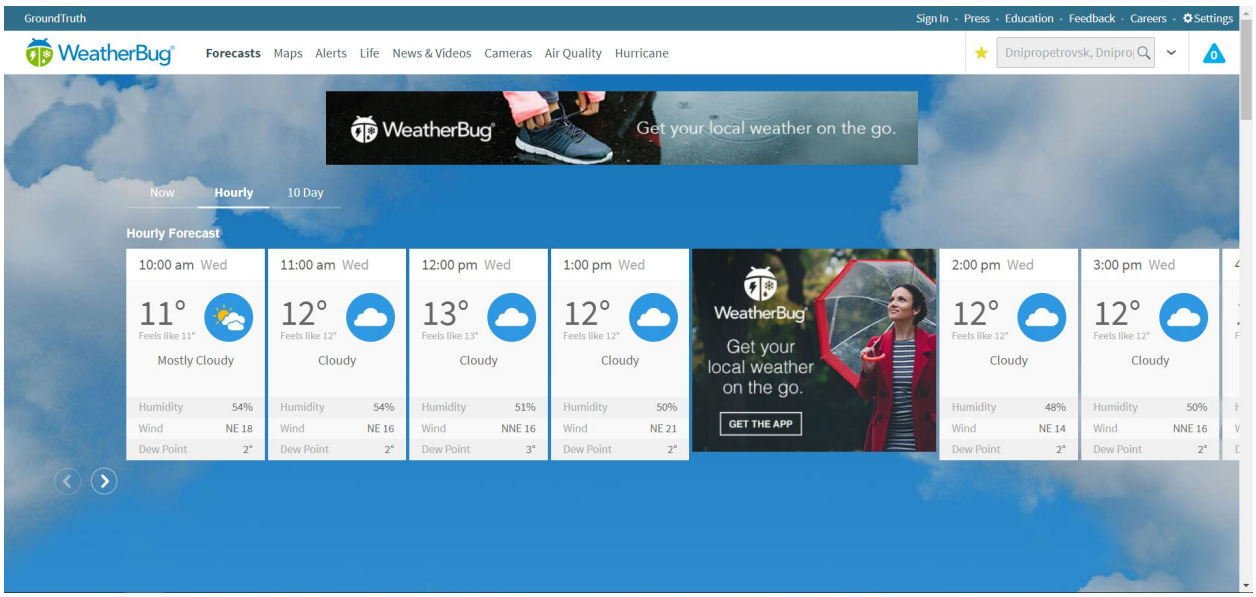

Рис. 1.17. Другий екран

Третій екран – сторінка прогнозу на 10 днів. На цьому екрані розташовані:

– елементи відображення погоди на кожен день протягом 10 днів.

Вигляд інтерфейсу третього екрану представлений на рис. 1.18.

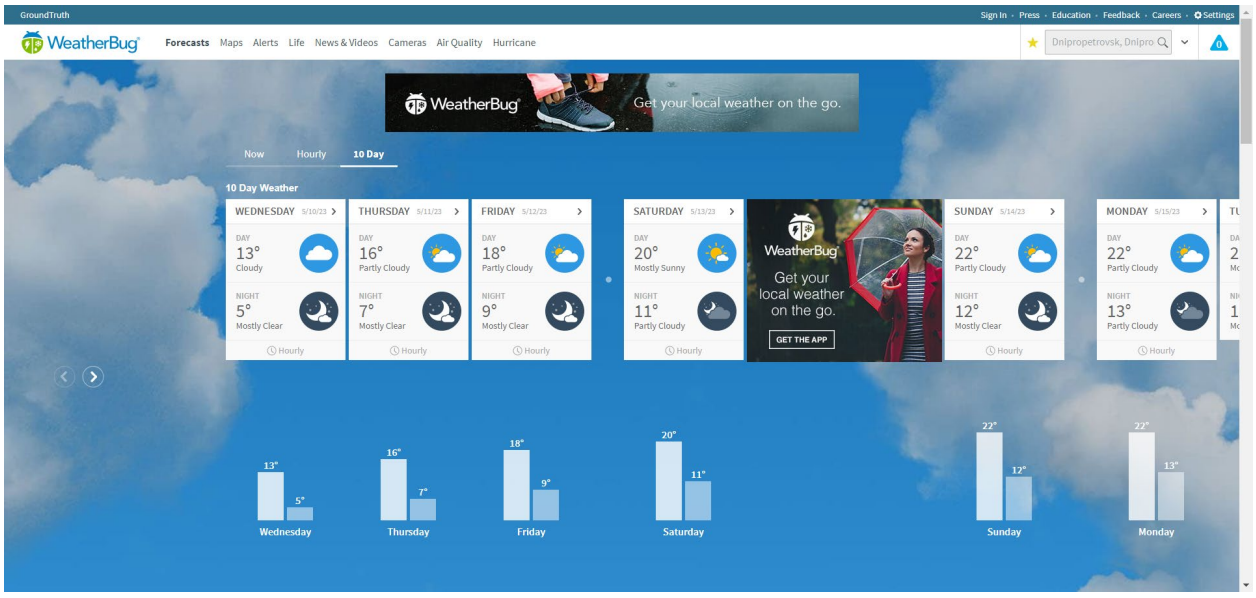

Рис. 1.18. Третій екран

Четвертий екран – сторінка мапи-радару. На цьому екрані розташовані:

– мапа з відображенням циклонів.

Вигляд інтерфейсу четвертого екрану представлений на рис. 1.19.

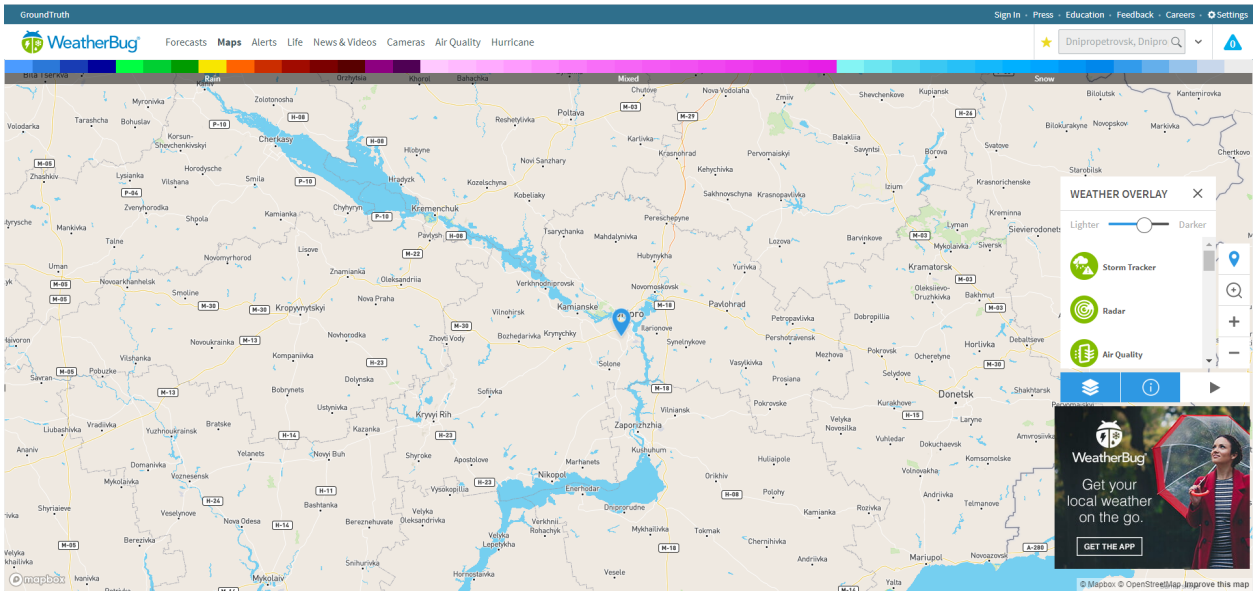

Рис. 1.19. Четвертий екран

П’ятий екран – сторінка з даними про здоров’я. На цьому екрані розташовані:

– елементи відображення даних про здоров’я.

Вигляд інтерфейсу п’того екрану представлений на рис. 1.20.

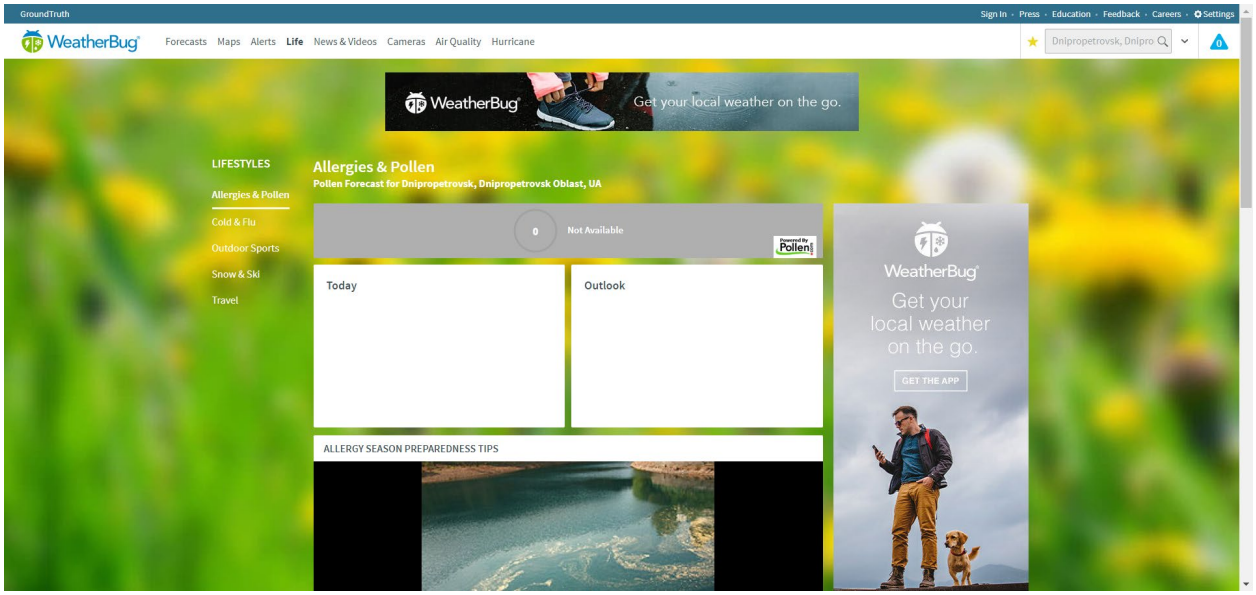

Рис. 1.20. П’тий екран

Шостий екран – сторінка якості повітря. На цьому екрані розташовані:

– мапа із якістю повітря.

Вигляд інтерфейсу шостого екрану представлений на рис. 1.21.

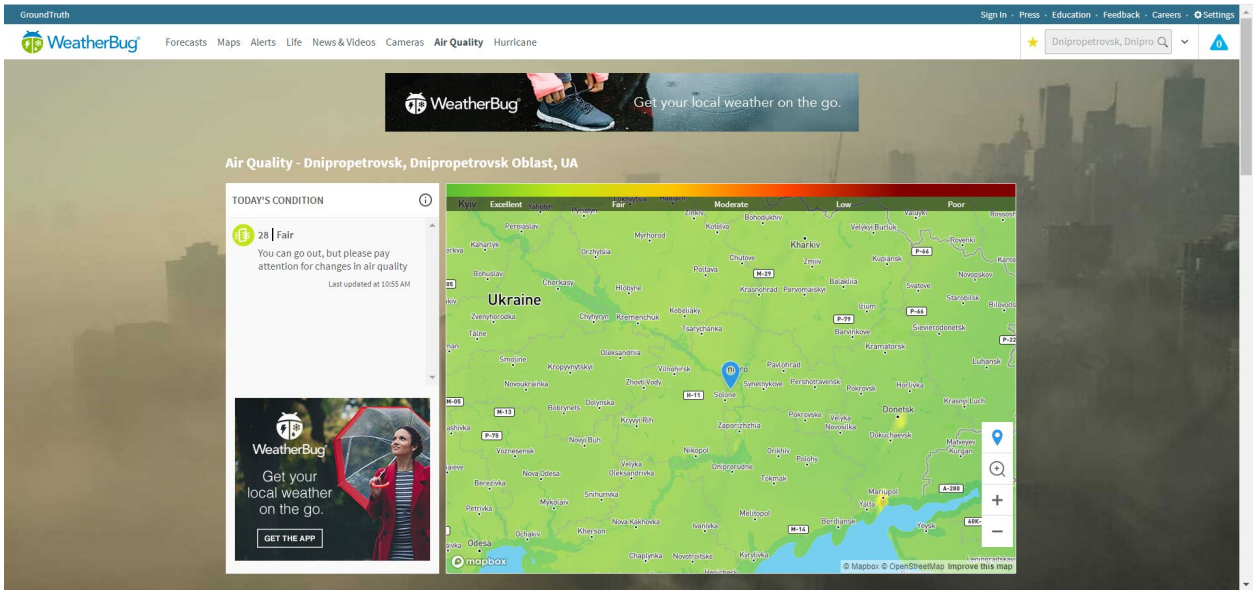

Рис. 1.21. Шостий екран

1.2. Аналіз коментарів та оцінок автору статті аналогів погодних застосунків.

AccuWeather – додаток, що займає перше місце серед всіх представлених і має високу оцінку автора.

Плюси:

– велика кількість режимів;
– збалансована кольорова палітра.

Мінуси:

– не до кінця дороблено інтерфейсну частини додатку.

The Weather Channel – третій за рейтингом веб застосунок, оскільки рейтинг має тільки мобільну частину. Автор відмічає високу точність даних.

Плюси:

– велика кількість режимів;
– інтуїтивно-зрозумілий інтерфейс;

- велика кількіст налаштувань;
- збалансовані кольори інтерфейсу;
- кросплатформеність;
- можливість вибору мови.

Мінуси:

- мала кількість другорядних даних;
- відсутня можливість вибору теми інтерфейсу.

WeatherBug досить середній застосунок, займає 9 місце у рейтингу.

Плюси:

- велика кількість режимів;
- велика кількість посилань на погодні статті;
- унікальність режимів;
- інтуїтивно-зрозумілий інтерфейс;
- велика кількість налаштувань.

Мінуси:

- не зручний механізм вибору варіантів відповідей;
- загруженість контентом;
- платна підписка для відключення реклами.

1.3. Постановка задачі

Постановка задачі для "розробки web-застосунку порівняння прогнозів погоди" полягає у розробці та реалізації онлайн-інструменту, який дозволить користувачам отримувати та порівнювати дані про погоду з різних джерел.

Основною метою цього веб-застосунку є забезпечення користувачам доступу до точної, достовірної та зручно організованої інформації про погоду

з різних метеорологічних джерел. Додаток повинен надавати можливість порівняння даних з різних погодних ресурсів, таких як метеорологічні станції, метеорологічні служби та інші джерела погодної інформації.

Висновки до розділу:

Під час опису аналогів було проаналізовано роботу веб додатків, оцінку виділено їх плюси та мінуси. Опираючись на них можно скласти список основних функціональних вимог та описати план для подальшого проектування та розробки мобільного додатку.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ ДОДАТКУ

2.1. Задачі проекту

На основі проведеного опису аналогів ми виділяємо наступні задачі проекту:

- збір даних: Розробка механізму для збору даних з різних джерел, включаючи API метеорологічних служб та інші доступні джерела інформації про погоду;
- зберігання даних: Створення системи для зберігання та управління отриманими погодними даними, що дозволить ефективно організовувати та зберігати великий обсяг інформації;
- візуалізація даних: Розробка інтуїтивно зрозумілого інтерфейсу користувача, який дозволить візуально представити погодні дані з різних джерел у зручній формі, такі як графіки, діаграми або таблиці;
- порівняння даних: Розробка механізму для порівняння погодних даних з різних джерел та відображення різниць і схожостей між ними.

2.2. Функціональні вимоги

Програмний продукт повинний забезпечувати можливість вибирати локацію для перегляду погоди і прогнозу погоди.

Локація може бути обрана:

– серед запропанованих міст;

– після набирання символів у пошукову строку із подальшим вибором запропанованих варіантів.

Програмний продукт повинний забезпечувати можливість збереження погодних даних у базу даних із відповідним повідомленням.

Програмний продукт повинен забезпечувати можливість відображати збережені дані для обраної локації у вигляді графіку.

Програмний продукт має відображати прогноз погоди на тиждень з поточного дня із можливістю перегляду додаткових даних.

Програмний продукт має генерувати відповіді для збереження і для отримання даних.

Діаграма прецедентів зображена на рис. 2.1

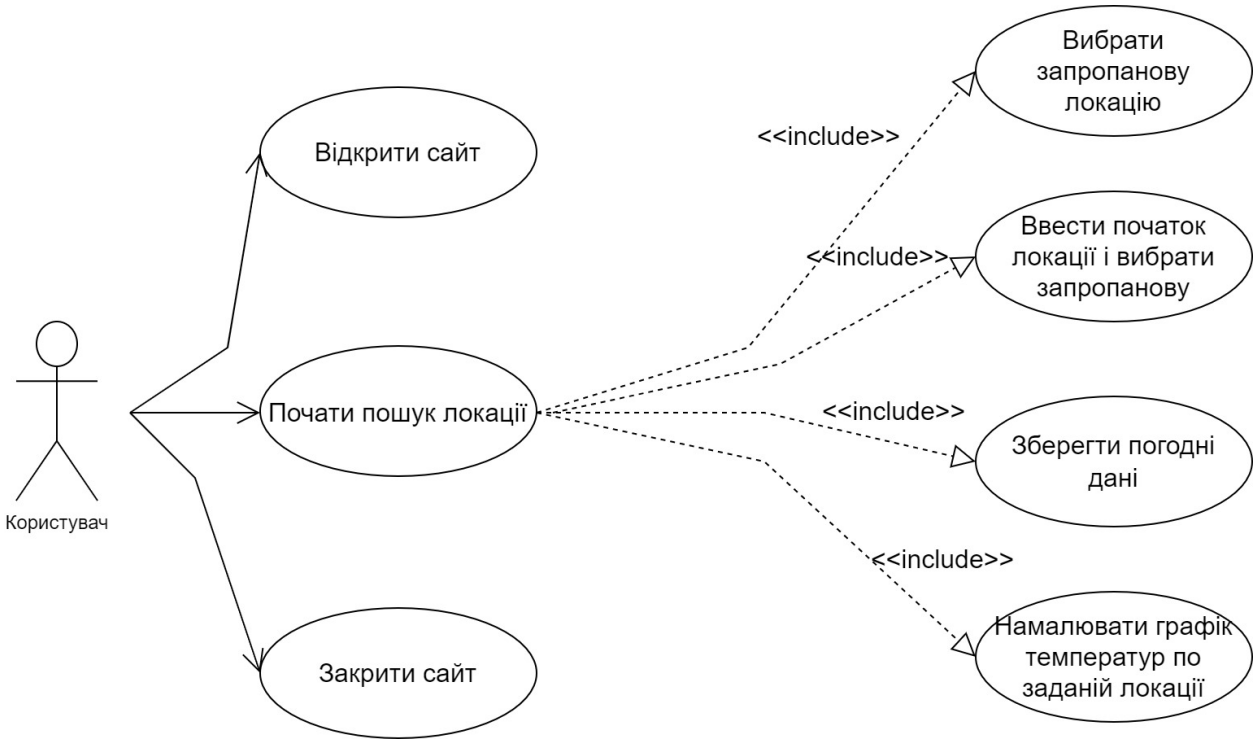

Рис. 2.1. Діаграма прецедентів

2.3. Вхідні та вихідні дані

Веб-додаток передбачає послідовне виконання дій, які не дадуть змоги перейти на наступний етап виконання програми. У разі виключення якогось етапу, на екрані нічого не зміниться.

Вхідними даними є:

– локація для пошуку (символи):

- буквенні символи;

– погодні дані:

- температура трьох погодних сервісів (число);
- локація (строка символів);
- дата збереження.

Без вибору локації не отримаємо погодних даних і не зможемо намалювати графік порівнянь.

Вимоги до вихідних даних:

Вихідними даними є:

– погодні дані з 3 різних станцій:

- температура;
- температура як відчувається;
- вологість;
- тиск;
- опис погоди;
- номер малюнку;

– запропановані локації стандартні або за першими символами.

2.4. Вибір мови програмування

Для розробки веб додатку була вибрана мова програмування JavaScript з використанням фреймоворку React та Express.

При розробці ми переслідували мету використання однієї мови для написання повноцінного додатку для реалізації як клієнської частини так і серверної.

Серед переваг було виділено:

Популярність та екосистема: JavaScript є однією з найпопулярніших мов програмування у світі веб-розробки. Це означає, що буде доступ до великої кількості ресурсів, документації, бібліотек та фреймворків, які підтримуються спільнотою. JavaScript також підтримується більшістю сучасних веб-

браузерів, що дозволяє виконувати код на клієнтській стороні без необхідності встановлення додаткових плагінів.

Реактивний підхід: React є одним з найпопулярніших фреймворків для розробки користувацького інтерфейсу. Він пропонує компонентний підхід до побудови інтерфейсу, де кожен елемент (компонент) може бути перефакторизований, перевикористаний та керований окремо. Це полегшує розробку складних користувацьких інтерфейсів і сприяє покращенню швидкодії додатків. З React ви також можете використовувати JSX, який дозволяє комбінувати HTML-подібний код з JavaScript, що полегшує розробку та розуміння коду.

Серверна сторона: Express є легким та гнучким фреймворком для створення серверних додатків на JavaScript. Він надає можливості для маршрутизації, обробки запитів, створення API та багато іншого. Express дозволяє швидко створювати сервери з мінімальним навантаженням, а також має велику спільноту розробників, що забезпечує наявність підтримки.

2.5. Опис компонентно-орієнтованого підходу

Компонентно-орієнтоване програмування це підхід у якому набір компонентів взаємодіють з іншими компонентами через контейнер. Кожний побудований нами компонент являє собою окрему підпрограму, яка описує свою функціональність та також дає право до змін інтерфейсу.

Серед переваг компонентно-орієнтованого підходу над об’єктно-орієнтованим в нашому випадку можно виділити:

Перевикористання коду: Компонентно-орієнтований підхід дає можливість перевикористовувати компоненти в різних частинах додатку. Кожен компонент може бути незалежним імітатором, який можна використовувати в різних контекстах. Це спрощує розробку, підтримку і оновлення коду, оскільки зміни в одному компоненті автоматично поширюються на всі місця, де він використовується.

Декларативний підхід: React пропонує декларативний підхід до розробки інтерфейсу. Замість безпосереднього втручання в DOM (Document Object Model), ви описуєте, як ваш інтерфейс повинен виглядати на різних станах додатку, а React самостійно оновлює DOM, забезпечуючи ефективне рендерінг та реагування на зміни.

Швидкість та ефективність: React використовує віртуальний DOM (Virtual DOM), що дозволяє ефективно оновлювати лише ті частини інтерфейсу, які змінилися. Замість повного перерендерингу всієї сторінки React порівнює віртуальний DOM з реальним DOM та виконує мінімальні зміни, що приводить до зменшення навантаження на браузер та покращує продуктивність додатку.

Управління станом: React має потужні засоби для управління станом додатку. Зокрема, він пропонує вбудований механізм під назвою "стейт" (state), який дозволяє зберігати та оновлювати дані в компонентах.

Програма яка написана з використанням компонентно-орієнтованого підходу поділена на велику кількість модулів, що дозволяє швидко виправляти, змінювати та доповнювати функціональності. Таким чином, цей підхід виправляє більшість недоліків попередника та привносить багато переваг, таких як легка структурованість, виправлення та доповнюваність, що особливо важливо у розробці масштабних програмних продуктів.

2.6. Проектування екранів прототипу веб додатку

Прототип веб додатку складатимесь із однією сторінки, проти із дорисуванням нових компонентів:

– компонент пошуку;

– компоненти погоди на сьогодні для представлених погодних сервісів;

– компонент прогнозу погоди для одного з погодних сервісів;

– компонент графік.

2.7. Проектування інтерфейсу користувача прототипу веб додатку

Прототип веб додатку повинен мати такі компоненти:

– компонент “Search” (бібліотечна компонента):
  – поле вводу;
  – кнопка пошуку;
  – випадаючий список.
– компонент “CurrentWeather”:
  – текстове поле місто;
  – текстове поле опис погоди;
  – іконка погоди;
  – текстове поле температури;
  – текстове поле як відчувається температура;
  – текстове поле швидкості вітру;
  – текстове поле вологість;
  – текстове поле тиску.
– компонент “Forecast” (бібліотечна компонента):
  – текстове поле температури;
  – текстове поле як відчувається температура;
  – текстове поле швидкості вітру;
  – текстове поле вологість;
  – текстове поле тиску.
– компонент “Chart”.

Компонент Search - компонент інтерфейсу через який здійнюється пошук міста, для якого виконується запит погодних даних.

Компонент CurrentWeather – компонент інтерфейсу, який виводить на екран температурні дані з метеостанції.

Компонент Forecast – компонент інтерфейсу, який виводить на екран температурні дані на наступні 7 днів.

Компонент Chart – компонент інтерфейсу, який виводить на екран графік з порівнянням температурних даних із різних погодних сервісів.

2.8. Проектування порядку роботи із компонентами прототипу веб додатку

Для послідовного і коректоного виконання веб-додатку слід дотримуватись наступного плану дій, які були продемонстровані за допомогою діаграми станів:

На рис. 2.2 приведена діаграма інтерфейсу користувача

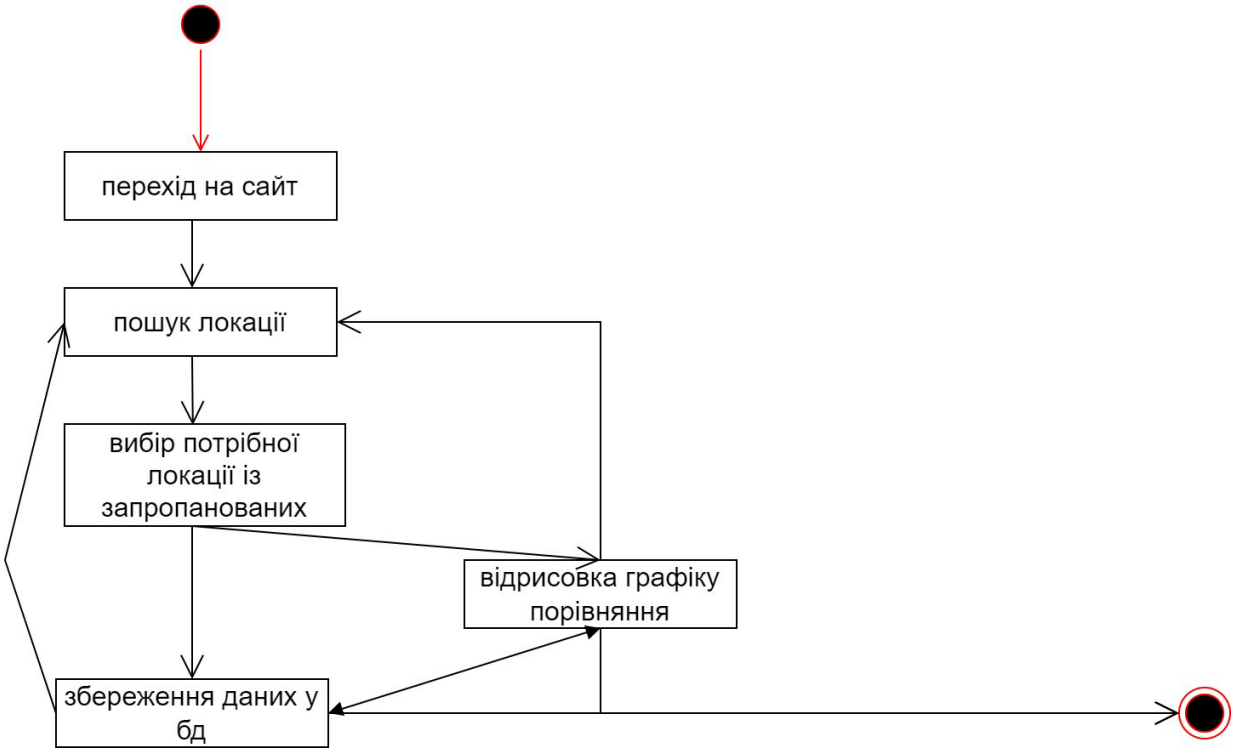

Рис. 2.2. Діаграма станів

2.9. Проектування дизайну прототипу мобільного додатку

Для кожного з компонентів веб додатку були розроблені ескізи.

Компоненти:

– компонент Search розмір середній займає 80% ширини екрану;

– компонент CurrentWeather розмір середній квадратоподібний, займає 20-30% оскільки таких буде 3;

– компонент Forecast розмір середній займає 80% ширини екрану, по висоті маленький, оскільки відображаєтсья 7 таких компонентів поспіль;

– компонент Chart розмір великий, квадратоподібний елемент графік, займає 50% по висоті і 90% по ширині екрану.

Ескізи форм компонентів приведені на рис. 2.3

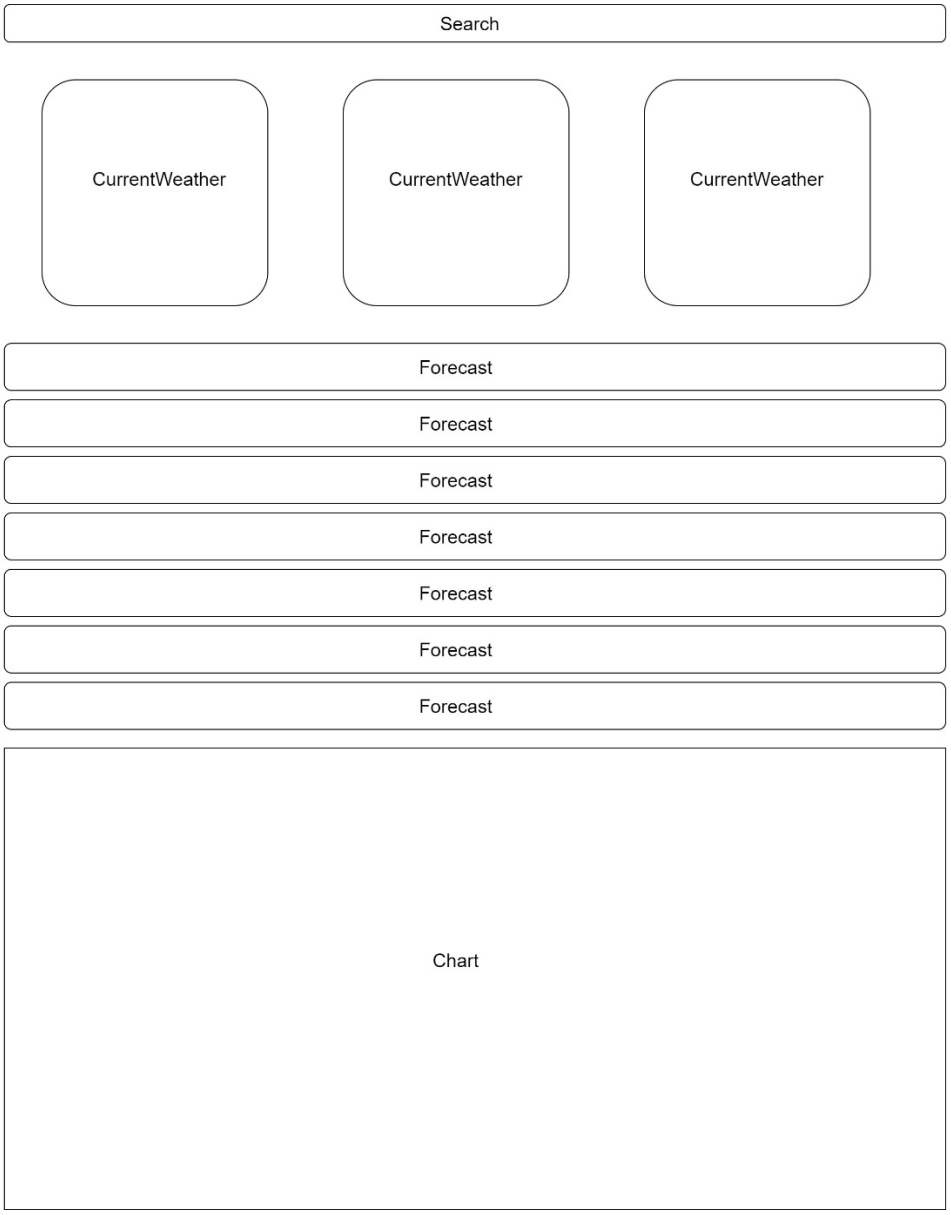

Рис. 2.3. Ескізи форм компонентів

2.10. Розробка елементів прототипу веб додатку

Для кожного елементу було створено функцію для відображення компонентів:

– компонент пошуку:

```
const  Search = () => {
  return <AsyncPagionate/>;
}
```

– компонент поточної погоди:

```
const CurrentWeather = () => {
  return <div></div>;
}
```

– компонент прогнозу:

```
const Forecast = () => {
  return <div></div>;
}
```

– компонент графіку:

```
const LineChart = () => {
  return <Line/>;
}
```

2.11. Розробка інтерфейсу користувача прототипу веб додатку

Для кожного з визначень було створено свій стиль та компонент:

– компонент “Search”:

– велике поле вводу.

Компонент Search:

```
<AsyncPaginate
    placeholder="Search for city"
    debounceTimeout={600}
    value={search}
    onChange={handleOnChange}
    loadOptions={loadOptions}
  />
```

Стиль комопненту:

```
* {
    display: flex;
    }
```

Вигляд компоненту – поле вводу зображено на рис. 2.4

Рис. 2.4. Вигляд компоненту – поле вводу

Компонентом CurrentWeather:

```
<div className="weather">
    <div className="top">
      <div >
        <p className="city">{data.city}</p>
        <p className="weather-description">{data.description}</p>
      </div>
      <img alt="weather" className="weather-icon" src={`icons/${data.icon}.png`} />
    </div>

    <div className="bottom">
      <p className="temperature">{Math.round(data.temperature)}°C</p>
      <div className="details">
        <div className="parameter-row">
          <span className="parameter-label">Details</span>
        </div>

        <div className="parameter-row">
          <span className="parameter-label">Feels like</span>
          <span className="parameter-value">{Math.round(data.feels_like)}°C</span>
        </div>

        <div className="parameter-row">
          <span className="parameter-label">Wind</span>
          <span className="parameter-value">{data.wind_speed} m/s</span>
        </div>
```

```
        <div className="parameter-row">
          <span className="parameter-label">Humidity</span>
          <span className="parameter-value">{data.humidity}%</span>
        </div>

        <div className="parameter-row">
          <span className="parameter-label">Pressure</span>
          <span className="parameter-value">{data.pressure} hPa</span>
        </div>
      </div>
    </div>
  </div>
```

Стиль комопненту:

```
.weather {
  width: 300px;
  border-radius: 6px;
  box-shadow: 10px -2px 20px 2px rgb( 0 0 0 / 30%);
  color: #fff;
  background-color: #333;
  margin: 20px auto;
  padding: 0 20px 20px 20px;
}

.top,
.bottom {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.city {
  font-weight: 600;
  font-size: 18px;
  line-height: 1;
  margin: 0;
  letter-spacing: 1px;
}
```

```
.weather-description {
  font: 400;
  font-size: 14px;
  line-height: 1;
  margin: 0;
}

.weather-icon {
  width: 100px;
}

.temperature {
  font-weight: 600;
  font-size: 70px;
  width: auto;
  letter-spacing: -5px;
  margin: 10px 0;
}

.details {
  width: 100%;
  padding-left: 20px;
}

.parameter-row {
  display: flex;
  justify-content: space-between;
}

.parameter-label {
  text-align: left;
  font-weight: 400;
  font-size: 12px;
}

.parameter-value {
  text-align: right;
  font-weight: 600;
```

```
  font-size: 12px;
}
```

Вигляд компоненту зображено на рис. 2.5

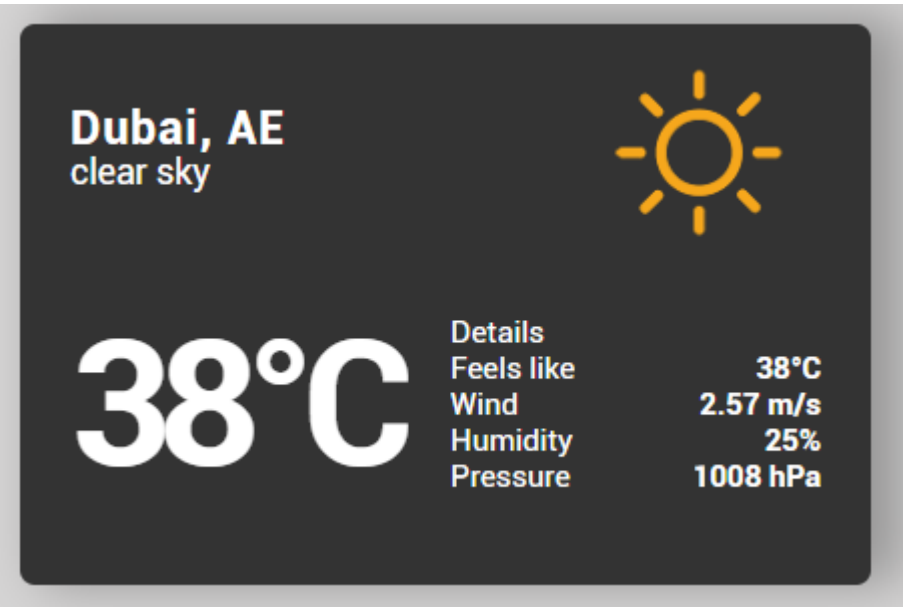

Рис. 2.5. Вигляд компоненту

– велика кнопка.

Компонент Forecast:

```
<>
   <label className="title">Daily</label>
   <Accordion allowZeroExpanded>
     {data.list.splice(0, 7).map((item, idx) => (
      <AccordionItem key={idx}>
        <AccordionItemHeading>
          <AccordionItemButton>
            <div className='daily-item'>
              <img alt='weather' className='icon-small' src={`icons/${item.weather[0].icon}.png`}/>
              <label className='day'>{forecastDays[idx]}</label>
              <label className='description'>{item.weather[0].description}</label>
              <label className='min-max'>{Math.round(item.main.temp_min)}°C
              / {Math.round(item.main.temp_max)}°C</label>
            </div>
          </AccordionItemButton>
        </AccordionItemHeading>
        <AccordionItemPanel>
```

```
          <div className='daily-details-grid'>
           <div className='daily-details-grid-item'>
            <label>Pressure</label>
            <label>{item.main.pressure} hPa</label>
           </div>

           <div className='daily-details-grid-item'>
            <label>Humidity</label>
            <label>{item.main.humidity}%</label>
           </div>

           <div className='daily-details-grid-item'>
            <label>Clouds</label>
            <label>{item.clouds.all}%</label>
           </div>

           <div className='daily-details-grid-item'>
            <label>Wind speed</label>
            <label>{item.wind.speed} m/s</label>
           </div>

           <div className='daily-details-grid-item'>
            <label>Sea level: </label>
            <label>{item.main.sea_level} m</label>
           </div>

           <div className='daily-details-grid-item'>
            <label>Feels like: </label>
            <label>{Math.round(item.main.feels_like)}°C</label>
           </div>
          </div>
         </AccordionItemPanel>
        </AccordionItem>
      ))}
     </Accordion>
    </>
```

Стиль контейнеру:

```
.title {
```

```
  font-size: 23px;
  font-weight: 700;
}

.daily-item {
  background-color: #f5f5f5;
  border-radius: 15px;
  height: 40px;
  margin: 5px;
  display: flex;
  align-items: center;
  cursor: pointer;
  font-size: 14px;
  padding: 5px 20px;
}

.icon-small {
  width: 40px;
}

.day {
  color: #212121;
  flex: 1 1;
  font-weight: 600;
  margin-left: 15px;
}

.description {
  flex: 1 1;
  margin-right: 15px;
  text-align: right;
}

.min-max {
  color: #757575;
}

.daily-details-grid {
```

```
  grid-row-gap: 0;
  grid-column-gap: 15px;
  row-gap: 0;
  column-gap: 15px;
  display: grid;
  flex: 1 1;
  grid-template-columns: auto auto;
  padding: 5px 15px;
}

.daily-details-grid-item {
  display: flex;
  height: 30px;
  align-items: center;
  justify-content: space-between;
}

.daily-details-grid-item label:first-child {
  color: #757575;
}

.daily-details-grid-item label:last-child {
  color: #212121;
}
```

Вигляд компоненту зображено на рис. 2.6

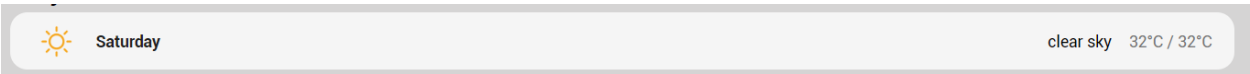

Рис. 2.6. Вигляд компоненту

Компонент LineChart:

```
< Line data={chartData} />
```

Стиль контейнеру:

```
* : {

}
```

Вигляд компоненту зображено на рис. 2.7

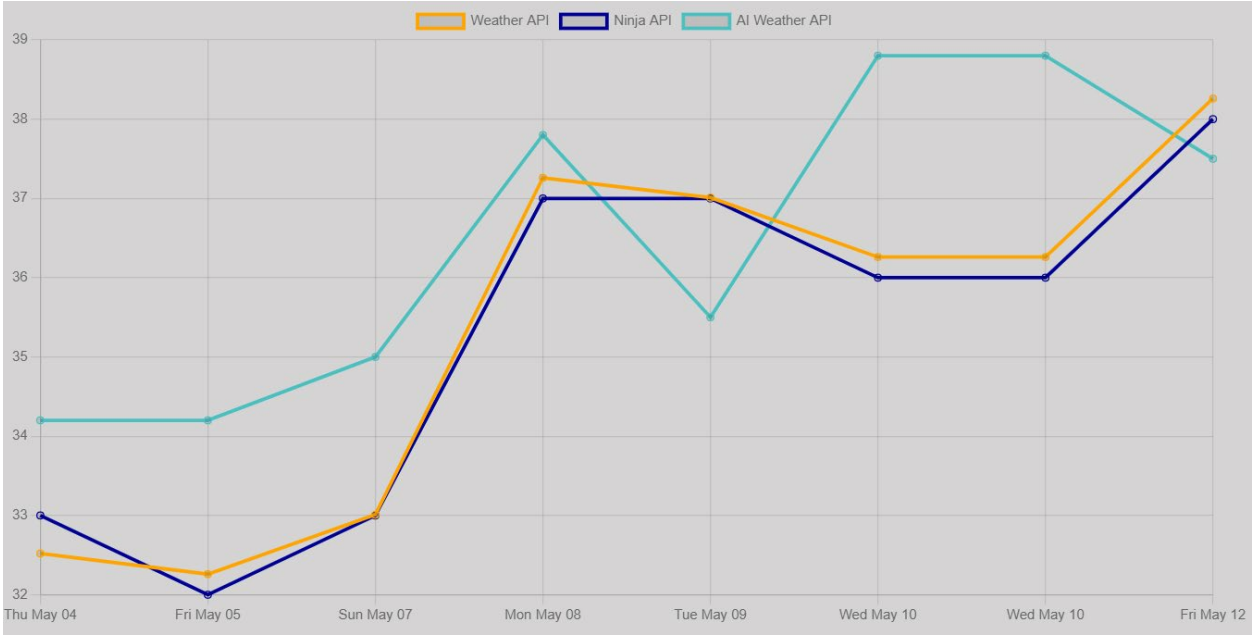

Рис. 2.7. Вигляд компоненту – scrollview

2.12. Структура компонентів системи

Система компонентів зображена на рис. 2.8

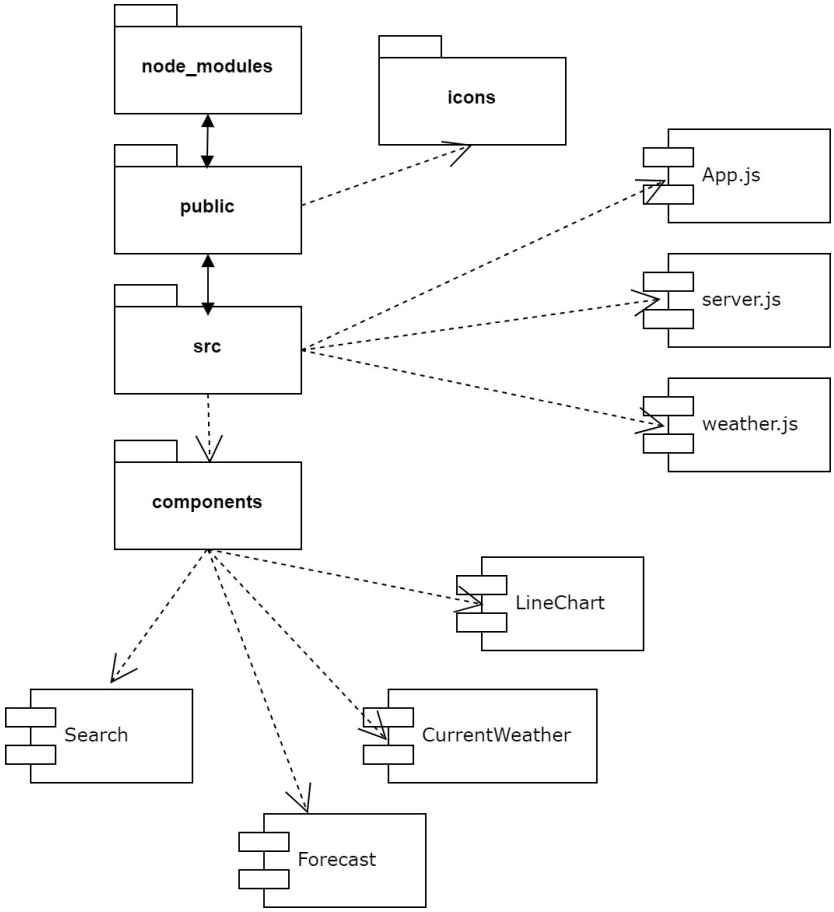

Рис. 2.8. Структура компонентів системи

2.13. Розробка дизайну прототипу веб додатку

Головний екран зображений на рис. 2.9

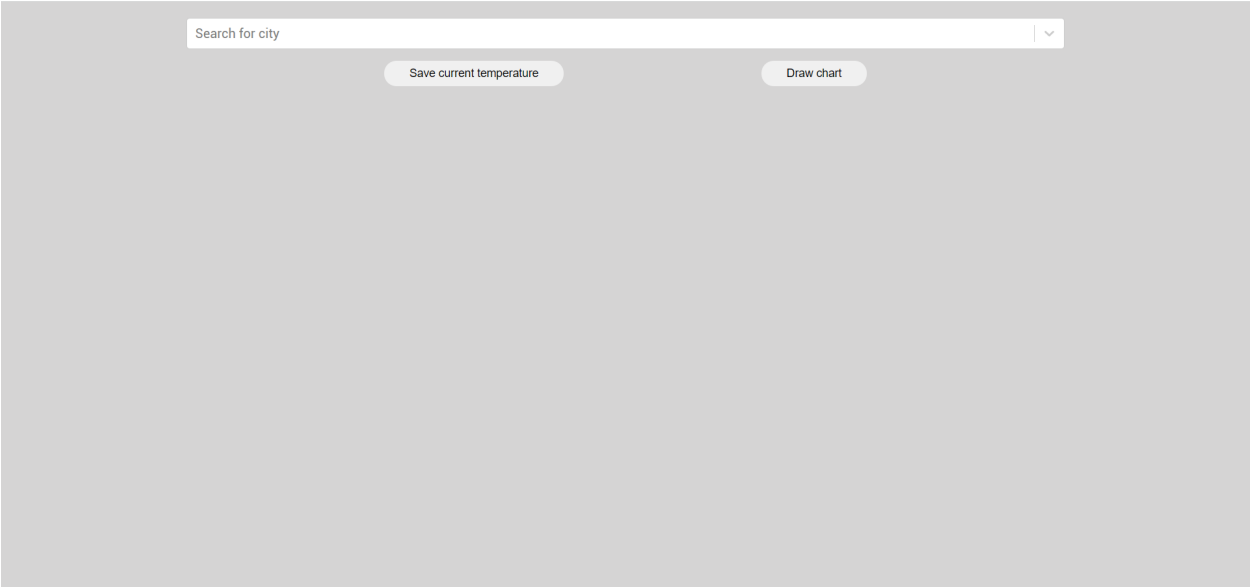

Рис. 2.9. Головний екран

Екран із потоною температурою і прогнозом на 7 днів зображений на рис. 2.10

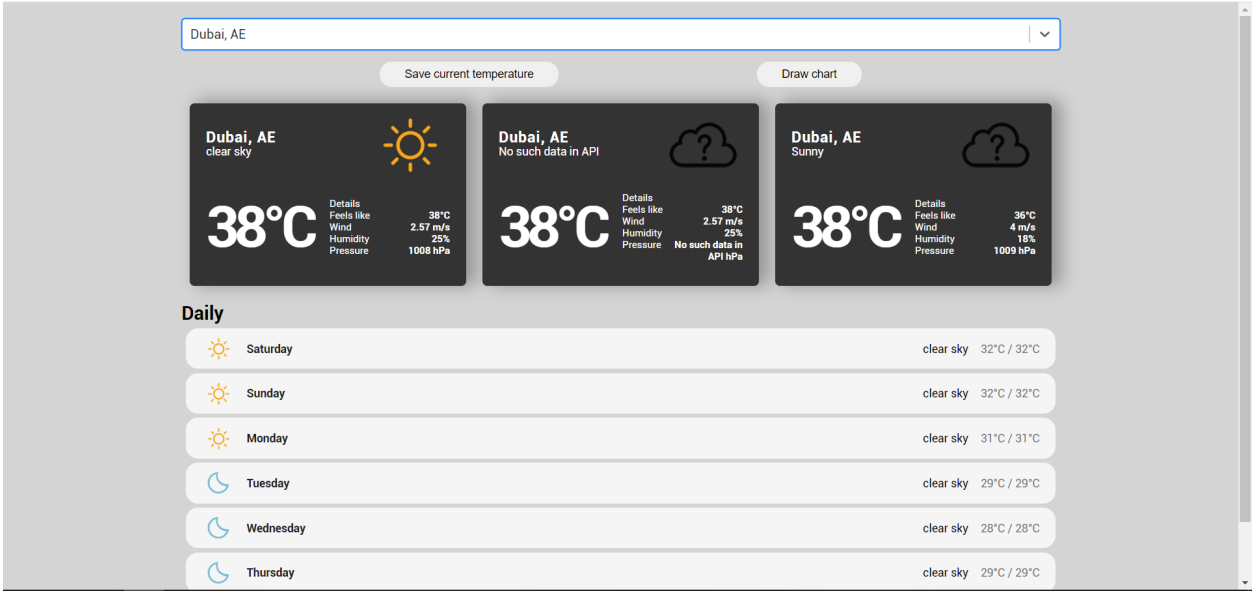

Рис. 2.10. Екран погоди і прогнозу

Екран із деталями прогнозу зображений на рис. 2.11;

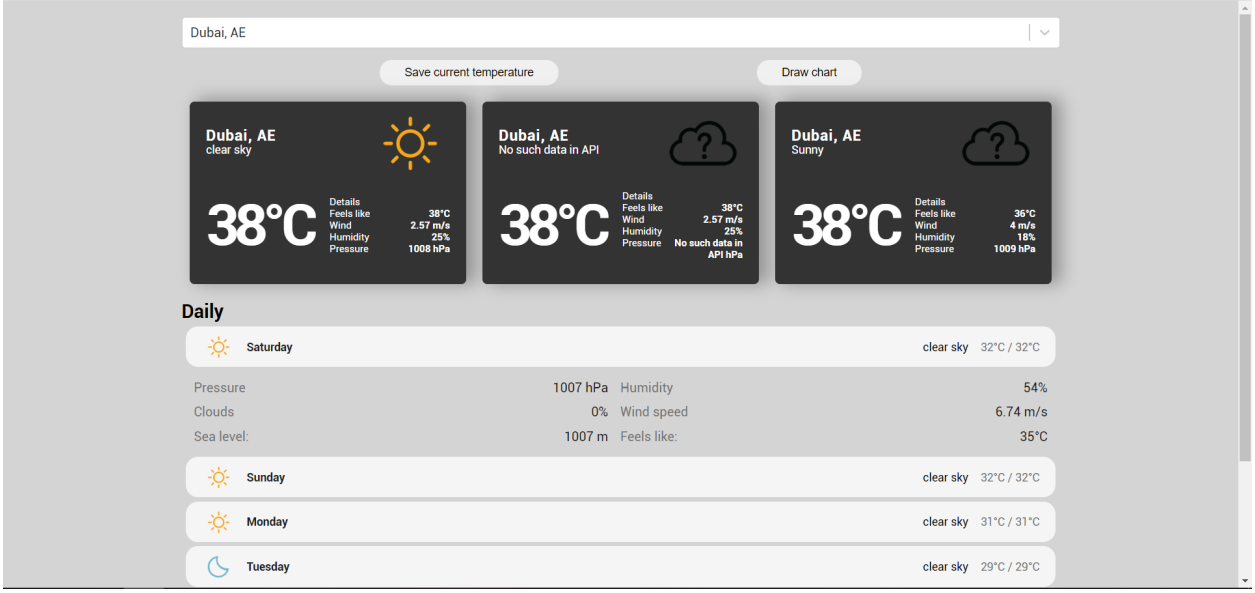

Рис. 2.11. Екран деталей прогнозу

Екран із графіком порівняння зображений на рис. 2.12;

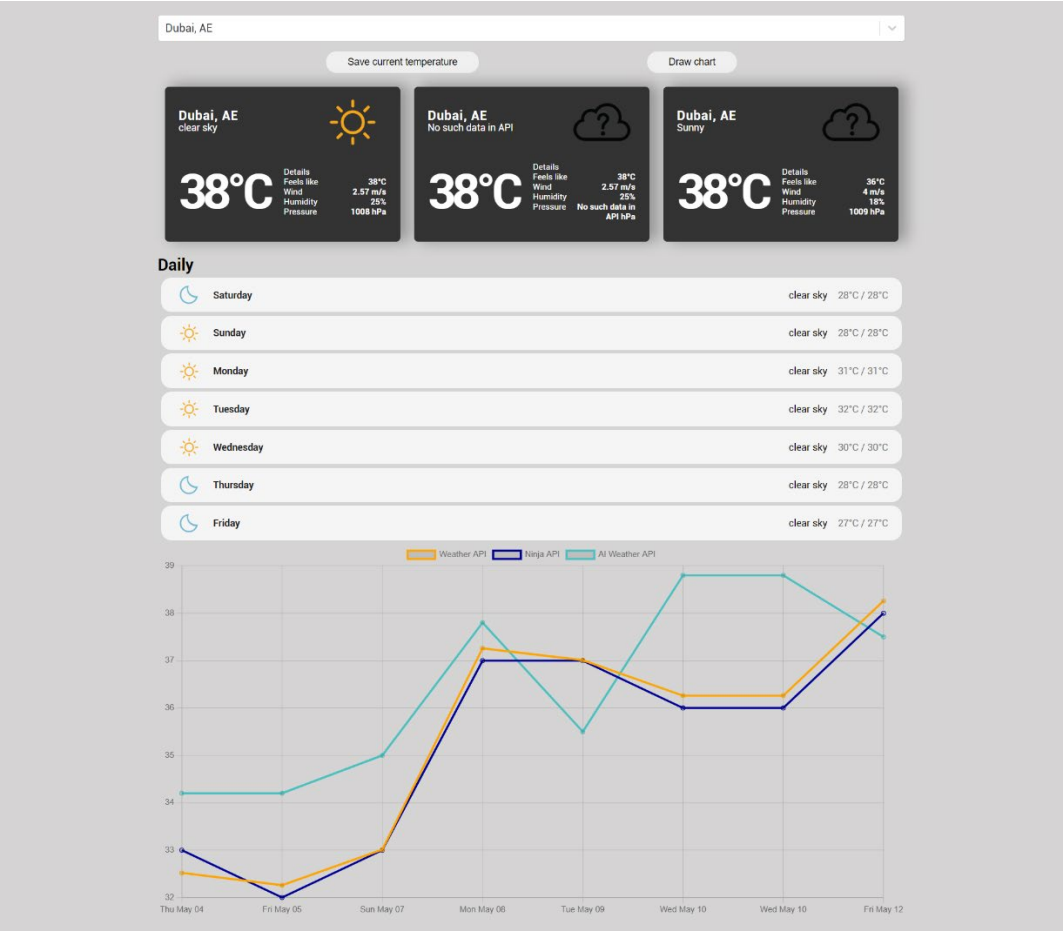

Рис. 2.12. Екран із графіком порівняння

2.14. Розробка навігації між екранами прототипу веб додатку

Оскільки при розробці використовується методика SPA (single page application), немає потреби у використанні якоїсь складної навігації. Додаток використовує один HTML файл з динамічною підгрузкою компонентів.

Висновки до розділу

Після опису аналогів, в процесі проектування, було складено задачі проекту та описані функціональні вимоги разом з вхідними та вихідними даними до веб додатку.

До описаних характеристик була підібрана мова програмування та підхід з якими найкраще можна виконати цю задачу.

У процесі проектування було спроектовано та розроблено прототип веб додатку. Для нього було створено:

– 1 екран;

– ескізи компонентів;

– розроблені компоненти;

– спроектований інтерфейс користувача;

– представлена модель експлуатації на прикладі діаграми станів;

– розроблений та представлений дизайн прототипу компонентів веб додатку.

На основі прототипу мобільного додатку було вирішено зробити наступні зміни:

– замінити елменти середнього розміру на великі та змінити їх розташування.

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1. Розробка динаміки системи

Діаграма діяльності зображена на рис. 3.1.

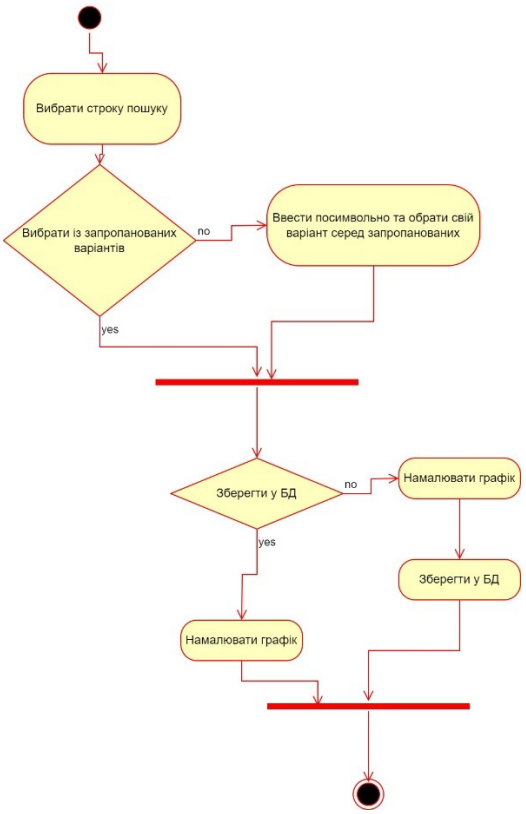

Рис. 3.1. Діаграма діяльності

Діаграма послідовності зображена на рис. 3.2.

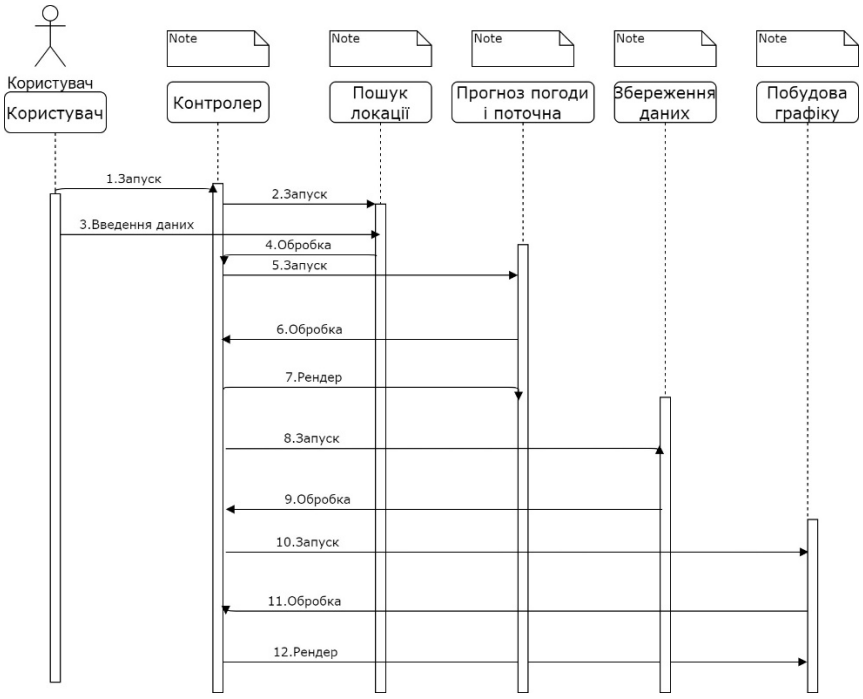

Рис. 3.2. Діаграма послідовності

3.2. Розробка екранів веб додатку

Веб додаток при 1 сторінці налічує 4 стани:

– головна сторінка при повній задянності налічує 14 комопнентів:

- 1 компонент пошуку – текстове поле;
- 3 компоненти поточної погоди;
- 1 контейнер під прогноз погоди;
- 7 компонентів для прогнозу;
- 1 компонент для відрисовки графіку;
- 1 контейнер під кнопки;
- 2 кнопки вибору збереження в БД і відрисовки графіку.

Перший стан відображення – користувач перейшов на сторінку прогнозу погоди:

- 1 компонент пошуку – текстове поле;
- 1 контейнер під кнопки;
- 2 кнопки вибору збереження в БД і відрисовки графіку.

Другий стан відображення – користувач вибрав локацію для пошуку:

- 1 компонент пошуку – текстове поле;
- 1 контейнер під кнопки;
- 2 кнопки вибору збереження в БД і відрисовки графіку;
- 3 компоненти поточної погоди;
- 1 контейнер під прогноз погоди;
- 7 компонентів для прогнозу.

Третій стан відображення – користувач зберіг дані у БД:

- 1 компонент пошуку – текстове поле;
- 1 контейнер під кнопки;
- 2 кнопки вибору збереження в БД і відрисовки графіку;
- 3 компоненти поточної погоди;

- 1 контейнер під прогноз погоди;
- 7 компонентів для прогнозу;
- тимчасове браузерне повідомлення про успішність.

Четвертий стан відображення – користувач натиснув кнопку відрисовки графіку порівняння по заданій локації:

- 1 компонент пошуку – текстове поле;
- 3 компоненти поточної погоди;
- 1 контейнер під прогноз погоди;
- 7 компонентів для прогнозу;
- 1 компонент для відрисовки графіку;
- 1 контейнер під кнопки;
- 2 кнопки вибору збереження в БД і відрисовки графіку.

3.3. Розробка інтерфейсу мобільного додатку

Інтерфейс користувача для першого стану зображений на рис. 3.3.

Рис. 3.3. Інтерфейс користувача для першого стану

Інтерфейс користувача для другого стану зображений на рис. 3.4.

Рис. 3.4. Інтерфейс користувача для другого стану

Інтерфейс користувача для третього стану зображений на рис. 3.5.

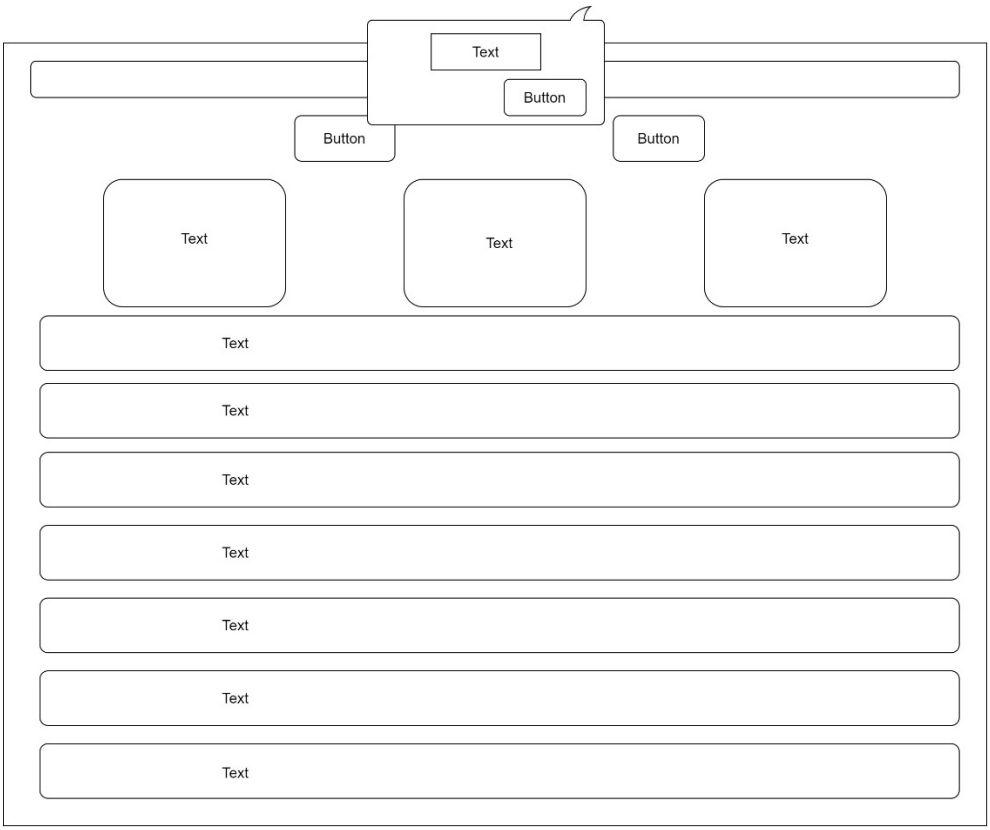

Рис. 3.5. Інтерфейс користувача для третього стану

Інтерфейс користувача для четвертого стану зображений на рис. 3.6.

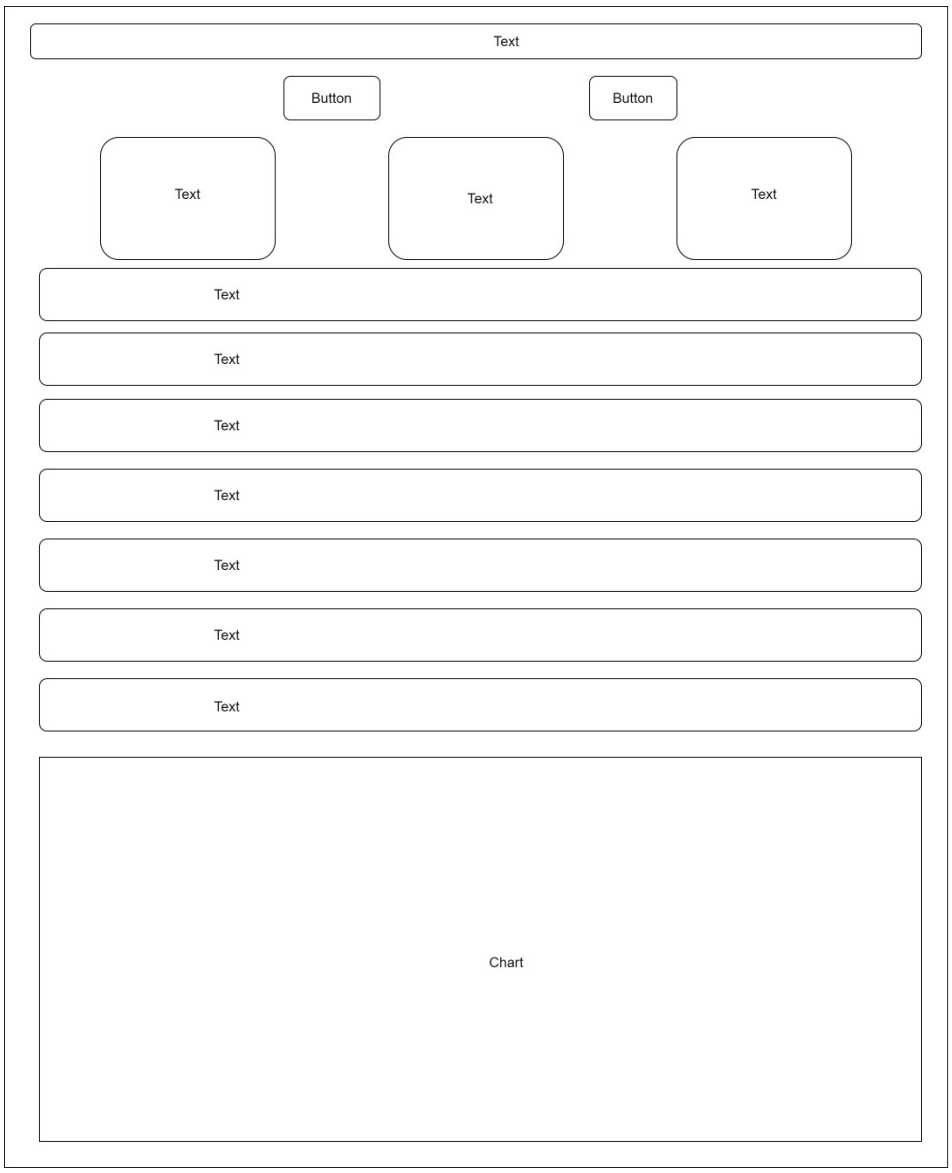

Рис. 3.6. Інтерфейс користувача для четвертого стану

3.4. Розробка навігації веб додатку

Немає необхідност розробки навігації, оскільки відбувається динамічна підгрузка компонентів на основі стану даних.

3.5. Розробка дизайну мобільного додатку

Дизайн екрану для першого стану зображений на рис. 3.7.

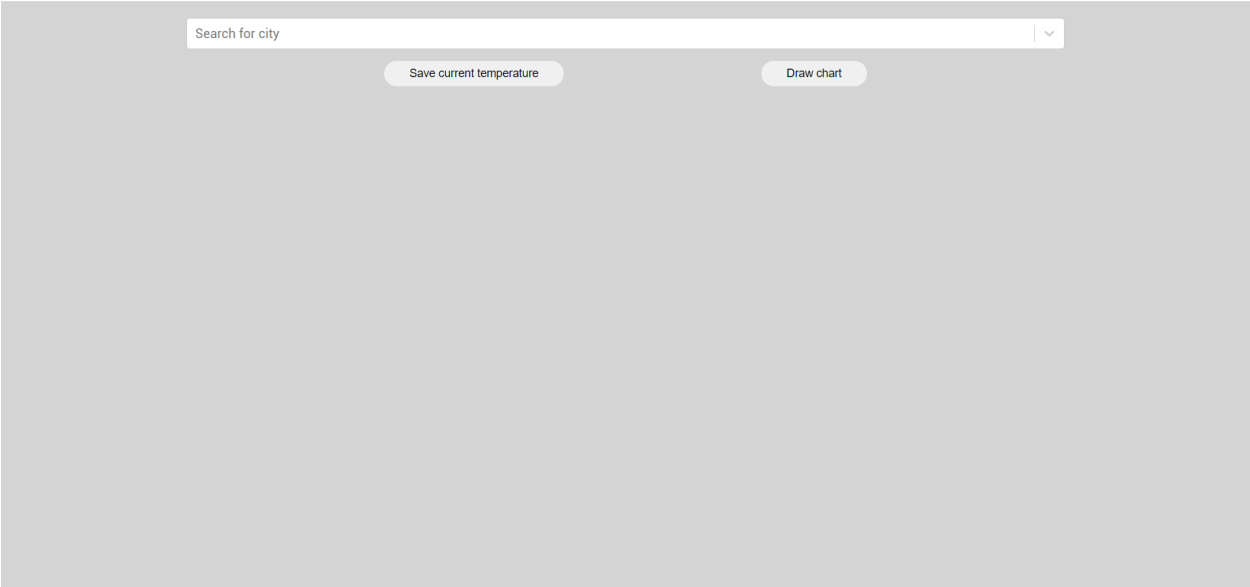

Рис. 3.7. Дизайн екрану першого стану

Дизайн екрану для другого стану зображений на рис. 3.8.

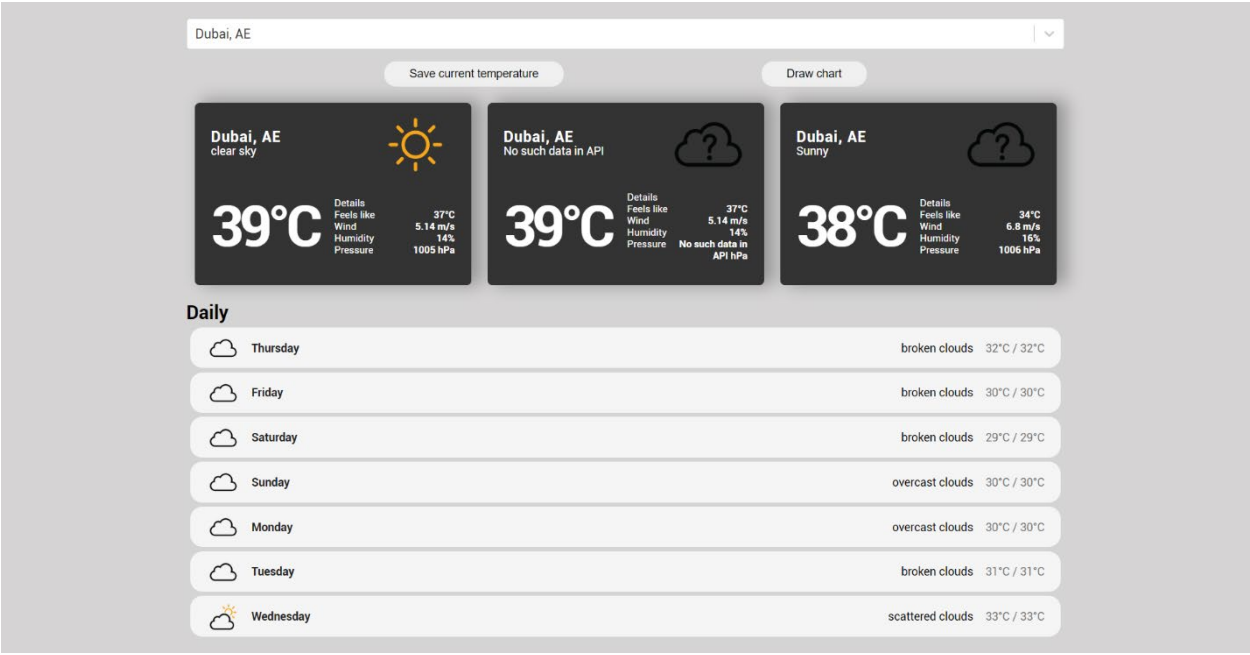

Рис. 3.8. Дизайн екрану другого стану

Дизайн екрану для третього стану зображений на рис. 3.9.

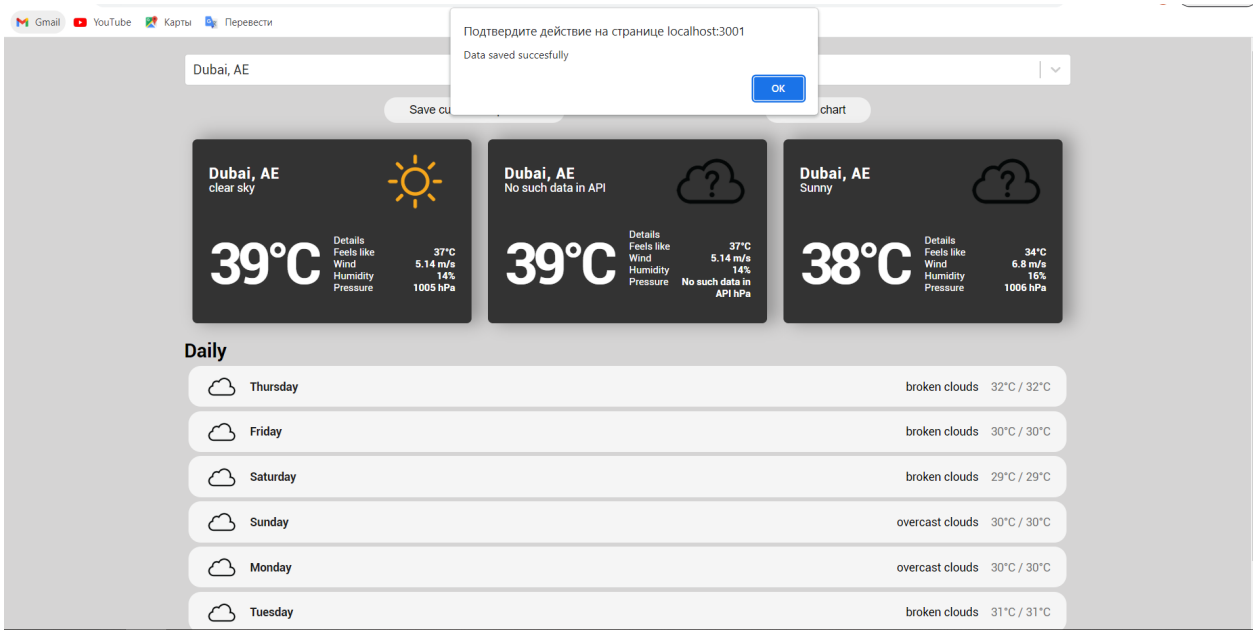

Рис. 3.9. Дизайн екрану третього стану

Інтерфейс користувача для четвертого стану зображений на рис. 3.10.

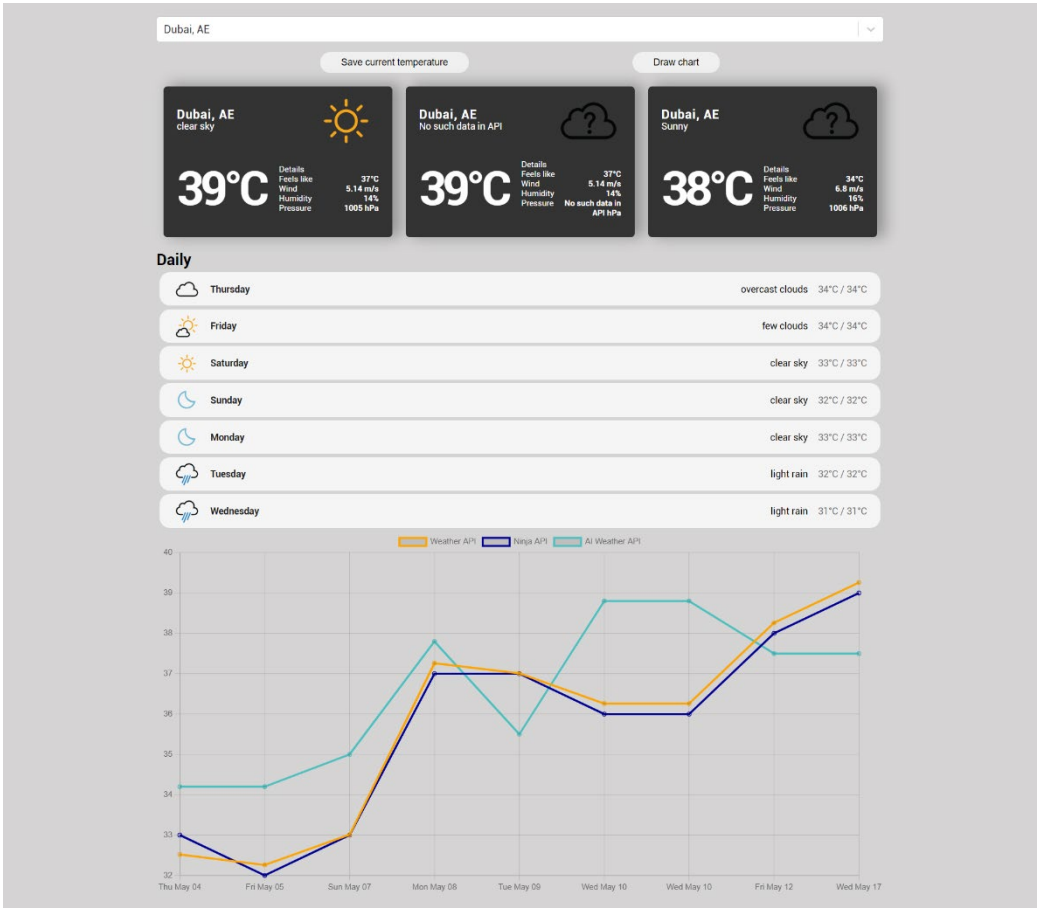

Рис. 3.10. Дизайн екрану четвертого стану

Висновки до розділу

Після опису аналогів, та проектування було розроблено динаміку роботи системи, екрани станів веб додатку та інтерфейс екранів.

У процесі розробки було розроблено робочу версію веб додатку. Для нього було створено:

– 1 екран;

– розроблені 4 компоненти;

– розроблений інтерфейс користувача;

– розроблений та представлений дизайн екранів веб додатку.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

4.1. Вибір методів тестування

Для тестування функцій було вибрано метод чорної скриньки.

Тестування чорною скринькою базується на тому що нам не відомо як влаштована система – засноване на тестуванні роботи з зовнішніми інтерфейсами.

За допомогою цього метода можно знайти помилки таких категорій:

– помилкові реалізації функцій, або повна їх відсутність;

– помилки інтерфейсу системи, яку тестуємо;

– помилкова організація структури даних, або організації доступу;

– помилкова поведінка, або повна відсутність продуктивності системи.

Таким чином можемо концентруватись на роботі програми, а не на тому, як вона це робить.

У методі тестування чорною скринькою можна виділити такі переваги та недоліки:

Переваги:

– при тестуванні не потрібно знати мову програмування та вивчати особливості реалізації;

– тестування кінцевого продукту дає змогу визначити неточності у специфікації.

Недоліки:

– тестуються не всі шляхи виконання програми;

– деякі тести можуть виявитись надлишковими.

Тестування білою скринькою базується на тому що нам відомі всі деталі реалізації системи – засноване на коді або структурі коду.

За допомогою цього метода зможемо знайти помилки таких категорій - помилкова організації внутрішньої структури компонента або структури.

Таким чином, бачивши всі вхідні дані та внутрішню організацію можемо більш детально протестувати систему.

У методі тестування білою скринькою можна виділити такі переваги та недоліки:

Переваги:

– почати тестувати систему можна все на ранніх етапах розробки;

– більша кількість шляхів виконання програми може бути розглянута.

Недоліки – для тестування методом білої скриньки необхідно дуже детально знати технологію розробки.

4.2. Тестування

Функція для тестування – handleOnClickSave, hanldeOnClickDraw;

Код функції:

```
const handleOnClickSave = async (e) => {
   e.preventDefault();
   let result = await fetch(
   'http://localhost:3000/save-temp', {
      method: "post",
      body: JSON.stringify({ currentWeather, currentWeatherNinjaApi, currentWeatherAIApi }),
      headers: {
         'Content-Type': 'application/json'
      }
   })
   result = await result.json();
   console.warn(result);
   if (result) {
      alert("Data saved succesfully");
   }
 }

   const handleOnClickDraw = (e) => {
   e.preventDefault();
   const temperatureFetch = fetch( 'http://localhost:3000/draw-chart?' + new URLSearchParams({city:
currentWeather ? currentWeather.city : null})
   )
   .then((response)=>{
     return response.json()
   })
   .then((data)=>{
```

```
    if(data){
      alert("Data have got from server");
      console.log("Data have got from server");
      console.log(data);
      setTemperatureData(data);
    }
    else
      alert('Something went wrong!');
  })
  .catch((err)=>console.log(err));
}
```

Тести для функції handleOnClickSave:

Тест 1:

- вибрана локація – Дубай.

Вхідні дані:

- currentWeather: {
  city: ‘Dubai, AE’,
  description: ‘clear sky’,
  icon: ‘01d’,
  temperature: 36.01,
  feels_like: 37.85,
  wind_speed: 7.2,
  humidity: 36,
  pressure: 1007
  };
- currentWeatherNinjaApi: {
  city: ‘Dubai, AE’,
  description: ‘No such data in Api’,
  icon: ‘uknown’,
  temperature: 36.01,
  feels_like: 37.85,
  wind_speed: 7.2,
  humidity: 36,
  pressure: 1007
  };
- currentWeatherApi: {
  city: ‘Dubai, AE’,
  description: ‘Overcast’,
  icon: ‘01d’,
  temperature: 36.01,

feels_like: 37.85,
wind_speed: 7.2,
humidity: 36,
pressure: 1007
}.

Вихідні дані:

– сформований об’єкт, який запишеться у БД
{
_id: ObjectID(‘646612be5906b211df0f408a’),
tempApiOne: 36.26,
tempApiTwo: 36,
tempApiThree: 40.8,
city: "Dubai, AE",
date: 2023-05-18T11:57:50.052+00:00,
__v: 0
};
– повідомлення у браузері “Data saved succesfully”.

Тест 2:

– жодну локацію не обрано.

Вхідні дані:

– currentWeather = null;
– currentWeatherNinjaApi = null;
– currentWeatherApi = null.

Вихідні дані:

– консольне повідомлення про помилку “Type error: cannot read propertie of undefined”

Тести для функції handleOnClickDraw:

Тест 1:

– обрана локація Дубай і об’єкт з таким містом присутній у БД.

Вхідні дані:

– currentWeather.city = ‘Dubai, AE’.

Вихідні дані:

– графік із отриманими даними із БД по заданій локації

[{

_id: ObjectID(‘646612be5906b211df0f408a’),

tempApiOne: 36.26,

tempApiTwo: 36,

tempApiThree: 40.8,

city: "Dubai, AE",

date: 2023-05-18T11:57:50.052+00:00,

__v: 0

},

{

_id: ObjectID(‘646612be5906b211df0f408a’),

tempApiOne: 37.26,

tempApiTwo: 37,

tempApiThree: 41.8,

city: "Dubai, AE",

date: 2023-05-20T11:57:50.052+00:00,

__v: 0

}];

Тест 2:

– обрано локацію ‘Kiev’, про яку не міститься записів у БД.

Вхідні дані:

– currentWeather.city = ‘Kiev’.

Вихідні дані:

– пустий масив з серверу;
– порожній графік.

Тест 3:

– не обрано локацію.

Вхідні дані:

– currentWeather = null.

Вихідні дані:

– пустий масив з серверу;
– порожній графік.

Тестування чорною скринькою:

Метод еквівалентних розбиттів – функція приймає 12 вхідних параметрів, вони використовуються в умовах розгалуження, значення параметрів перевіряються безпосередньо в функції.

Таблиця 4.1 – класів еквівалентності для функції handleOnClickSave

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Object currentWeather, currentWeatherNinjaApi, currentWeatherApi (значеннння з різних погодних сервісів)	Об’єкти типу { city: ‘Dubai, AE’, description: ‘clear sky’, icon: ‘01d’, temperature: 36.01, feels_like: 37.85, wind_speed: 7.2, humidity: 36, pressure: 1007} (1)	Один з об’єктів приймає значення null (3)
	Всі об’єкти мають значення null (2)	

Таблиця 4.2 – класів еквівалентності для функції handleOnClickDraw

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Object currentWeather	Значення атрибуту currentWeather.city типу string, наприклад ‘Dubai, AE’, що міститься як значення поля в БД (1)	currentWeather = null (3)
	Значення атрибуту currentWeather.city типу string, наприклад ‘Ivano-Frankivs’, що відсутнє у БД (2)	

Таблиця 4.3 - тести для методів еквівалентних розбиттів для функції handleOnClickSave

Класи, що покриваються	Тест	Вхід	Вихід

1	1	currentWeather: {
		city: ‘Dubai, AE’,
		description: ‘clear sky’,
		icon: ‘01d’,
		temperature: 36.01,
		feels_like: 37.85,
		wind_speed: 7.2,
		humidity: 36,
		pressure: 1007
		};
		currentWeatherNin-jaApi: {
		city: ‘Dubai, AE’,
		description: ‘No such data in Api’,
		icon: ‘uknown’,
		temperature: 36.01,
		feels_like: 37.85,
		wind_speed: 7.2,
		humidity: 36,
		pressure: 1007
		};
		currentWeatherApi: {
		city: ‘Dubai, AE’,
			{
			_id: ObjectID (‘646612be5906b211df0f408a’),
			tempApiOne: 36.26,
			tempApiTwo: 36,
			tempApiThree: 40.8,
			city: "Dubai, AE",
			date: 2023-05-18T11:57:50.052+00:00,
			__v: 0
			};
			повідомлення у браузері “Data saved succesfully”.

Закінчення таблиці 4.3

		description: ‘Over-cast’, icon: ‘01d’, temperature: 36.01, feels_like: 37.85, wind_speed: 7.2, humidity: 36, pressure: 1007 }	
2	2	currentWeather = null, currentWeatherNin-jaApi = null, currentWeatherApi = null	консольне повідомлення про помилку “Type error: cannot read prop-ertie of undefined” зі сторони сервера

3	3	currentWeather: {
		city: ‘Dubai, AE’,
		description: ‘clear sky’,
		icon: ‘01d’,
		temperature: 36.01,
		feels_like: 37.85,
		wind_speed: 7.2,
		humidity: 36,
		pressure: 1007
		};
		currentWeatherNin-jaApi: null
		;
			Запис в БД, де tempApiOne або tempApiTwo, або tempApiThree = null

Закінчення таблиці 4.3

		currentWeatherApi: {
		city: ‘Dubai, AE’,
		description: ‘Over-cast’,
		icon: ‘01d’,
		temperature: 36.01,
		feels_like: 37.85,
		wind_speed: 7.2,
		humidity: 36,
		pressure: 1007
		}	

Таблиця 4.4 - тести для методів еквівалентних розбиттів для функції handleOnClickDraw

Класи, що покриваються	Тест	Вхід	Вихід
1	1	currentWeather.city = ‘Dubai, AE’	Рендер компоенти Chart із даними з об’єктів { _id: ObjectID (‘646612be5906b211df0f408a’), tempApiOne: 36.26, tempApiTwo: 36, tempApiThree: 40.8, city: "Dubai, AE", date: 2023-05-18T11:57:50.052+00:00, __v: 0

Закінчення таблиці 4.4

			}; { _id: ObjectID (‘646612be5906b211df0f408b’), tempApiOne: 37.26, tempApiTwo: 37, tempApiThree: 39.8, city: "Dubai, AE", date: 2023-05-18T11:57:50.052+00:00, __v: 0 };
2	2	currentWeather.city = ‘Kiev’	данні з серверу [], рендер компоненти Chart без даних
3	3	currentWeather = null	данні з серверу [], рендер компоненти Chart без даних

Метод граничних умов – функція не має обмежень, які можна було би перевірити методів граничних умов.

Метод припущення про помилку – припущення для функції handleOnclickSave:

– Якщо наступні параметри (currentWeather, currentWeatherNinjaApi, currentWeatherApi) будуть дорівнювати null, то ми не отримаємо ніяких статистичних даних у БД.

– Якщо один з наступних параметрів (currentWeather, currentWeatherNinjaApi, currentWeatherApi) буде дорівнювати null, то ми отримаємо не повні статистичні дані.

Метод припущення про помилку – припущення для функції handleOnclickDraw:

- Якщо погодні дані про обране місто не було збережено у БД, то графік буде пустий.
- Якщо місто не було обрано, нічого шукати у БД, то графік буде пустий.

Тести для припущень не були розроблені, оскільки ці припущення покриваються тестами з методу еквівалентних розбиттів.

Таблиця 4.5 – метод функціональних діаграм для функції handleonClickSave

Причини та наслідки	У/п	Значення	Помітки про присутність	
			1	2
Причини	П1	currentWeather != null	+	-
	П2	currentWeatherNinjaApi != null	-	-
	П3	currentWeatherWeatherApi != null	+	-
	П4	currentWeather = null	-	+
	П5	currentWeatherNinjaApi = null	+	+
	П6	currentWeatherWeatherApi = null	-	+
Наслідки	Н1	Об’єкт з даними не буде записано у БД	-	+
	Н2	Один з полів даних у бд tempApiOne, tempApiTwo, tempApiThree = null	+	-

Таблиця 4.6 – метод функціональних діаграм для функції handleOnClickDraw

Причини та наслідки	У/п	Значення	Помітки про присут-ність	
			1	2
Причини	П1	(currentWeather.city = Weather.findByKey(currentWeather.city)) == false	+	-
	П2	currentWeather == null	+	-
	П3	(currentWeather.city = Weather.findByKey(currentWeather.city)) == true	-	+
	П4	currentWeather != null	-	+
Наслідки	Н1	Пустий графік	+	-
	Н2	Графік із даними	-	+

На рис. 4.1 зображена функціональна діаграма для функції handleOn-ClickSave.

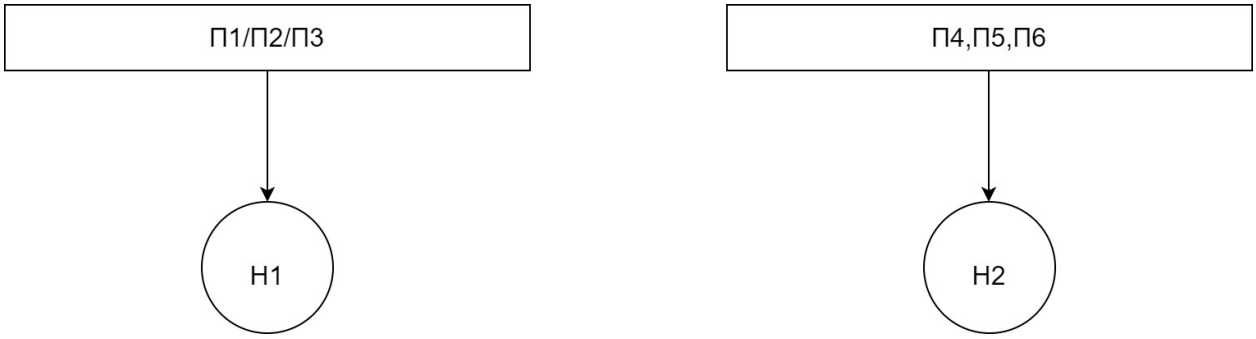

Рис. 4.1. Функціональна діаграма

На рис. 4.2 зображена функціональна діаграма для фукнції handleOn-ClickDraw.

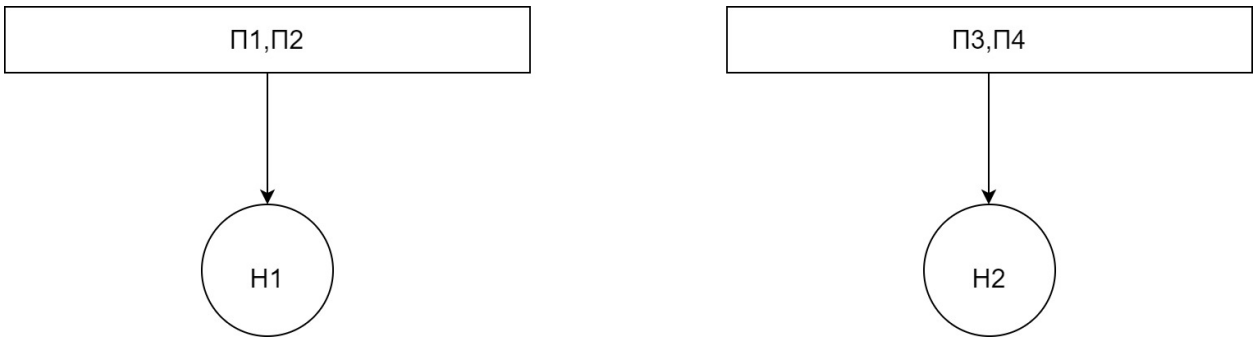

Рис. 4.2. Функціональна діаграма

Для функції handleOnClickSave:

- до першого наслідку приводять причини під номерами – П1або П2 або П3;
- до другого наслідку приводять причини під номерами – П4, П5, П6.

Для функції handleOnClickDraw:

- до першого наслідку приводять причини під номерами – П1, П2;
- до другого наслідку приводять причини під номерами – П3, П4.

Висновки до розділу

Вибрані функції було протестовано методом чорної скриньки. Під час роботи використані та описані наступні методи:

– методи чорної скриньки:

- метод еквівалентних розбиттів;
- метод припущення про помилку;
- метод функціональних діаграм.

Різноманітні своїм підходом методи дали змогу протестувати роботу програми та перевірити її на можливі помилки.

Загальні висновки

Результатом роботи є веб застосунок розроблений на мові JavaScript з використанням фреймворку React та Express, який користується трьома різними джерелами погодних даних (Weather API, API-Ninjas API, AI Weather API) і порівнює їх точність, виводячи у вигляді графіка.

На осовнові розглянутих та описаних аналогів, з оцінкою їх плюсів та мінусів, були поставлені задачі проектування та розробки проекту.

На етапі проектування були визначені вхідні та вихідні умови, функціональні вимоги, спроектовані необхідні компоненти з їх унікальним інтерфейсом та дизайном. Розробка прототипу веб додатку дозволила одразу протестувати функціональні можливості для їх покращення та переопрацювання спірних моментів.

На етапі розробки, після розробки та користування прототипом мобільного додатку було змінено:

– кількість розроблених компонентів збільшилась;

– зовнішнього вигляду екрану;

– розміщення компонентів на веб сторінці додатку;

– зовнішній вигляд дизайну компонентів.

Веб додаток був протестований за допомогою методу чорної скриньки. Різноманітність підходів методу чорної скриньки дав можливість під різнами кутами подивитись на програму та протестувати її функціонал.

Ще одним важливим висновком є необхідність розробки алгоритмів, які дозволять ефективно порівнювати прогнози погоди. У ході дослідження було розглянуто класичний підхід до порівняння, а саме статичстичний. Існує необхідність розвинення підходів порівняння: застосування статистичних методів, використання машинного навчання та інших технологій. Результати порівняння можуть бути використані для оцінки якості прогнозів та визначення

найбільш достовірних джерел інформації про погоду. Це дозволить забезпечити користувачів точнішою та надійнішою інформацією при виборі джерела прогнозів погоди.

У результаті розробки було досягнуто створення зручного, сучасного та адаптивного веб-інтерфейсу для web-застосунку "Порівняння прогнозів погоди". Веб-інтерфейс забезпечує користувачам зручний доступ до функціональності застосунку на різних пристроях та розмірах екрану. Він використовує сучасні дизайнерські рішення та ергономічні принципи, що забезпечують зручну навігацію та інтуїтивно зрозумілі елементи управління. Адаптивний дизайн дозволяє автоматично адаптувати веб-інтерфейс до різних пристроїв та екранних розмірів, забезпечуючи зручне використання застосунку незалежно від пристрою, на якому він запущений.

Список використаних джерел

1. Навчальний посібник для підготовки кваліфікаційної роботи на здобут-тя ОС Бакалавр 2.

2. Технічна документація з React [Електронний ресурс] https://ua.reactjs.org/docs/getting-started.htm

3. Warmer J, Kleppe A (1999) Мова обмежень об'єктів: точне моделювання за допомогою UML. Аддісон-Веслі Лонгман, Лондон

4. Spinellis D, Raptis K (2000) Добування компонентів: процес та його мова шаблонів. Інформаційні та програмні технології 42: 609–617

5. Quatrani T (1997) Візуальне моделювання з Rational Rose та UML. Аддісон-Веслі Лонгман, Лондон

6. Meyer B (1997) Об'єктно-орієнтована конструкція програмного забезпечення, 2-ге видання. Прентіс Холл, Енглвуд Кліпп, Нью-Джерсі

7. Кобрин С (2000) Моделювання компонентів і фреймворків за допомогою UML. Повідомлення ACM 43(10): 31–38

8. [Електронний ресурс] Mobile Vs. Desktop Usage - https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics

9. [Електронний ресурс] The benefits of using web-based applications - https://www.geeks.ltd.uk/about-us/blog/details/eQU5Ip/the-benefits-of-usingweb-based-applications

10. [Електронний ресурс] The pros and cons of building a Mobile app vs a Web app - https://designli.co/blog/the-pros-and-cons-of-building-a-mobile-app-vs-aweb-app/

11. UX – це не UI [Електронний ресурс]: cmsmagazine.ru – Режим доступу: http://www.cmsmagazine.ru/library/items/usability/ux-is-not-ui/

12. Васильєв А.М. Java. Об'єктно-орієнтоване програмування. Навчальний посібник. - СПб.: Пітер, 2011. - 400 с.

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського державного університету науки і технологій

Анатолій РАДКЕВИЧ

18.02.23

РОЗРОБКА WEB-ЗАСТОСУНКУ «ПОРІВНЯННЯ ПРОГНОЗІВ ПОГОДИ»

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850. 01289-01-ЛЗ

Представники

підприємства-розробника

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

18.02.23

Керівник розробки

Олександр ІВАНОВ

18.02.23

Виконавець

Кирило ДОРОГОКУПЛЯ

18.02.23

Норм-контролер

Світлана ВОЛКОВА

18.02.23

2023

ЗАТВЕРДЖЕНО

44165850. 01289-01-ЛЗ

РОЗРОБКА WEB-ЗАСТОСУНКУ «ПОРІВНЯННЯ ПРОГНОЗІВ ПОГОДИ»

Технічне завдання

44165850.01289-01

Листів 14

2023

ЗМІСТ

1 ВВЕДЕННЯ

Веб застосунок «Порівняння прогнозів погоди» має на меті надати користувачам зручний спосіб порівняння прогнозів погоди з різних джерел для забезпечення точної та надійної інформації про погодні умови.

В сучасному світі, коли погода впливає на багато аспектів нашого повсякденного життя, важливо мати доступ до якісної інформації про погоду. Однак, існує багато джерел та сервісів, які надають прогнози погоди, і вони можуть відрізнятись за точністю та достовірністю інформації. Це може створювати певні труднощі для користувачів, які прагнуть знайти найбільш достовірний та зручний джерело прогнозів погоди.

Веб застосунок "Порівняння прогнозів погоди" вирішує цю проблему, надаючи можливість порівнювати прогнози погоди з трьох найпопулярніших джерел в одному зручному місці. Застосунок буде дозволяти користувачам вводити місцезнаходження або вибирати існуючі місця. Після цього, застосунок отримуватиме прогнози погоди із запропанованих джерел і відображатиме їх в зручному форматі, що дозволить користувачам переглянути графік порівняння та визначити найбільш достовірний прогноз.

Програмний продукт є безкоштовний, що робить його більш цікавим на фоні інших аналогів, але варто розуміти, що це рекламний продукт.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 07.12.22 №1209 ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи - “РОЗРОБКА WEB-ЗАСТОСУНКУ ПОРІВНЯННЯ ПРОГНОЗІВ ПОГОДИ”. Керівник - доцент Іванов О. П.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт завантажує прогноз погоди і поточні погодні умови із трьох метео ресурсів, основуючись на вибраній локації.

Експлуатаційне призначення – за допомогою програмного продукту виконується аналіз точності даних метеостанцій, для полегшення повсякденного життя людини. Програмний продукт може використовуватись як у великих метеостанціях, так і буденному житті будь-якої людини, що дає змогу більше детально розглянути метеостанції і дані, які поширюються у мережу.

4 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Програмний продукт повинний забезпечувати можливість вибирати локацію для перегляду погоди і прогнозу погоди.

Локація може бути обрана:

- серед запропанованих міст;

після набирання символів у пошукову строку із подальшим вибором запропанованих варіантів.

Програмний продукт повинний забезпечувати можливість збереження погодних даних у базу даних із відповідним повідомленням.

Програмний продукт повинен забезпечувати можливість відображати збережені дані для обраної локації у вигляді графіку.

Програмний продукт має відображати прогноз погоди на тиждень з поточного дня із можливістю перегляду додаткових даних.

Програмний продукт має генерувати відповіді для збереження і для отримання даних.

Вимоги до вхідних даних:

Веб-додаток передбачає послідовне виконання дій, які не дадуть змоги перейти на наступний етап виконання програми. У разі виключення якогось етапу, на екрані нічого не зміниться.

Вхідними даними є:

- локація для пошуку (символи):
  - буквенні символи;
- погодні дані:
  - температура трьох погодних сервісів (число);
  - локація (строка символів);
  - дата збереження.

Без вибору локації не отримаємо погодних даних і не зможемо намалювати графік порівнянь.

Вимоги до вихідних даних:

Вихідними даними є:

– погодні дані з 3 різних станцій:
  – температура;
  – температура як відчувається;
  – вологість;
  – тиск;
  – опис погоди;
  – номер малюнку;
  – запропановані локації стандартні або за першими символами.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

- при запиті даних з серверу показ відповідних повідомлень про стан роботи програми та результати виконання операцій;
- наявність архівної копії тексту програми на зовнішньому носії;
- наявність резервної копії бази даних на зовнішньому носії.

4.3 Вимоги експлуатації

Програмний продукт повинен використовуватись у приміщеннях які відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.13-99 (див. табл. 1).

Таблиця 1. Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.01-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		Оптимальна	Оптимальна	Оптимальна
Холодна	легка-1-а	22-24	40-60	0,1
	легка-1-б	21-23	40-60	0,1
Тепла	легка-1-а	23-25	40-60	0,1
	легка-1-б	22-24	40-60	0,2

Працювати з програмою може людина, що має навички роботи з персональним комп’ютером, роботоюв браузері та ознайомлена з керівництвом користувача програмного продукту.

4.4 Вимоги до складу та параметрів технічних засобів

Продукт, що розробляється повинен використовуватись на будь-якому пристрої, де є браузер, тому формуються наступні мінімальні вимоги до пристроїв, що мають наступні характеристики:

- роздільна здатність дисплею – HD (1280x720);
- оперативна пам’ять – 2 ГБ;
- вбудована пам’ять – 4 ГБ;
- операційна система – Wibdows, macOS X, Chrome OS, Linux;
- частота процесора – 1.6 ГГц.

4.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем починаючи від мінімальних версій для опреаційних систем:

- Windows 7 і більш пізніші версій;
- macOS X 10.9 і пізніші версії;
- Chrome OS всі моделі від 2013 року;
- Linux – робота залежить від дистрибутивів, драйверів і среди робочого стола;

Оскільки програмний продукт являє собою веб-додатком, необхідно мати у наявності відповідно до операційної системи браузер:

- Chrome мінімальна версія 85;
- Firefox мінімальна версія 81;
- Safari мінімальна версія 14;
- Opera мінімальна версія 64;
- Microsoft Explorer мінімальна версія 11.

4.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

На упаковці повинно бути вказана назва продукту, номер версії, мінімальні системні вимоги.

На зворотній стороні упаковки вказується розробник та його юридична адреса.

Програмний продукт **WEB-ЗАСТОСУНОК "ПОРІВНЯНН ПРОГНОЗІВ ПОГОДИ"** **Мінімальні системні вимоги:** -- роздільна здатність дисплею – HD (1280x720); – оперативна пам'ять – 2 ГБ; – вбудована пам'ять – 4 ГБ; – операційна система – Wibdows, macOS X, Chrome OS, Linux; – частота процесора – 1.6 ГГц.	Програмний продукт **WEB-ЗАСТОСУНОК "ПОРІВНЯНН ПРОГНОЗІВ ПОГОДИ"** **Розробник: Дорогокупля К.О.** **Кафедра "КІТ", УДУНТ** **м.Дніпро, вул.Лазаряна 2** **2023**

4.7 Вимоги до транспортування та зберігання

Транспортування повинне забезпечувати збереження програмного продукту його цілісність і запобігання несанкціонованого доступу до нього. Програмний виріб міститься на віддаленному фізичному носії, та використовується через знаходження у браузері.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

– текст програми.

Вся документація програмного додатку повинна задовольняти вимоги до програмної документації.

.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиці 1. – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 -18.02.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23

7 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Іванов О. П.

Прийом здійснюється комісією у складі:

- Горячкін В. М. (керівник підрозділу);
- Іванов О. П. (керівник розробки).

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с

Додаток Б

Текст програми

Файл App.js

```
import { useEffect, useState } from 'react';
import './App.css';
import {
  WEATHER_API_URL,
  WEATHER_API_KEY,
  WEATHER_API_NINJA_URL,
  weatherNinjaApiOptions,
  WEATHER_API_AI_URL,
  weatherAIApiOptions } from './api';
import Search from './components/search/search';
import CurrentWeather from './components/current-weather/current-weather';
import Forecast from './components/forecast/forecast';
import LineChart from './components/chart/line-chart';


const decodeCurrentWeatherResponse = (response, city) => {
 return {
  city: city.label,
  description: response.weather[0].description,
  icon: response.weather[0].icon,
  temperature: response.main.temp,
  feels_like: response.main.feels_like,
  wind_speed: response.wind.speed,
  humidity: response.main.humidity,
  pressure: response.main.pressure
 }
}

/*const convertResponseiconToDefined = (icon) => {
  switch (icon) {
    case 'mostly_cloud':
      return '03d';
      break;
    case 'overcast':
      return '04d';
      break;
    default:
      break;
  }
}*/

const decodeCurrentWeatherNinjaResponse = (response, city) => {
  return {
    city: city.label,
    description: "No such data in API",
    icon: "unknown",
    temperature: response.temp,
    feels_like: response.feels_like,
    wind_speed: response.wind_speed,
    humidity: response.humidity,
    pressure: "No such data in API"
  }
}

const decodeCurrentWeatherAIResponse = (response, city) => {
  return {
    city: city.label,
    description: response.current.summary,
    icon: "unknown",
    temperature: response.current.temperature,
    feels_like: response.current.feels_like,
    wind_speed: response.current.wind.speed,
    humidity: response.current.humidity,
    pressure: response.current.pressure
  }
}

function App() {

  const [currentWeather, setCurrentWeather] = useState(null);
```

```
  const [forecast, setForecast] = useState(null);

  const [temperatureData, setTemperatureData] = useState(null);

  const [currentWeatherNinjaApi, setCurrentWeatherNinjaApi] = useState(null);
  const [currentWeatherAIApi, setCurrentWeatherAIApi] = useState(null);

  const handleOnSearchChange = (searchData) => {
    setTemperatureData(null);
    const [lat, lon]  = searchData.value.split(" ");

    const currentWeatherFetch =
fetch(`${WEATHER_API_URL}/weather?lat=${lat}&lon=${lon}&appid=${WEATHER_API_KEY}&units=metric`);
    const forecastFetch = fetch(`${WEATHER_API_URL}/forecast?lat=${lat}&lon=${lon}&appid=${WEATHER_API_KEY}&units=metric`);

    const currentWeatherNinjaApiFetch = fetch(`${WEATHER_API_NINJA_URL}/weather?lat=${lat}&lon=${lon}`, weatherNinjaApiOptions);

    const currentWeatherAIApi =
fetch(`${WEATHER_API_AI_URL}/current?lat=${lat}&lon=${lon}&timezone=auto&language=en&units=metric`, weatherAIApiOptions);

    Promise.all([
      currentWeatherFetch,
      forecastFetch,
      currentWeatherNinjaApiFetch,
      currentWeatherAIApi])
      .then(async (response) => {
        const weatherResponse = await response[0].json();
        const forecastResponse = await response[1].json();

        const weatherResponseNinja = await response[2].json();
        const weatherResponseAI = await response[3].json();

        console.log("New API response");
        console.log(weatherResponseAI);

        setCurrentWeather(decodeCurrentWeatherResponse(weatherResponse, searchData));
        setForecast({ city: searchData.label, ...forecastResponse});

        setCurrentWeatherNinjaApi(decodeCurrentWeatherNinjaResponse(weatherResponseNinja, searchData));

        setCurrentWeatherAIApi(decodeCurrentWeatherAIResponse(weatherResponseAI, searchData));
      })
      .catch((err)=> console.log(err));

  }

 const handleOnClickSave = async (e) => {
    e.preventDefault();
    let result = await fetch(
    'http://localhost:3000/save-temp', {
        method: "post",
        body: JSON.stringify({ currentWeather, currentWeatherNinjaApi, currentWeatherAIApi }),
        headers: {
            'Content-Type': 'application/json'
        }
    })
    result = await result.json();
    console.warn(result);
    if (result) {
        alert("Data saved succesfully");
    }
  }

  useEffect(()=>{
    console.log("useEffect");
    console.log(temperatureData);
  }, [temperatureData])

  const handleOnClickDraw = (e) => {
    e.preventDefault();
    const temperatureFetch = fetch( 'http://localhost:3000/draw-chart?' + new URLSearchParams({city: currentWeather ? currentWeather.city :
null}))
    )
    .then((response)=>{
      return response.json()
    })
    .then((data)=>{
      if(data){
        alert("Data have got from server");
```

```
      console.log("Data have got from server");
      console.log(data);
      setTemperatureData(data);
    }
    else
      alert('Something went wrong!');
  })
  .catch((err)=>console.log(err));
 }

 return (
  <div className="container">
   < Search onSearchChange={handleOnSearchChange} />
   <div className='container__button-group'>
    <button onClick={handleOnClickSave}>
     Save current temperature
    </button>
    <button onClick={handleOnClickDraw}>
     Draw chart
    </button>
   </div>
   <div className="container__daily-weather">
    {currentWeather && < CurrentWeather data={currentWeather} />}
    {currentWeatherNinjaApi && < CurrentWeather data={currentWeatherNinjaApi} />}
    {currentWeatherAIApi && < CurrentWeather data={currentWeatherAIApi} />}
   </div>

   {forecast && < Forecast data={forecast} />}

   {temperatureData && < LineChart chartData={temperatureData} />}

  </div>
 );
}

export default App;
```

Файл api.js

```
export const geoApiOptions = {
        method: 'GET',
        headers: {
                'X-RapidAPI-Key': 'd383e60ceamsh61085673c8cd4c4p150888jsn18de1618c455',
                'X-RapidAPI-Host': 'wft-geo-db.p.rapidapi.com'
        }
};

export const GEO_API_URL = "https://wft-geo-db.p.rapidapi.com/v1/geo";

export const WEATHER_API_URL = 'https://api.openweathermap.org/data/2.5';
export const WEATHER_API_KEY = '095e1f4448516f03cc412833003f7d97';

export const weatherNinjaApiOptions = {
        method: 'GET',
        headers: {
                'X-RapidAPI-Key': 'd383e60ceamsh61085673c8cd4c4p150888jsn18de1618c455',
                'X-RapidAPI-Host': 'weather-by-api-ninjas.p.rapidapi.com'
        }
};

export const WEATHER_API_NINJA_URL = 'https://weather-by-api-ninjas.p.rapidapi.com/v1';
export const WEATHER_API_NINJA_KEY = 'd383e60ceamsh61085673c8cd4c4p150888jsn18de1618c455';



export const weatherAIApiOptions = {
        method: 'GET',
        headers: {
                'X-RapidAPI-Key': 'd383e60ceamsh61085673c8cd4c4p150888jsn18de1618c455',
                'X-RapidAPI-Host': 'ai-weather-by-meteosource.p.rapidapi.com'
        }
};

export const WEATHER_API_AI_URL = 'https://ai-weather-by-meteosource.p.rapidapi.com';
export const WEATHER_API_AI_KEY = 'd383e60ceamsh61085673c8cd4c4p150888jsn18de1618c455';
```

Файл search.js

```
import { useState } from "react";
```

```
import { AsyncPaginate } from "react-select-async-paginate"
import { GEO_API_URL, geoApiOptions } from "../../api";

const Search = ({onSearchChange}) => {

  const [search, setSearch] = useState(null);

  const loadOptions = (inputValue) => {
    return fetch(`${GEO_API_URL}/cities?minPopulation=100000&namePrefix=${inputValue}`,
    geoApiOptions
    )
    .then(response => response.json())
    .then(response => {
      return {
        options: response.data.map((city)=>{
          return {
            value: `${city.latitude} ${city.longitude}`,
            label: `${city.name}, ${city.countryCode}`,
          }
        })
      }
    })
    .catch(err => console.error(err));
  }

  const handleOnChange = (searchData) => {
    setSearch(searchData);
    onSearchChange(searchData);
  }

  return (
    <AsyncPaginate
      placeholder="Search for city"
      debounceTimeout={600}
      value={search}
      onChange={handleOnChange}
      loadOptions={loadOptions}
    />
  );
}

export default Search;
```

Файл current-weather.js

```
import "./current-weather.css"

const CurrentWeather = ( { data } ) => {
  return (
    <div className="weather">
      <div className="top">
        <div >
          <p className="city">{data.city}</p>
          <p className="weather-description">{data.description}</p>
        </div>
        <img alt="weather" className="weather-icon" src={`icons/${data.icon}.png`} />
      </div>

      <div className="bottom">
        <p className="temperature">{Math.round(data.temperature)}°C</p>
        <div className="details">
          <div className="parameter-row">
            <span className="parameter-label">Details</span>
          </div>

          <div className="parameter-row">
            <span className="parameter-label">Feels like</span>
            <span className="parameter-value">{Math.round(data.feels_like)}°C</span>
          </div>

          <div className="parameter-row">
            <span className="parameter-label">Wind</span>
            <span className="parameter-value">{data.wind_speed} m/s</span>
          </div>

          <div className="parameter-row">
            <span className="parameter-label">Humidity</span>
            <span className="parameter-value">{data.humidity}%</span>
          </div>
```

```
        <div className="parameter-row">
          <span className="parameter-label">Pressure</span>
          <span className="parameter-value">{data.pressure} hPa</span>
        </div>
      </div>
    </div>
  </div>
 );
}

export default CurrentWeather;
```

Файл forecast.js

```
import {
 Accordion,
 AccordionItem,
 AccordionItemHeading,
 AccordionItemPanel,
 AccordionItemButton
} from 'react-accessible-accordion';
import './forecast.css'

const WEEK_DAYS = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"];

const Forecast = ({data}) => {

 const dayInAWeek = new Date().getDay();
 const forecastDays = WEEK_DAYS.slice(dayInAWeek, WEEK_DAYS.length).concat(WEEK_DAYS.slice(0, dayInAWeek));

 return(
 <>
  <label className="title">Daily</label>
  <Accordion allowZeroExpanded>
    {data.list.splice(0, 7).map((item, idx) => (
     <AccordionItem key={idx}>
       <AccordionItemHeading>
         <AccordionItemButton>
           <div className='daily-item'>
             <img alt='weather' className='icon-small' src={`icons/${item.weather[0].icon}.png`}/>
             <label className='day'>{forecastDays[idx]}</label>
             <label className='description'>{item.weather[0].description}</label>
             <label className='min-max'>{Math.round(item.main.temp_min)}°C
             / {Math.round(item.main.temp_max)}°C</label>
           </div>
         </AccordionItemButton>
       </AccordionItemHeading>
       <AccordionItemPanel>
         <div className='daily-details-grid'>
           <div className='daily-details-grid-item'>
             <label>Pressure</label>
             <label>{item.main.pressure} hPa</label>
           </div>

           <div className='daily-details-grid-item'>
             <label>Humidity</label>
             <label>{item.main.humidity}%</label>
           </div>

           <div className='daily-details-grid-item'>
             <label>Clouds</label>
             <label>{item.clouds.all}%</label>
           </div>

           <div className='daily-details-grid-item'>
             <label>Wind speed</label>
             <label>{item.wind.speed} m/s</label>
           </div>

           <div className='daily-details-grid-item'>
             <label>Sea level: </label>
             <label>{item.main.sea_level} m</label>
           </div>

           <div className='daily-details-grid-item'>
             <label>Feels like: </label>
             <label>{Math.round(item.main.feels_like)}°C</label>
           </div>
```

```
          </div>
        </AccordionItemPanel>
      </AccordionItem>
    ))}
  </Accordion>
 </>
 );
}

export default Forecast;
```

Файл line-chart.js

```
import React from "react";
import { Line } from "react-chartjs-2";
import { Chart as ChartJS} from 'chart.js/auto'

function LineChart({chartData}) {
 return < Line data={chartData} />
}

export default LineChart;
```

Файл server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const Weather = require('./weather.js');
const cors = require('cors');

const PORT = 3000;

const app = express();

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cors());

app.get("/", (req, res) => {
 res.send("Hello world!");
});

const db = 'mongodb+srv://kirildora02:Pass123@cluster0.vpqfqrp.mongodb.net/?retryWrites=true&w=majority';

mongoose
.connect(db, { useNewUrlParser: true, useUnifiedTopology: true })
.then((res) => console.log('Connected to DB'))
.catch((error) => console.log(error));

app.post('/save-temp', (req, res) => {
 const { tempApiOne, tempApiTwo, tempApiThree, city } = {
   tempApiOne: req.body.currentWeather.temperature,
   tempApiTwo: req.body.currentWeatherNinjaApi.temperature,
   tempApiThree: req.body.currentWeatherAIApi.temperature,
   city: req.body.currentWeather.city
 };
 const weather = new Weather({ tempApiOne, tempApiTwo, tempApiThree, city});
 weather
   .save()
   .then((result) => res.send(result))
   .catch((error) => {
     console.log(error);
   })
});

app.get(`/draw-chart`, (req, res) => {
 let city = req.query.city;
 console.log("location have got from ui");
 console.log(city);

 if(city) {
   Weather
   .find({'city': city})
   .then((data) => {
     console.log("Data we got from db");
     console.log(data);
     const dataToSend = {
       labels: data.map((item) => item.date.toString().slice(0,10)),
```

```
      datasets: [
        {
          label: "Weather API",
          data: data.map((item) => item.tempApiOne),
          borderColor: ['rgba(255, 165, 0, 1)']
        },
        {
          label: "Ninja API",
          data: data.map((item) => item.tempApiTwo),
          borderColor: ['rgba(0, 0, 148, 1)']
        },
        {
          label: "AI Weather API",
          data: data.map((item) => item.tempApiThree),
          borderColor: ['rgba(75, 192, 192, 1)']
        }
      ]
    }
    console.log("Data we converted to send to ui");
    console.log(dataToSend);
    res.send(dataToSend);
  })
  .catch((error) => {
    console.log(error);
  });
 }
 else
  console.log("There is no such data about chosen city");

});

app.listen(PORT, ()=> {console.log("App listening on port 3000");});
```

Файл weather.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const weatherSchema = new Schema({
 tempApiOne: {
  type: Number,
  required: true,
 },
 tempApiTwo: {
  type: Number,
  required: true,
 },
 tempApiThree: {
  type: Number,
  required: true,
 },
 city: {
  type: String,
  required: true,
 },
 date: {
  type: Date,
  default: Date.now
 }
});


const Weather = mongoose.model('Weather', weatherSchema);


module.exports = Weather;
```