МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
(підпис)
грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія моніторингу повітряних тривог»
здобувача групи ІН.м – 33 Кошман Владислав Олександрович

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Владислав КОШМАН
(підпис)

Керівник
доцент кафедри комп'ютерних наук,
д.т.н., старший науковий
співробітник
Микола БУДНИК
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-33 Кошман Владислав Олександрович

1. Тема роботи: «Інформаційна технологія моніторингу повітряних тривог»
затверджую наказом по СумДУ від *«03» грудня 2024 року № 1257-VI*
2. Термін здачі здобувачем кваліфікаційної роботи _____
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються: Були використані такі технологія як API для отримання інформації про повітряні тривоги, flask для написання серверної частини на мові програмування python, 3) Розробка інформаційної система була повністю закінчена, у висновку було отримано готовий сайт та телеграм-бота 4) Після розробки додатку було проведено тестування, яке показало що інформація на сайті є корисною та актуальної. Розроблена система допомогає інформувати користувача про небезпеки.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	10.09.2024	
2	*Огляд технологій, що використовуються для розробки сайтів повітряних тривог*	25.09.2024	
3	*Розробка архітектури сайту*	30.10.2024	
4	*Написання сайту та телеграм-бота*	15.11.2024	
5	*Тестування та аналіз отриманих результатів*	25.11.2024	
	Оформлення пояснювальної записки до кваліфікаційної роботи	01.12.2024	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 67 стр., 19 рис., 2 додаток, 9 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки системи про інформування людей о небезпеках повинні постійно розвиватися, створення нових систем призводить до більшої конкуренції та інновацій у сфері, що в свою чергу допомогає людям

Об’єкт дослідження — Процес створення новинний веб-ресурсів на основі сторонніх API

Мета роботи — Створення веб-додатку для надання людям своєчасної інформації про небезпеки

Методи дослідження — Отримання та обробка даних, їх аналіз для надання інформації

Результати — розроблено інформаційну веб-систему, мета якої повідомляти користувачів про повітряні небезпеки. Створено телеграм-бот і можливість підписки на сповіщення про статус тривоги. Розроблений продукт успішно протестовано

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ МОНІТОРИНГУ ПОВІТРЯНИХ ТРИВОГ, HTML, CSS, FLASK, JAVASCRIPT.

ЗМІСТ

ВСТУП

У сучасному світі, де непередбачувані ситуації можуть траплятися в будь-який момент, система новин та сповіщення про тривоги й катаклізми відіграє ключову роль у захисті життя і майна. Миттєвий доступ до достовірної інформації під час надзвичайних подій є критичним для ухвалення правильних рішень, зокрема для своєчасної евакуації чи пошуку безпечного укриття. Поява різних інформаційних платформ, що надають людям актуальні сповіщення про потенційні загрози, є результатом технологічного прогресу і відповідає на потребу суспільства в оперативній комунікації та захисті. Системи сповіщення, такі як застосунки для мобільних пристроїв, онлайн-сайти та спеціальні канали у соціальних мережах, дозволяють поширювати інформацію майже миттєво. Наприклад, в країнах, які регулярно страждають від землетрусів, таких як Японія, функціонує розгалужена система попереджень, яка за кілька секунд до поштовху сповіщає населення через різноманітні медіа-канали, зокрема мобільні повідомлення та теле- і радіотрансляції. У США існує система оповіщення про торнадо, яка також використовує інтернет і телебачення для своєчасного інформування громадян. Особливої важливості питання сповіщення набуло в умовах збройного конфлікту. Потреба в інформуванні громадян про повітряні тривоги є життєво важливою для зменшення кількості жертв серед цивільного населення. В таких випадках швидкий доступ до інформації про загрозу дозволяє людям захистити себе та своїх близьких, знайти укриття або евакуюватися до безпечнішого місця. На цьому тлі виникає необхідність у створенні надійного та зручного ресурсу, який дозволив би оперативно моніторити ситуацію і отримувати сповіщення про загрози в реальному часі.

Актуальність:

І хоча подібні ресурси вже існують в деякій кількості. Потрібно не забувати що чим більший вибір тим більша конкуренція, а конкуренція приводить до інновацій та покращення продукту. Не можливо створити ідеальний сайт з першого разу. Завжди знайдеться той хто додасть в схожу тему своє бачення та покращить деякий з аспектів. З цього випливає те що тема сайту повітряних тривог є актуальною. Він зможе привнести нові ідеї до вже інсуючих, та зробити досвід більш приємнішим.

Тема: Інформаційна технологія моніторингу повітряних тривог

Мета: Надання людям своєчасної інформації про небезпеки

Об'єкт дослідження: Процес створення новинний ресурсів

Предмет дослідження: Розробка веб-сайту з наглядною інформацією про актуальні небезпеки по регіонам.

Гіпотеза: Навіть при великий кількості схожих продуктів, всі вони відрізняються один від одного та привносять щось нове в сферу. Створення нового продукту також покращить загальну картину

1 АНАЛІТИЧНИЙ ОГЛЯД

В першу чергу, важливо розглянути сучасні підходи до збору, обробки та передачі інформації в реальному часі. З огляду на специфіку та швидкість подій, веб-ресурс повинен мати надійні алгоритми отримання даних та їх миттєвого поширення до користувачів. По-друге, не менш значущим є аналіз інтерфейсу користувача та зручності використання сайту. У кризових ситуаціях, коли швидкість реакції може бути критичною, сайт має бути інтуїтивним та легким у користуванні навіть для тих, хто рідко користується цифровими технологіями. Це включає простоту дизайну, чіткість візуальних елементів та зручну навігацію, що дозволяє користувачам швидко знайти необхідну інформацію. Третім важливим аспектом є аналіз інфраструктурної стійкості сайту, тобто його здатність витримувати великий обсяг відвідувачів та одночасно зберігати стабільну швидкість роботи навіть при пікових навантаженнях. Це може включати розгляд хмарних рішень та інші технології, що підвищують надійність і продуктивність платформи.

Також варто звернути увагу на питання безпеки. Сайт, що інформує про повітряні тривоги, повинен забезпечувати захист особистих даних користувачів та запобігати можливим кіберзагрозам, адже інформація, яку він містить, є вкрай чутливою.

1.1 Джерела та збір даних про повітряні небезпеки

Основою для створення системи моніторингу повітряних тривог є надійні джерела інформації, такі як державні та приватні сервіси, що надають дані про поточний стан загроз. Це можуть бути офіційні канали повідомлень, системи раннього оповіщення, місцеві органи влади, а також сторонні інтеграції з існуючими системами моніторингу. Дані мають надходити в систему миттєво, що потребує налаштування автоматичних алгоритмів збору з різних джерел. Основні джерела інформації про повітряні тривоги:

Державні системи оповіщення та офіційні канали зв’язку. Найбільш надійними джерелами інформації є державні служби, що забезпечують безпеку та координують дії в умовах надзвичайних ситуацій. Це можуть бути підрозділи

національної оборони, системи цивільного захисту, служби з надзвичайних ситуацій та спеціалізовані державні агентства. Зазвичай ці служби мають свої канали для публікації інформації про тривоги, наприклад, через офіційні веб-сайти, SMS-розсилки або соціальні мережі. Також у деяких країнах використовуються спеціальні програми для мобільних пристроїв, які оперативно повідомляють про загрози.

Місцеві органи влади. Місцеві органи влади часто розповсюджують інформацію про повітряні тривоги серед населення у своїх адміністративних межах. Це можуть бути сайти місцевих адміністрацій, офіційні акаунти в соціальних мережах, мобільні додатки або навіть місцеві телерадіостанції. Місцеві ресурси дозволяють деталізувати інформацію для конкретного регіону, що зручно для користувачів, які хочуть дізнатися про ситуацію саме в своєму місті чи районі.

Радіотехнічні та метеорологічні станції. У деяких випадках для виявлення загроз, таких як авіаудари чи природні катаклізми, використовуються дані з радіолокаційних станцій та метеорологічних датчиків. Наприклад, у випадку небезпеки радіолокаційні станції можуть фіксувати наближення повітряних об'єктів та передавати інформацію на основні сервери обробки даних. Ці системи можуть бути як частиною державних служб, так і належати до приватних мереж, що співпрацюють з органами безпеки.

Спеціалізовані сервіси збору даних (API). Існують також спеціалізовані сервіси збору інформації, що надають дані через API (Application Programming Interface). Наприклад, сервіси, які використовують відкриті API, можуть отримувати інформацію з різних джерел, зокрема від міжнародних організацій, моніторингових агентств та комерційних провайдерів даних. Використання таких API дозволяє об’єднати інформацію з багатьох джерел в одному місці та забезпечити централізований збір даних.

Краудсорсинг та соціальні мережі. Під час кризових ситуацій джерелом інформації можуть бути також соціальні мережі та платформи краудсорсингу. Люди часто діляться інформацією про повітряні тривоги або інші загрози в соціальних мережах, що дозволяє швидко отримати дані про ситуацію в конкретному місці.

Однак цей тип даних потребує обов’язкової перевірки, оскільки ймовірність хибної інформації є високою.

З цього можна прийти до висновку що у сучасному світі, під час створення сайту повітряних тривог найкраще всього підійдуть такі варіанти як парсинг інформації з офіційних каналів державної системи сповіщення та отримання даних з вже інснуючих спеціалізованих API. В обох випадках інформація буде достовірна та перевірена і випадки коли вона такою не є дуже рідкі. Також, як не основний блок надання інформації також можна створити стрічку з новинами від неофіційних каналів, але треба забезпечити гарну модерацію, щоб публікуємий контент не порушував закон.

1.2 API для збору інформації

Для збору та аналізу інформації для сучасних сайтів повітряної тривоги використовуються API. Серед найпопулярніших доступних API можна виділити такі як:

- Alerts in ua

Сервіс alerts.in.ua відображає інформацію про повітряні тривоги та інші загрози на мапі України. Усі дані про загрози беруться з офіційних джерел, такі як канал “Повітряна тривога”, ОВА, Суспільне, ДСНС, тощо. Для доступу до API використовується персональний токен, який можна отримати після заповнення форми. Також є обмеження - 8-10 запитів в хвилину з однієї IP адреси. При перевищенні ліміту сервер буде повертати код 429 Too many requests. При систематичному порушені IP та token будуть заблоковані.

- Air Raid Alert API

Цей API також дозволяє отримувати інформацію про повітряні тривоги в Україні в режимі реального часу. Для доступу до API також потрібно мати ключ доступу. Його можна отримати, якщо зв’язатися з командою розробників. Максимальна частота запитів з одної адреси: 10/сек. Максимальна частота запитів по одному API-ключу: 100/сек. Якщо будуть перевищені зазначені ліміти, користувач отримає HTTP 429.

- API Повітряних тривог

REST API, повітряних тривог на території України. Розробниками було прийнято рішення проксювати існуючі розрізнені джерела даних і звести їх до якогось одного уніфікованого та компактного формату. Таймаут кешування сирих даних з боку імплементації - 3 секунди.

Вибір конкретного API залежить від задач, які він повинен виконувати та мови програмування. Зазвичай для написання більшості сайтів використовуються такі мови як Python та PHP. А шкидкість реакції API на запити залежить від того скільки користувачів будуть заходити на сайт та перевіряти інформацію. Тож загалом для реалізації самого простого сайту підійде будь-який апі, який має актуальну інформацію.

1.3 Дизайн та структура сайту

При створені сайту немало важливим є і те як користувач буде його бачити. Для сайтів з новинами або для сайтів які попереджають про тривогу найкраще за все підійде мінімалістичний інтерфейс, который не буде навантажений зайвою інформацією. Головною задачею є саме швидке попередження користувача про небезпеку. Тому при заході на сайт, він повинен бачити не вікно реєстрації чи головну сторінку з описом сайту, а саме актуальну інформацію.

Якщо звернути увагу на офіційний сайт то можна побачити що на головній сторінці користувача одразу зустрічає актуальна мапа, на якій помічені небезпечні зони та типи небезпеки, рисунок 1.1. також у лівому боці екрану розписані відношення кольорів до ситуації на мапі, це зроблено для того щоб користувач не намагався сам розібратися що саме вони значать. На карті також є актуальний час що також свідчить про те що події відображаються у реальному часі.

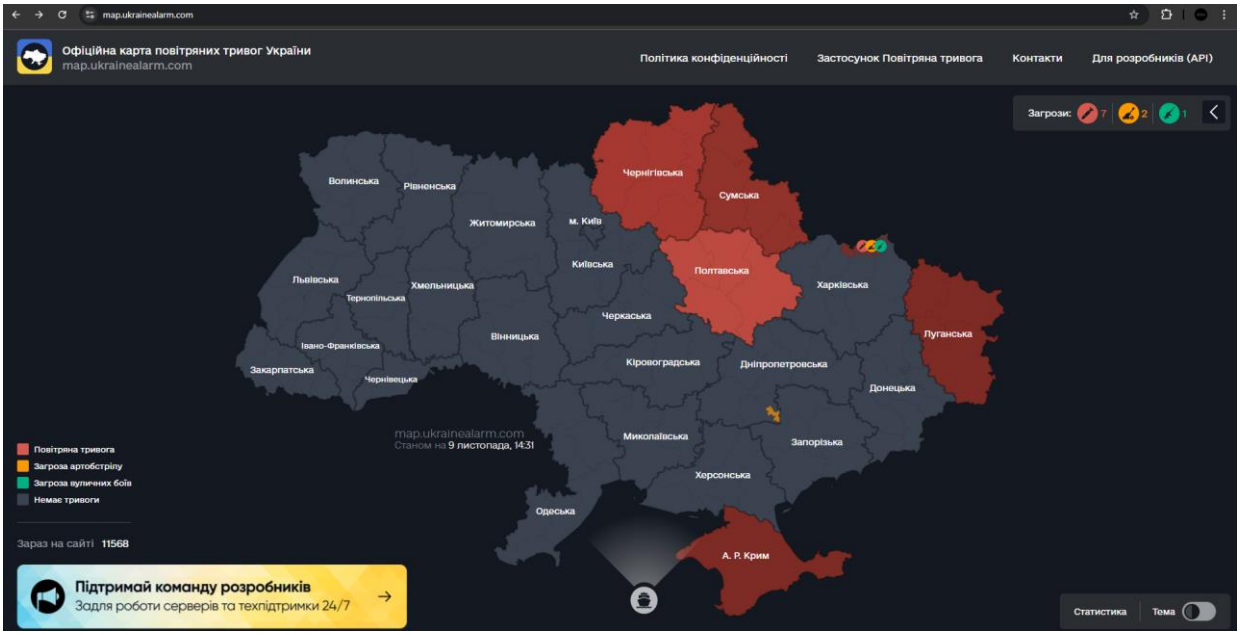

Рисунок 1.1 – Мапа повітряних тривог

Джерело [9]

Але окрім самої мапи на сайті більше немає важливої інформації. Тобто стрічку з новинами чи статистику користувач буде змушений шукати самостійно, або використовувати інший сервіс. Наприклад на неофіційному сайті alerts.in.ua є набагато більша кількість інформації, загалом там присутня історія тривог, вбудована стрічка новин та статистика, рисунки 1.2-1.3.

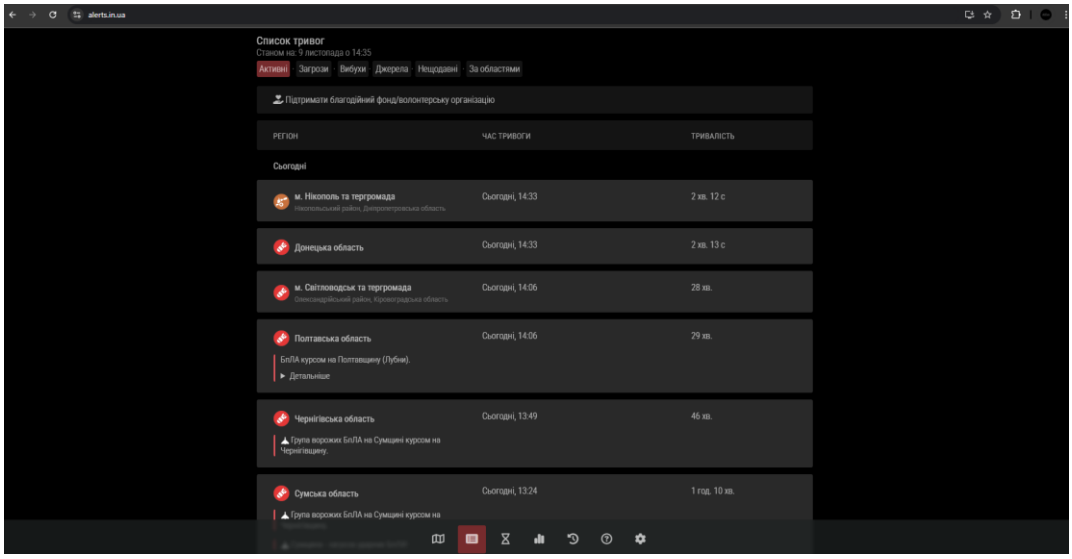

Рисунок 1.2 – Історія тривог

Джерело [9]

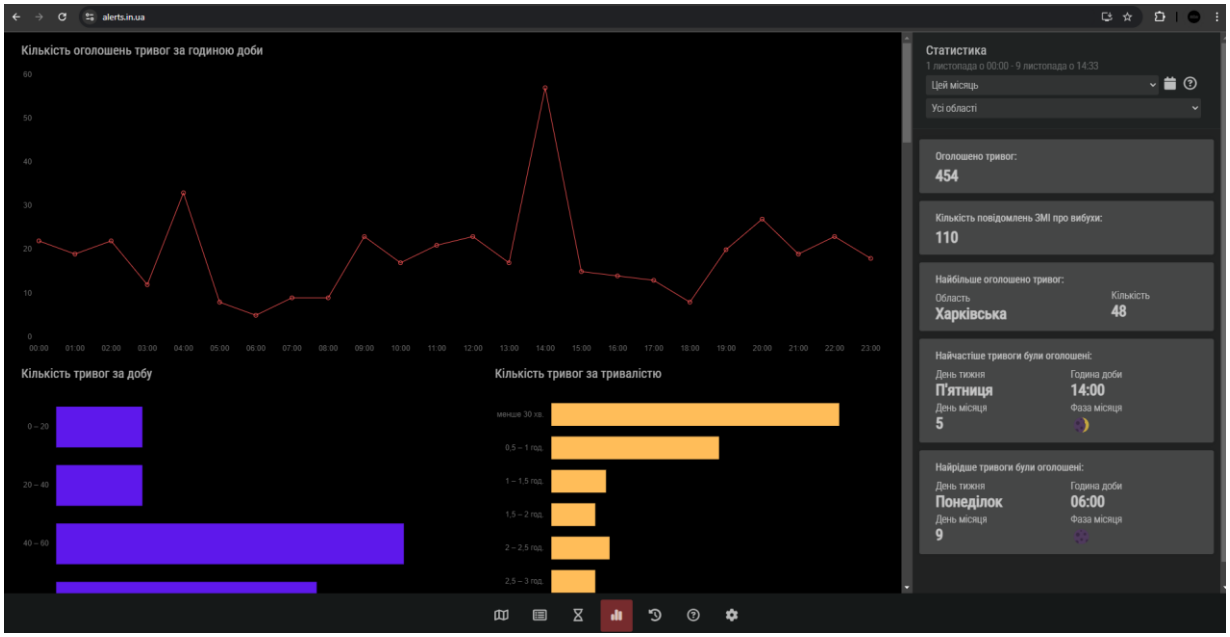

Рисунок 1.3 – Статистика тривог

Джерело [9]

На основі цих двох прикладів можна прийти до висновку що різні сайти пропонують різний набір інформації. Дяки з них націлені на швидке попередження про загрозу і не мають більш ніяко функціоналу окрім мапи тривог. Деякі сайти несуть в собі мету надати більш широку інформації і статистику для аналізу, це може включати в себе як останні регіони з тривогами так і історії самих тривог. Жоден з сайтів не є ідеальним і кожен з них можна доповнити додатковими пунктами та елементами щоб надавати більш ширшу інформації або робити це більш лаконічно та швидко.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Мета та задачі дослідження

Розробка веб-сайту для моніторингу та відображення інформації про повітряні тривоги на основі загальнодоступного API. Сайт повинен надавати користувачам актуальну інформацію про тривоги, а також статистичні дані для аналізу частоти та географічного розподілу тривог.

Основними елементами дослідження, які потрібні у реалізації є:

- Аналіз вимог до сайту: Визначення основних функцій та характеристик, яких потребує сайт для ефективного моніторингу повітряних тривог. Це включає інтерактивну картографію, повідомлення про тривоги, статистику та історичні дані.
- Вибір інструментів та технологій: Оцінка переваг використання Python та Flask для розробки серверної частини сайту, а також вибір найбільш підходящого API для отримання даних про повітряні тривоги.
- Проектування архітектури сайту: Створення структури веб-сайту, визначення взаємодії між сервером, базою даних і клієнтським інтерфейсом.
- Розробка інтерфейсу користувача: Створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, що дозволяє швидко отримувати актуальну інформацію про тривоги та переглядати статистику.
- Інтеграція з API: Реалізація механізмів отримання даних про повітряні тривоги з відкритих джерел через API та їх відображення на сайті в реальному часі.
- Розробка статистичних функцій: Реалізація можливості для користувачів переглядати статистику, таку як кількість тривог по дням, по регіонах та інші корисні метрики.
- Тестування та оптимізація: Перевірка працездатності сайту, усунення можливих помилок та оптимізація швидкодії для забезпечення стабільної роботи при великих навантаженнях.

2.2 Вибір інструментів та методів реалізації

Серед розробників сучасних сайтів найпопулярнішою стала саме мова Python.

Python є однією з найбільш популярних мов програмування завдяки своїй простоті, зручності та потужній екосистемі бібліотек. Однією з основних переваг Python є його синтаксична простота, що дозволяє розробникам швидко впроваджувати нові ідеї та реалізувати складні проекти без зайвих труднощів. Це особливо важливо для розробки веб-додатків, де швидкість реалізації та обробки запитів є критичними. Python має популярні веб-фреймворки, такі як Flask і Django, що дозволяють розробляти як малі, так і масштабовані веб-додатки. Flask особливо підходить для малих проектів і мікросервісів завдяки своїй легкості, тоді як Django ідеально підходить для більших проектів із складною бізнес-логікою. Для інтеграції з іншими веб-сервісами Python надає бібліотеки, які полегшують роботу з API, такі як requests, aiohttp та Flask-RESTful. Також ця мова відома своєю зручністю у роботі з API завдяки популярній бібліотеці requests. Завдяки підтримці численних протоколів і форматів даних (JSON, XML, YAML), Python забезпечує легку інтеграцію з різними відкритими джерелами даних. Також потрібна не забувати що у порівнянні з іншими мовами для написання веб-додатків пайтон має більший обсяг бібліотек для обробки та аналізу даних, такі як pandas, NumPy, matplotlib та seaborn. Це робить Python ідеальним вибором для реалізації статистичних функцій та побудови графіків і діаграм для відображення даних про повітряні тривоги.

За статистикою 2023 року Python є 3 найпоширенішою мовою програмування у світі, рисунок 2.1.

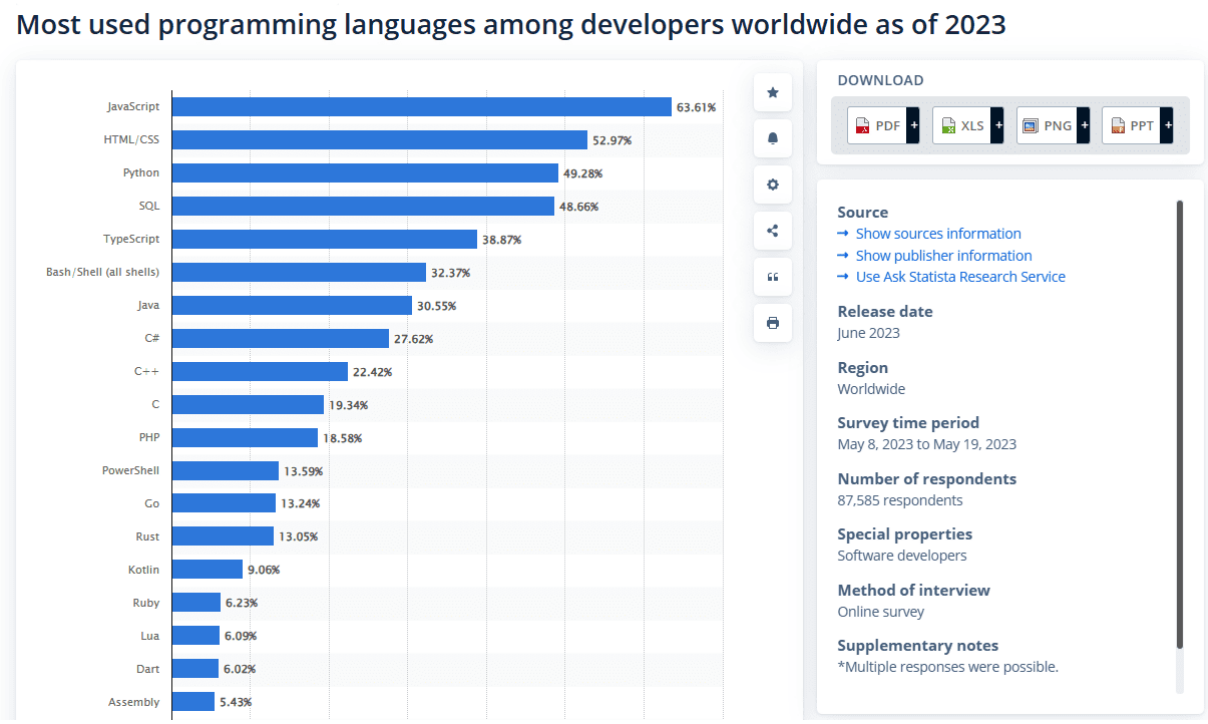

Рисунок 2.1 – Статистика мов програмування

Джерело [6]

Якщо розглядати мінуси та плюси пайтона то можна прийти до висновку що позитивні сторони цієї мови це:

Читабельність і простота: чіткий синтаксис і структура Python покращують читабельність коду та знижують вартість обслуговування програми.

Велика кількість бібліотек. Завдяки багатій стандартній бібліотеці та численним стороннім бібліотекам, таким як NumPy, Pandas і TensorFlow, ви можете прискорити розробку в різних доменах.

Підтримка спільноти: велика й активна спільнота сприяє співпраці, надаючи достатньо ресурсів, посібників і підтримки.

Універсальність: Python підходить для різних програм, включаючи веб-розробку (Django, Flask), науку про дані, машинне навчання та автоматизацію.

Якщо вибирати між Django та Flask потрібно розуміти основні сторони обох фреймворків. Flask — це легкий мікрофреймворк, який не накладає жодних жорстких обмежень на структуру проекту. Це дозволяє розробникам створювати додатки з нуля, адаптуючи код під конкретні вимоги. Він має мінімальний набір компонентів,

які розширюються за допомогою сторонніх бібліотек. Django — це більш важкий фреймворк, який пропонує багато вбудованих функцій, таких як система аутентифікації, адміністрування, форми та інше. Це дозволяє розробляти більш складні проекти з великою кількістю вбудованих функцій, але вимагає більше часу на налаштування і навчання.

Серед переваг Flask можна зазначити Простий і швидкий старт, завдяки мінімалізму. Можливість легко розширюватися для додавання тільки потрібних компонентів (наприклад, для інтеграції з API або додавання статистики).Гнучкість у виборі архітектури та бібліотек.

Серед переваг Django можна заначити великий набір вбудованих компонентів, що спрощує розробку великомасштабних проектів. Можливість швидко розробити складний сайт з великою кількістю користувачів і адміністративних функцій.

Якщо розглядати фрейморк саме для сайту повітряних тривог, який за своєю структурою є дуже простим можна прийти до висновку що Flask є більш гнучким і швидким у використанні для малих проектів, таких як сайт повітряних тривог. Оскільки він не включає багато вбудованих компонентів, цей фреймворк дозволяє досягти високої продуктивності, обробляючи запити швидко. Також він добре підходить для реалізації мікросервісів, що може бути корисно. Django може бути надмірним для простого проекту, оскільки його вбудовані компоненти не завжди потрібні для малих сайтів. Хоча Django забезпечує високу продуктивність, його функції вимагають більше ресурсів і часу на налаштування.

Для написання клієнтської частини сайту будуть використовуватися HTML, CSS та JavaScript. Це основні технології для створення інтерактивних веб-сторінок. Кожна з цих технологій має свої специфічні функції, і їх поєднання дозволяє створити ефективний і привабливий веб-сайт. Розглянемо роль кожної з цих технологій і чому вони є важливими для сайта повітряних тривог.

HTML є основою для створення будь-якого веб-сайту. Це мова розмітки, яка визначає структуру веб-сторінки. Вона описує, як повинні бути організовані елементи на сторінці, а також їх взаємодія.

CSS — це мова стилів, яка визначає, як виглядатиме HTML-елемент на веб-сторінці. CSS дозволяє контролювати кольори, шрифти, відступи, розміри та багато інших візуальних аспектів елементів.

JavaScript — це мова програмування, яка дозволяє додавати динамічні функції до веб-сторінки. За допомогою JavaScript можна створювати інтерактивні елементи, працювати з асинхронними запитами до серверу, а також обробляти події, що відбуваються на сторінці.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Планування архітектури сайту

Для сайту повітряних тривог було обрано клієнт-серверну архітектуру. Серверна частина буде реалізована за допомогою Python, використовуючи фреймворк Flask, який забезпечить ефективну роботу з API для отримання даних про тривоги. Клієнтська частина сайту буде створена за допомогою HTML, CSS та JavaScript, що дозволить реалізувати інтерфейс для відображення мапи тривог, статистики та інших елементів.

Основні компоненти архітектури:

- Серверна частина (Backend):
- Мова програмування: Python
- Фреймворк: Flask
- Інтерфейс з API для отримання даних про тривоги
- Обробка запитів від клієнтської частини
- Взаємодія з базою даних (для збереження статистики, історії тривог)

Клієнтська частина (Frontend):

- Мови: HTML, CSS, JavaScript
- Відображення мапи тривог
- Інтерактивні елементи для користувача (фільтри, кнопки підписки на сповіщення)
- Статистика у вигляді графіків та таблиць

Діаграма серверної частини відображає основні компоненти, які взаємодіють між собою для обробки запитів, отримання та обробки даних, а також для надання результатів клієнтському інтерфейсу, рисунок 3.1. Клієнт (браузер) надсилає запит на сервер через HTTP (наприклад, GET-запит для отримання даних про активні тривоги або статистику). Серверна частина (Flask) обробляє вхідні запити. Flask відповідає за маршрутизацію запитів та управління API-ендпоінтами:

- /alerts – отримує поточні дані про повітряні тривоги через API-запит до зовнішнього джерела.

- /statistics – отримує статистичні дані (наприклад, кількість тривог за певний період).
- /subscribe – обробляє підписку користувачів на сповіщення.

Flask робить запит до зовнішнього API, яке надає актуальну інформацію про повітряні тривоги. Зовнішнє API може бути, наприклад, API від урядових чи міських служб, що надають дані в реальному часі. Сервер також взаємодіє з базою даних для збереження історії тривог та даних користувачів, що підписались на сповіщення. Це може бути база даних SQLite або PostgreSQL, залежно від потреб. Після обробки запиту сервер формує відповідь у форматі JSON, яка містить актуальні дані про тривоги або статистику. Flask повертає сформовану відповідь клієнту, який її потім обробляє та відображає на веб-сторінці.

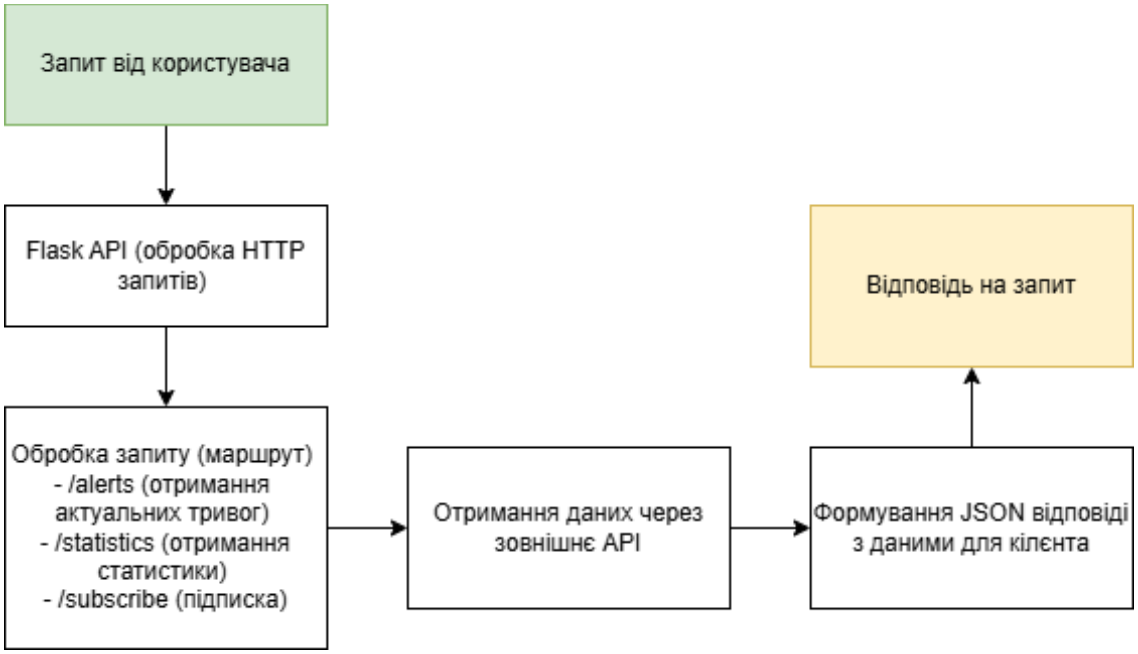

Рисунок 3.1 – Діаграма серверної частини

Джерело: побудовано автором

Сама структура буде нагадувати інші сайти, щоб користувач при заході вже зміг спокійно орієнтуватися та розуміти куди натискати, рисунок 3.2

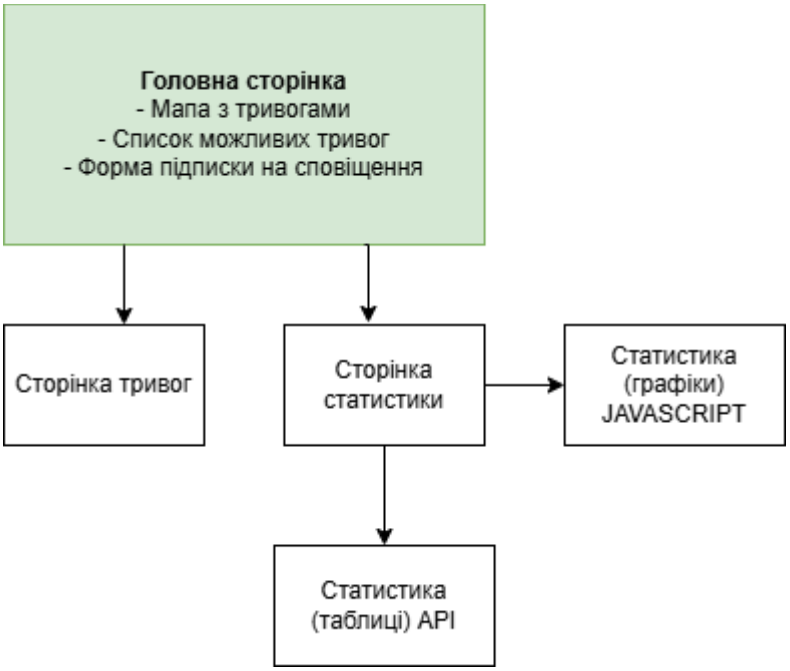

Рисунок 3.2 – Структура сайту

Джерело: побудовано автором

Головна сторінка - Це основний інтерфейс користувача, на якому відображаються такі елементи, як мапа з відмітками про активні повітряні тривоги, статистика та посилання для підписки на сповіщення.

Сторінка тривог – це не основна сторінка з типами тривог, хз описом та історією.

Сторінка статистики – надає актуальну статистику за останній час.

3.2 Планування серверної частини

Для серверної частини сайту вибрано Flask, оскільки він є легким, гнучким і має багато вбудованих можливостей для роботи з API. Flask дозволяє швидко налаштувати сервер та обробляти HTTP-запити. Мова Python була обрана через свою популярність у розробці веб-додатків та наявність широкої підтримки для роботи з API.

Завдання серверної частини:

- Обробка запитів від клієнтської частини (JavaScript).

- Виконання запитів до зовнішніх API для отримання актуальної інформації про повітряні тривоги.
- Виведення відповідних даних у форматі JSON для передачі на фронтенд.
- Можливість збереження історії тривог та статистики для подальшого аналізу.

Приклад роботи Flask з API:

```
from flask import Flask, jsonify
import requests
app = Flask(__name__)
@app.route('/api/air_alerts')
def get_air_alerts():
    response = requests.get('https://api.example.com/alerts')
    data = response.json()
    return jsonify(data)
if __name__ == '__main__':
    app.run(debug=True)
```

Дані будуть отримуватися для подальшої обробки у функціях пайтон. Всі отримані дані будуть мати свої характеристики, наприклад тип тривоги, дата початку тривоги, місцезнаходження та стан, таблиця 3.1.

Таблиця 3.1 – Приклад отриманих даних

Тип тривоги	дата	Місцезнаходження	Стан
Повітряна	2024-11-09	Київ	так
Повітряна	2024-11-09	Одеса	ні
Артилерійна	2024-11-09	Харків	так

3.3 Планування клієнтської частини (Frontend)

Головна сторінка є основним інтерфейсом, на якому користувач взаємодіє з сайтом. Вона повинна бути зручною, інтуїтивно зрозумілою та швидко завантажуваною. Основні компоненти головної сторінки: Назва сайту з логотипом, що містить інформацію про сайт (наприклад, "Повітряні тривоги України"). Інтерактивна мапа, яка відображає місцезнаходження тривог на карті в реальному часі. Посилання для підписки на сповіщення про нові тривоги за місцем перебування

користувача. Ві ці деталі будуть реалізовані за допомогою розмітки Html та наданням стилів CSS. Окремі елементи будуть мати анімації зроблені за допомогою Javascript

3.4 Розробка серверної частини

Серверна частина проекту була розроблена на основі веб-фреймворку Flask, який забезпечує простоту в розробці REST API та інтеграції з різними модулями Python. Код серверної частини допомогає отримувати інформацію для веб-додатку, який повідомляє про тривоги в Україні. Основний функціонал включає отримання даних з API, обробку та кешування інформації, а також її відображення у веб-інтерфейсі.

Основні компоненти серверної частини:

1. Фреймворк Flask

Flask запускає сервер, який працює до повної зупинки коду. Сервер здатен як віддавати дані клієнту так і отримувати дані з сторонніх API та баз даних

2. Кешування

Для зменшення кількості звернень до API та підвищення продуктивності серверної частини використовується бібліотека flask_caching. Було налаштовано просте кешування даних з використанням SimpleCache з таймаутом 60 секунд. alerts_in_ua має чіткий обмежувач на кількість можливих запитів на хвилину, тому відсутність кешуваня призвела б до неможливості отримати дані через перевантаження.

```
app.config['CACHE_TYPE'] = 'SimpleCache'
app.config['CACHE_DEFAULT_TIMEOUT'] = 60
cache = Cache(app)
```

3. Клієнт для роботи з API

Реалізовано інтеграцію з API сервісу Alerts UA через сторонню бібліотеку alerts_in_ua, яка була розроблена ентузіастами для надання доступу до інформації про тривоги в Україні. Бібліотека забезпечує отримання даних про тривоги за областями та активні тривоги. Для отримання унікального токену, який потрібен для доступу до API, був надісланий лист до розробників.

```
alerts_client = AlertsClient(token="унікальний токен")
@cache.cached(timeout=60, key_prefix='alert_statuses')
def get_alert_statuses():
    return alerts_client.get_air_raid_alert_statuses_by_oblast()
```

4. Роутинг

Сервер має кілька маршрутів для забезпечення функціоналу, включаючи:

- / — головна сторінка з відображенням активних тривог;

- /alerts_data — повертає дані про кількість тривог за останні 24 години;

- /alerts_summary — підсумок за типами тривог у кожному регіоні.

5. Обробка даних

Дані з API обробляються з використанням регулярних виразів для вилучення інформації та перетворення дат у потрібний формат. Використовується бібліотека pytz для роботи з часовими зонами, щоб забезпечити правильне відображення часу тривог.

6. JSON-файли

Частина даних зберігається в JSON-файлах. Це дозволяє створювати статистичні зведення та зменшувати навантаження на зовнішній API. Наприклад, маршрут /alerts_summary використовує ці дані для підрахунку кількості тривог за типами в кожному регіоні. JSON записи використовуються замість повноцінної бази даних. Через те що всі отримані з API дані можна помістити в одну таблицю, а також кількість потрібних даних обмежена останніми 24 годинами, було вирішено повністю відмовитися від БД та зберігати дані у JSON

7. Веб-інтерфейс

Дані передаються у HTML-шаблони з використанням render_template, що дозволяє створювати інтерактивний веб-інтерфейс.

Тож, підсумовуючи можна сказати що всі дії по отриманню, наданню та запису даних відбувається саме у flask сервері. Клієнтська частина надає запит до серверу, а той в свою чергу виконує дії, які відповідають маршруту запитів на стороні користувача, рисунок 3.3

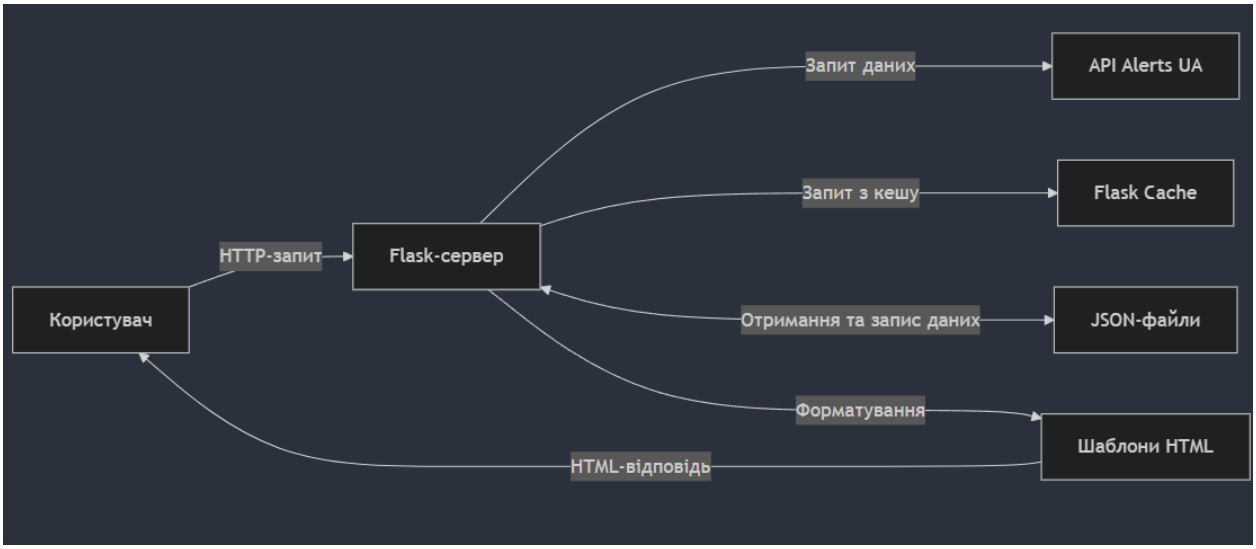

Рисунок 3.3 – графічна діаграма роботи серверу

Джерело: побудовано автором

3.5 Розробка головної сторінки

Головна сторінка сайту призначена для відображення поточного стану повітряної тривоги в різних областях України. Основний контент організований так, щоб користувач міг швидко і зручно отримати інформацію. Коли користувач заходить на сайт він бачить перед собою Карту України з позначеними областями в яких зараз триває тривога. У правому куті екрану знаходиться меню для переходу між різними сторінками веб-додатку. Знизу розташовані кольори та їх позначення, це зроблено для того щоб користувач краще орієнтувався про стан тривоги, особливо якщо він бачить сайт вперше, рисунок 3.4.

Рисунок 3.4 – Головна сторінка сайту

Джерело: побудовано автором(знімок екрану)

Також на сайті є можливість для зміни теми на світлу, рисунок 3.5.

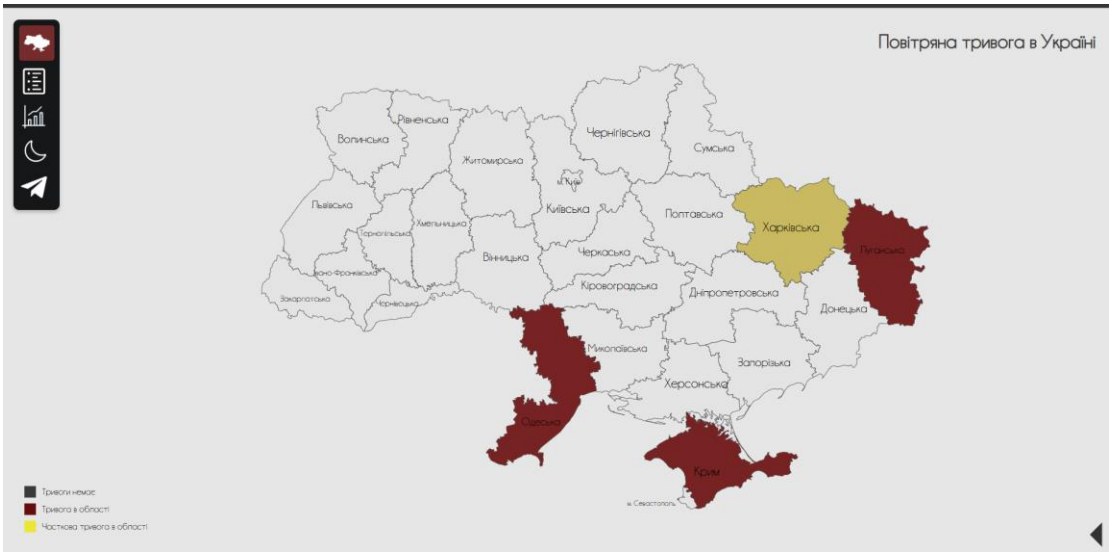

Рисунок 3.4 – Головна сторінка сайту(світла тема)

Джерело: побудовано автором(знімок екрану)

Зміна теми відбувається за допомогою натискання на відповідний елемент меню(місяць або сонце). Після того як користувач натискає на цей елемент то у відповідних елементів сайту змінюються класи:

```
if (mainBlock.classList.contains('light-mode')) {
        mainBlock.classList.remove('light-mode');
        mainBlock2.classList.remove('light-mode');
        infoBlock.classList.remove('light-mode');
        svg.classList.remove('light-mode');
        img.src = 'static/images/sun_icon.png';
    } else {
        mainBlock.classList.add('light-mode');
        mainBlock2.classList.add('light-mode');
        infoBlock.classList.add('light-mode');
        svg.classList.add('light-mode');
        img.src = 'static/images/moon_icon.png';
    }
```

При наведенні на області на мапі спливає блок для більш точного відображення стану у цій області. Цей блок реалізовано за допомогою tooltip та функції addEventListener, яка спрацьовує коли користувач наводить курсор, рисунок 3.6

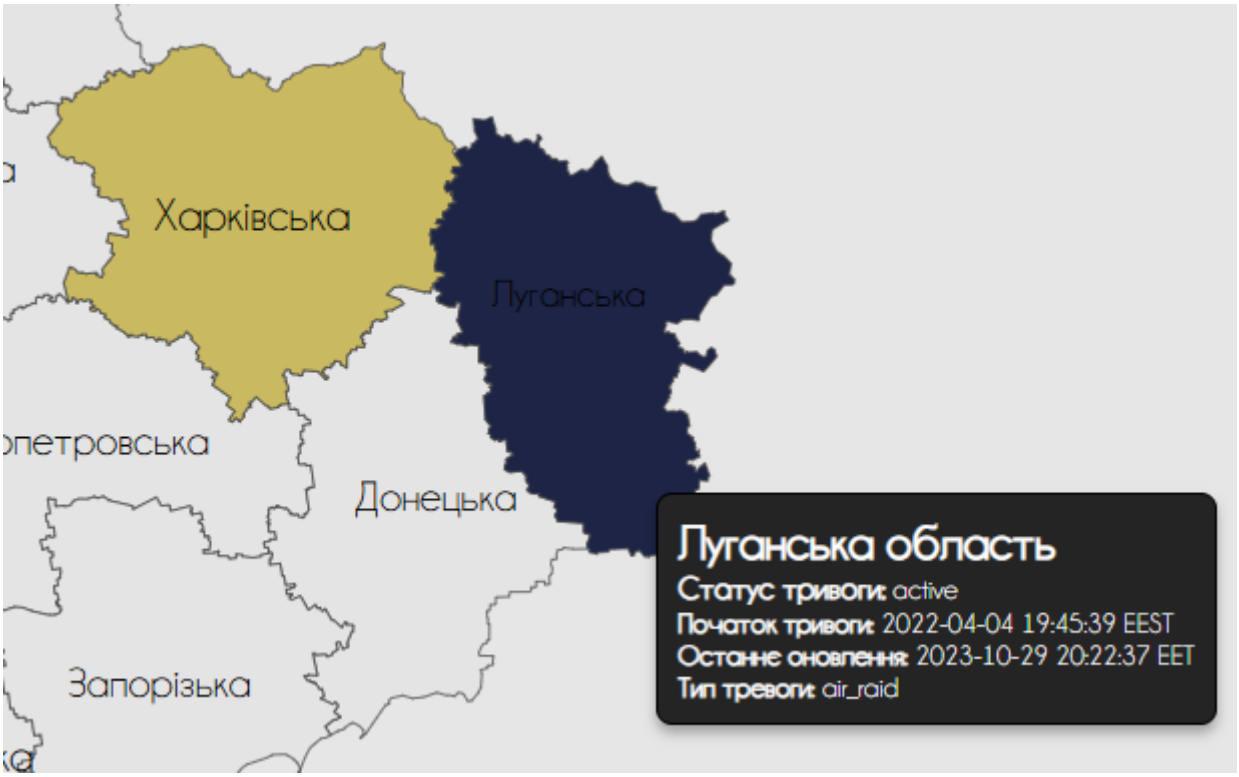

Рисунок 3.6 – Відображення додаткової інформації про область

Джерело: побудовано автором(знімок екрану)

Сама мапа була створено за допомогою SVG (Scalable Vector Graphics). Кожна область представлена шляхом <path> із унікальним атрибутом id, наприклад, м. Севастополь. У атрибуті title зберігається назва області. Колір області змінюється залежно від стану тривоги. Для цього використовується CSS для відображення меж областей та їх заливки.

Скрипт змінює стилі відповідних <path> (областей). Для кожного стану визначено класи:

- no-alarm для сірого кольору.
- alarm для червоного кольору.
- partly-alarm для жовтого кольору.

Також майже усі елементи головної сторінки написані з урахуванням адаптивності, тому що мають у собі процентне значення заповнення.

3.6 Розробка сторінки з списком тривог

Сторінка зі списком тривог потрібна для виведення інформації про всі поточні тривоги у списку. Ця сторінка представляє інтерактивний інтерфейс для відображення списку тривог у структурованому вигляді. Основний елемент HTML має ідентифікатор main-block2 і містить заголовок "Список тривог", а також шаблон для подальшого додавання елементів, які відображають інформацію про тип тривоги, її локацію та час.

У коді визначено два об'єкти: alertIcons і alertTypes.

- alertIcons зіставляє кожен тип тривоги з відповідною іконкою. Наприклад, для "міських боїв" використовується іконка static/images/urban_fights_icon.png. Іконки тривог залежать від типу.
- alertTypes визначає текстові підписи для типів тривог українською мовою, що дозволяє користувачам легко зрозуміти, про що йдеться. І пов’язати текстову та візуальну інформацію про стан тривоги

Функція formatDate перетворює дату у форматі, зручному для користувача. Вона приймає рядок з датою та часом, виділяє відповідні частини (день, місяць, рік, години, хвилини) і повертає їх у форматі дд.мм.рррр о гг:хх. Це робить відображення часу уніфікованим і зрозумілим.

Функція displayAlerts відповідає за динамічне створення та відображення блоків тривог. Вона приймає список даних alertData, де кожен елемент масиву містить інформацію про одну тривогу: локацію, час початку, час завершення та тип тривоги. Для кожної тривоги створюється новий div із класом alert-block, до якого додаються вже знайому елементи:

- Іконка тривоги, відповідно до її типу.
- Назва локації.
- Тип тривоги.
- Відформатований час тривоги.

- Іконка карти, яка відкриває Google Maps із зазначеною локацією в новій вкладці. Це допомогає користувачу отримати географічну інформацію про локацію тривоги.

Ця сторінка є динамічною та адаптивною, вона оновлюється залежно від розміру екрану та кількості вхідних даних, рисунок 3.7.

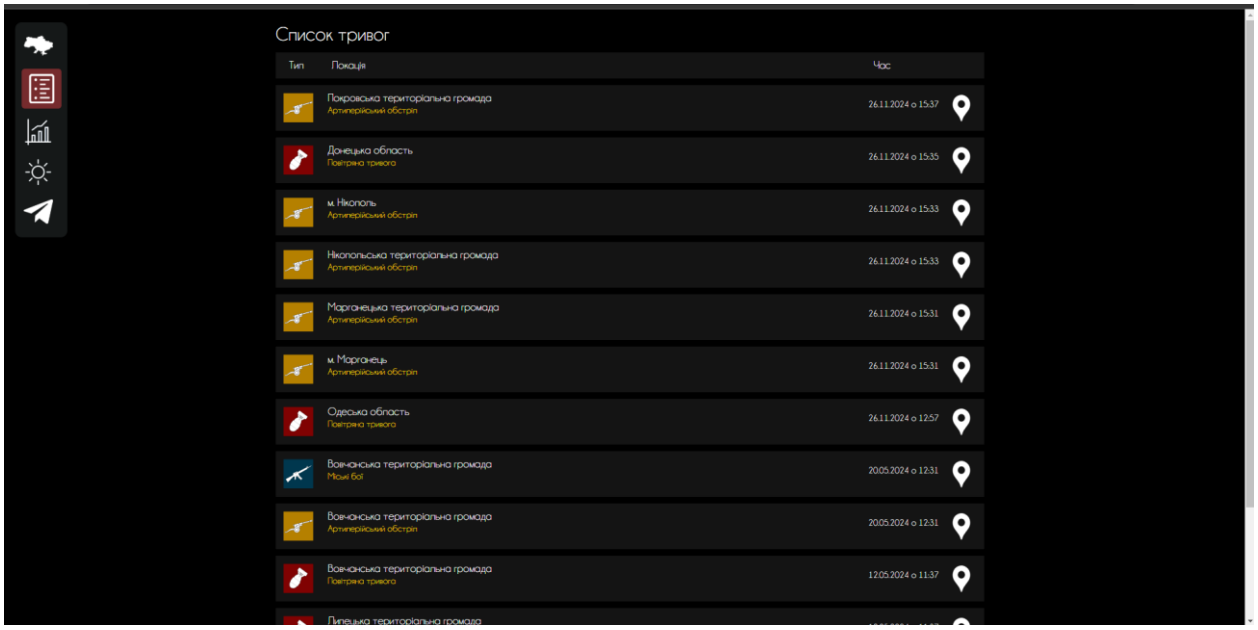

Рисунок 3.7 – Відображення списку тривог

Джерело: побудовано автором(знімок екрану)

3.7 Розробка сторінки з статистикою

Для отримання більш точної статистики по останнім тривогам, була створена додаткова сторінка, яка має статистику про тривоги за останні 24 години. За допомогою цієї статистики користувач може побачити в який час були тривоги та яка їх кількість, а також типи цих тривог, рисунок 3.8.

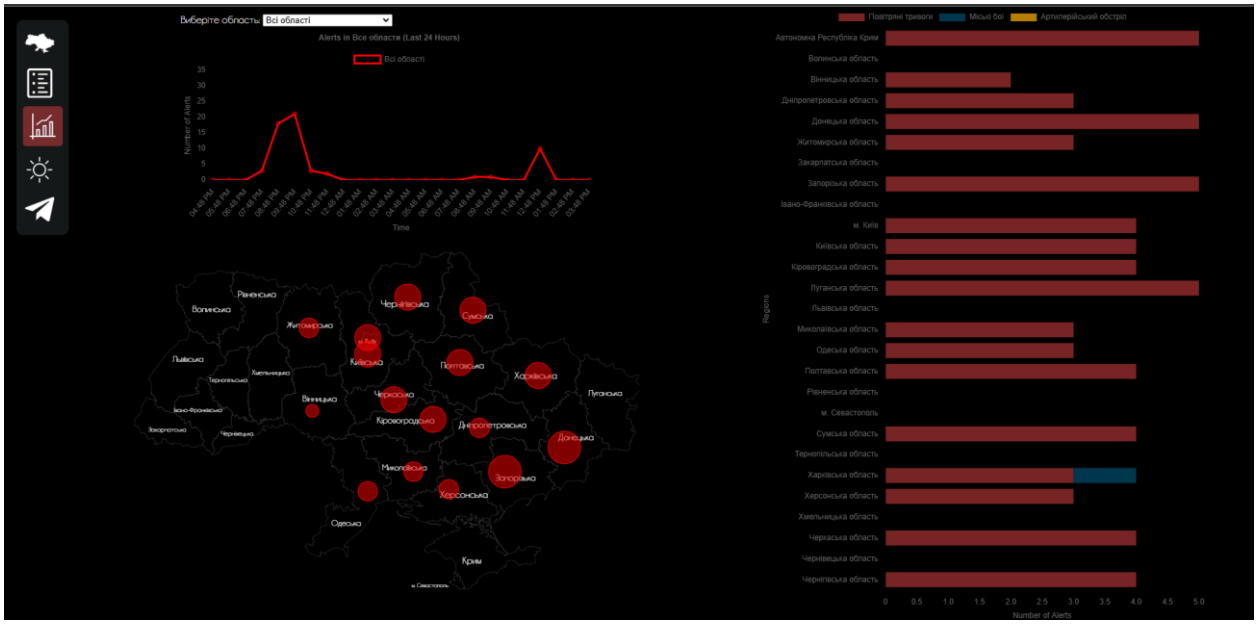

Рисунок 3.8 – Сторінка зі статистикою

Джерело: побудовано автором(знімок екрану)

Графіки на сторінці побудовані за допомогою бібліотеки Chart.js, яка дозволяє створювати динамічні та інтерактивні графіки. Візуалізація побудована за наступним алгоритмом:

1. Спочатку ініціалізується контейнер, в якому буде розміщений графік. Для цього використовується елемент з певними розмірами.

2. Дані для графіків отримуються через запити до файлу у форматі JSON. Дані включають показники, які мають бути відображені на графіку (час, кількість тривог, типи тривог).

3. Для побудови графіків використовуються функції Chart.js, які приймають ці дані та відображають їх у вигляді лінійних графіків, гістограм або кругових діаграм. Графіки налаштовуються так, щоб на осі X був час, а на осі Y — значення відповідних показників.

4. Усі графіки інтерактивні: при наведенні на точку на графіку з'являються підказки з деталями даних у цій точці (наприклад, точний час чи значення).

Дані для графіків завантажуються за допомогою функції Fetch, яка надає запит до серверу, котрий в свою чергу отримує записані дані з json файлу і надсилає їх клієнту. Малювання кола тривоги на мапі реалізоване за допомогою SVG (Scalable

Vector Graphics). Для кожної точки, яка позначає зону тривоги, створюється елемент <circle>. Для кожної зони тривоги використовуються географічні координати (широта і довгота), що визначають місце розташування кола на карті. Розмір кола змінюється в залежності від кількості тривог за останній час. Чим більше тривог тим більше коло у області. У правому куті екрану знаходиться гістограма, яка відображує порівняння кількості тривог по кожній області. Самі стовпчики поділені на декілька кольорів, які відсотково заповнюються в залежності від того яка кількість різних типів тривоги була. При наведені на стовпчики можна отримати інформацію про точну кількість, рисунок 3.9.

Рисунок 3.9 – гістограма

Джерело: побудовано автором(знімок екрану)

3.8 Розробка телеграм-бота для сповіщення

У меню веб-додатка є іконка з логотипом телеграм. При натисканні на цей логотип користувач потрапляє на сторінку чату з ботом. У самому чаті користувач отримує інформацію про поточний стан тривоги і його зміну. У самому початку користувач повинен підписатися на повідомлення про тривогу вибравши свою область, для цього існує команда /setarea, яка відкриває список існуючих областей, рисунок 3.10.

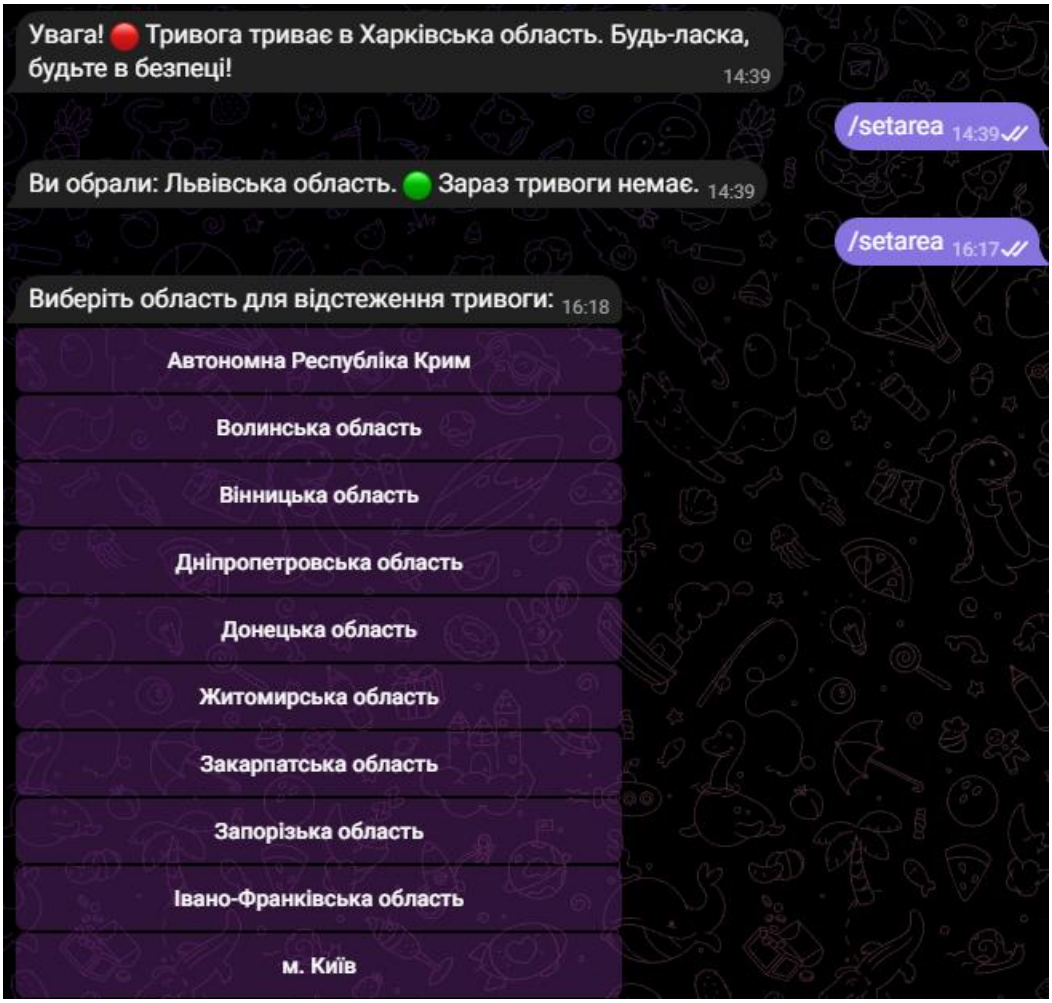

Рисунок 3.10 – телеграм-бот

Джерело: побудовано автором(знімок екрану)

Архітектура бота:

Токен та ініціалізація клієнта API.

У самому початку створюється токен для підключення до Телеграм-бота, а також ініціалізується клієнт AlertsClient з токеном для доступу до API, яке надає інформацію про повітряні тривоги в Україні. Цей клієнт використовує API для отримання статусу тривог по областях країни.

```
    telegram_token = "7779604508:AAGc8iFkv8pyoxS9wM24rVTeXYGPMbyL8kQ"
    alerts_client                                                    =
AlertsClient(token="59bdf80233698f3b73b2cdb0e065b7d906726fb4ab2203")
    areas_alerts = {}
    regions = [
        "Автономна Республіка Крим", "Волинська область", "Вінницька
область", "Дніпропетровська область", "Донецька область",…
```

Словник для зберігання вибору області користувача.

Словник areas_alerts використовується для зберігання інформації про вибір користувача щодо області, для якої він хоче отримувати оновлення про тривоги. У ньому зберігається, яка область була вибрана користувачем. Цей словник взаємодіє з функцією, яка дозволяє перевірити поточний статус тривоги для обраної області.

Список всіх областей.

Список regions містить назви всіх адміністративних одиниць України, що дозволяє користувачам обрати свою область для відстеження тривог. Цей список використовується для формування кнопок для вибору області, коли користувач вводить команду для налаштування області.

Функція check_alerts.

Ця функція відповідає за отримання актуальної інформації про тривоги для кожної області. Вона викликає метод get_air_raid_alert_statuses_by_oblast клієнта AlertsClient, що отримує дані про активні тривоги. Дані про кожну область обробляються, і для кожної області визначається, чи є тривога активною чи ні. Всі ці дані зберігаються в словнику alerts_status, де кожна область є ключем, а її статус (активна чи неактивна тривога) — значенням.

Функція update_alerts.

Ця асинхронна функція відповідає за відправку актуальних повідомлень користувачеві про стан тривоги в обраній області. Коли користувач вводить команду /checkalert, ця функція перевіряє, чи вибрав користувач область, і якщо так, то викликає функцію check_alerts, щоб отримати актуальний статус тривог для цієї області. Якщо тривога активна, бот надсилає користувачеві повідомлення з попередженням про тривогу, в іншому випадку — повідомлення про те, що тривоги немає. Якщо користувач не вибрав область, бот просить його зробити це за допомогою команди /setarea.

Функція set_area.

Ця функція активується командою /setarea, і вона дозволяє користувачеві вибрати область для відстеження тривог. Після виклику цієї команди бот відправляє

користувачеві інтерактивну клавіатуру з кнопками, що відповідають різним областям. Користувач натискає на одну з кнопок, і бот зберігає вибір користувача в сесії. Якщо користувач вибирає область, вона зберігається в context.chat_data, що дозволяє відслідковувати, яку саме область обрав користувач.

Функція button.

Ця функція обробляє натискання кнопок, що з'являються після команди /setarea. Кожна кнопка відповідає певній області, і при натисканні на кнопку бот зберігає вибір області в context.chat_data['area']. Після вибору області бот перевіряє, чи є в обраній області активна тривога, використовуючи функцію check_alerts. Якщо тривога активна, бот надсилає користувачеві попередження про небезпеку, а якщо тривоги немає — повідомлення, що тривога відсутня.

Функція start

Ця функція активується командою /start і є вітальним повідомленням, яке бот надсилає користувачеві при першому зверненні. У цьому повідомленні користувачу нагадують про команду /setarea, що дозволяє вибрати область для відстеження тривог.

Основна функція main

Ця функція є основною точкою входу для запуску бота. Вона ініціалізує застосунок бота за допомогою токену та додає відповідні обробники команд: обробник для команди /start, обробник для команди /setarea, обробник для перевірки тривог через /checkalert і обробник для обробки натискання кнопок. Після цього бот починає працювати в режимі опитування (run_polling), що дозволяє йому постійно перевіряти вхідні повідомлення та реагувати на них.

Перевірка тривог кожну хвилину

Після запуску бота в основній функції є циклічний процес, який кожну хвилину перевіряє статуси тривог по всіх областях за допомогою функції check_alerts. Цей процес здійснюється в окремому циклі while True, який виводить в консоль статуси тривог для кожної області. Це дозволяє отримувати поточні дані про тривоги та оновлювати їх у боті.

Загалом, цей телеграм-бот дозволяє користувачам швидко отримувати актуальну інформацію про повітряні тривоги в обраних областях, а також

налаштовувати область для моніторингу через інтерфейс з кнопками, що робить його зручним для використання в умовах, коли важлива швидка реакція на тривоги.

3.9 Збір та запис інформації про тривоги

Для збору інформації про тривоги та їх тип у JSON файл було написано скрипт, який виконував запит до серверу, отримував та обробляв дані.

Код починається з імпорту необхідних бібліотек і створення клієнта AlertsClient з токеном для доступу до API. Далі визначена функція record(), яка кешує запити до API для отримання статусів тривог по областях та списку активних тривог. Для кожної активної тривоги проводиться парсинг її даних за допомогою регулярних виразів. Зокрема, витягуються місце події, час початку та оновлення тривоги, а також тип тривоги. Часові мітки перетворюються на зручний формат із урахуванням київського часу. Всі ці дані зберігаються у вигляді списку для подальшої обробки.

Функція check_active_alerts() перевіряє зміни у статусах тривог, порівнюючи нові дані з попередніми статусами регіонів. Якщо статус регіону змінився (наприклад, з "без тривоги" на "активна"), то збирається інформація про відповідні тривоги та додається в список для запису. Якщо статус змінився в зворотний бік, попередній статус регіону просто оновлюється. Далі функція намагається завантажити існуючі дані з файлу alerts.json, додає нові записи до цього файлу та зберігає його. Якщо файл не існує або він порожній, створюється новий файл з необхідними даними. Нарешті, функція monitor_alerts() в циклі кожні 120 секунд викликає функції record() і check_active_alerts() для оновлення даних про тривоги. Цей цикл продовжується нескінченно, забезпечуючи постійний моніторинг і оновлення інформації.

3.10 Тестування

Після закінчення розробки веб-додатку та телеграм боту, було протестовано їх роботу. Тестування функціональності веб-додатку було зосереджено на перевірці, чи працюють всі компоненти сайту відповідно до вимог. Для цього було:

1. Перевірено, що мапа (SVG) правильно завантажується, і кожен елемент (наприклад, міста чи області) має коректні інтерактивні дії (наприклад, показ

інформації про тривогу при наведенні). Для цього було порівняно дві мапи, а саме мапу розроблену під час виконання роботи та мапу офіційного сайту повітряних тривог України. Головним критерієм було своєчасне та коректне оновлення інформації на мапі. Тестування показало що дані по областям співпадають та оновлюються в один час з офіційними, рисунок 3.11-3.12

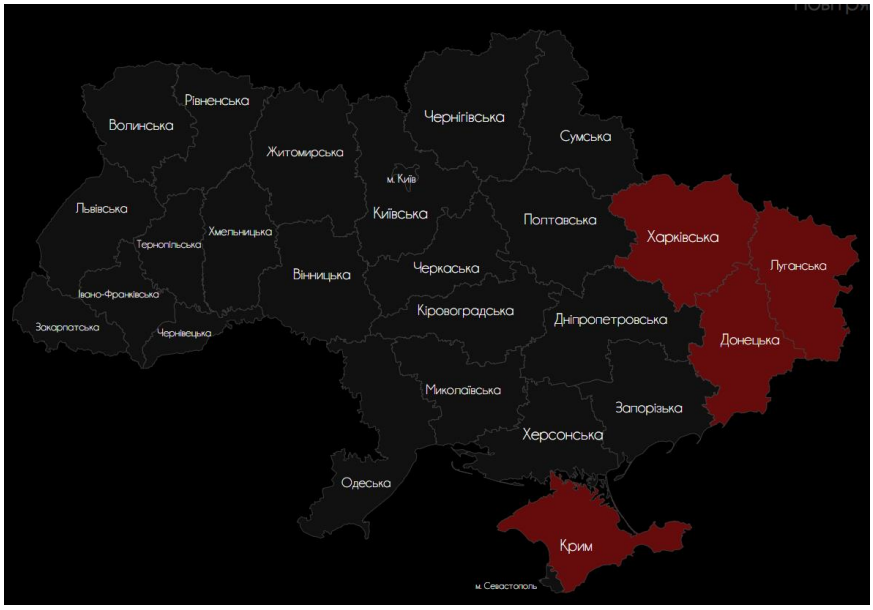

Рисунок 3.11 – мапа з розробленого сайту

Джерело: побудовано автором(знімок екрану)

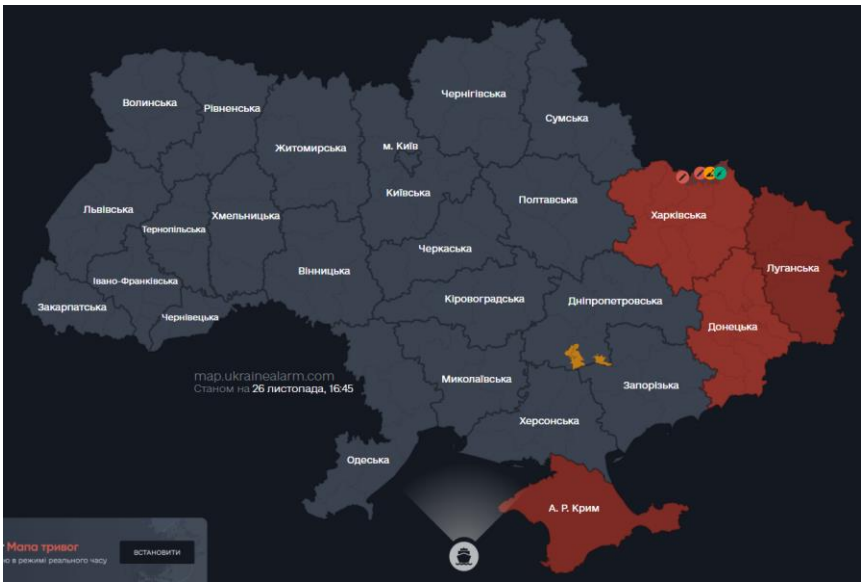

Рисунок 3.13 – мапа з офіційного сайту

Джерело: [9]

2. Було перевірено правильне відображення про тривоги в області ("Тривоги немає", "Тривога в області", "Часткова тривога в області"), яка оновлюються в залежності від зміни ситуації. Тестування з порівнянням з офіційних джерел також показало що додаткова інформація про тривоги є коректною

3. Було перевірено правильне розташування усіх областей та їх назв на мапі

Для перевірки адаптивності сайту було проведено тестування його функцій та відображення елементів меню при зміні розміру екрану. Тестування показало що сайт коректно зменшує розміри блоків, які вказані у відсотковому значені від розміру екрану, рисунок 3.13

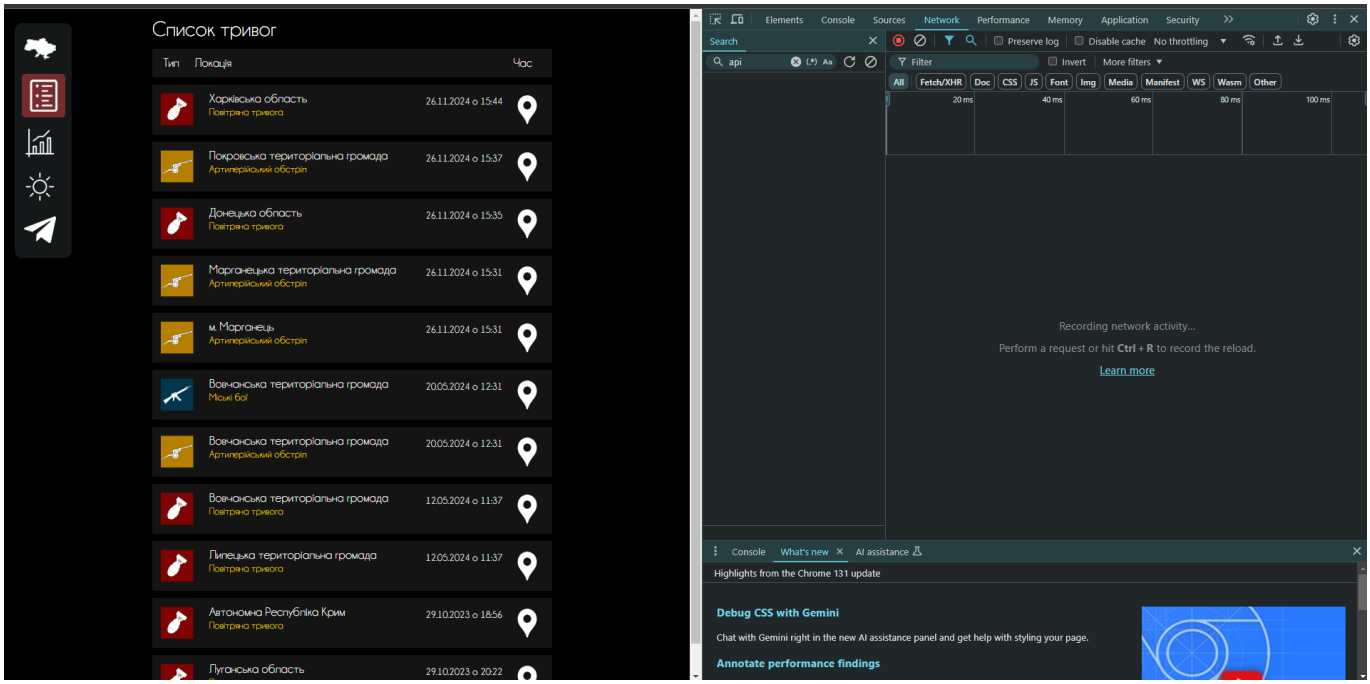

Рисунок 3.13 - адаптивність

Джерело: побудовано автором(знімок екрану)

Також була перевірена стійкість веб-додатку до великої кількості звернень. Через обмеження у 8-10 запитів на API, при перевищені ліміту повертається помилка, саме тому було важно розробити функцію кешування даних, які і беруться кожен раз при оновлені сторінок, а сам запит до API відбувається кожні 60 секунд. Це запобігає перевантаження сайту та зменшує можливість DoS-атаки. Веб-додаток показав

стійкість при великій кількості звернень, тому що йому не потрібно постійно звертатись до сервера при наданні даних клієнту.

Під час тестування телеграм-боту було встановлено декілька основних критеріїв, які повинен виконувати додаток:

1. Повідомлення про зміну статусу тривоги для регіону на який підписаний користувач
2. Повідомлення про статус тривоги при підписці на новий регіон
3. Можливість швидко змінити регіон для підписки

Усі функції телеграм-боту коректно працюють та відображають достовірну інформацію.

ВИСНОВКИ

Розробка інформаційної технології моніторингу повітряних тривог є важливим завданням, яке вимагає застосування сучасних технологій веб-розробки для забезпечення користувачам швидкого доступу до актуальної та важливої інформації. Метою цього проекту є створення ефективного веб-додатку, який надає користувачам можливість отримувати сповіщення про тривоги в реальному часі, переглядати географічне розташування цих тривог на інтерактивній карті та аналізувати статистику. Для розробки цього веб-додатку було обрано використання мови програмування Python, яка забезпечує високу швидкість розробки, масштабованість і гнучкість. Flask, один з найпопулярніших Python-фреймворків, ідеально підходить для створення веб-сайтів, які взаємодіють з API і мають потребу в швидкому обробленні запитів. Використання Flask дозволяє створювати легкі і швидкодіючі сервери, що є важливим аспектом для такого веб-додатку, де потрібна обробка великої кількості запитів в реальному часі. Для клієнтської частини сайту було обрано HTML, CSS та JavaScript, що дозволяє створити зручний, адаптивний і інтерактивний інтерфейс. HTML дає структуру сайту, CSS відповідає за стиль та оформлення, а JavaScript забезпечує динамічну взаємодію з користувачем, наприклад, оновлення даних без перезавантаження сторінки, відображення актуальних тривог на карті та налаштування сповіщень. Сайт побудований за принципом клієнт-серверної архітектури. Серверна частина, яка відповідає за отримання і обробку даних, взаємодіє з відкритим API для отримання інформації про повітряні тривоги в реальному часі. Вся обробка даних і логіка сайту реалізовані на сервері, а клієнтська частина виконує лише функції відображення даних, прийому вводу користувача та оновлення інтерфейсу. Клієнтська частина сайту є важливою для надання зручного та інтуїтивно зрозумілого інтерфейсу для користувачів. Мапа повітряних тривог надає інтерактивний спосіб перегляду тривог, що значно підвищує ефективність взаємодії з сайтом. Статистика, представлені графіки і таблиці, дозволяють користувачам аналізувати дані за певний період, що сприяє глибшому розумінню ситуації з тривогами.

У ході виконання кваліфікаційної роботи було виконано такі завдання:

1. Розроблена архітектура веб-додатку
2. Написаний повний код веб-додатку
3. Написано телеграм-бота для сповіщення про тривоги

Надалі планується збільшити функціонал сайту додавши більш велику кількість статистики для аналізу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Карта повітряних тривог України [електронний ресурс] - Режим доступу до ресурсу: https://vikna.tv/styl-zhyttya/karta-povitryanoyi-tryvogy-onlajn/ (дата звернення: 26.10.2024)
2. Карта повітряних тривог України онлайн [електронний ресурс] - Режим доступу до ресурсу: https://fakty.com.ua/ua/ukraine/suspilstvo/20241109-onlajn-karta-povitryanyh-tryvog/ (дата звернення: 26.10.2024)
3. alerts.in.ua [електронний ресурс] - Режим доступу до ресурсу: https://devs.alerts.in.ua/#documentationrate_limits/ (дата звернення: 26.10.2024)
4. Air Raid Alert API [електронний ресурс] - Режим доступу до ресурсу: https://alerts.com.ua// (дата звернення: 28.10.2024)
5. API Повітряних тривог [електронний ресурс] - Режим доступу до ресурсу: https://wiki.ubilling.net.ua/doku.php?id=aerialalertsapi (дата звернення: 28.10.2024)
6. 10 Best Web Development Languages to Explore In 2024 [електронний ресурс] - Режим доступу до ресурсу: https://www.intelivita.com/blog/web-development-languages/ (дата звернення: 30.10.2024)
7. Flask [електронний ресурс] - Режим доступу до ресурсу: https://flask.palletsprojects.com/en/stable/api// (дата звернення: 30.10.2024)
8. Python 3.13.0 documentation [електронний ресурс] - Режим доступу до ресурсу: https://docs.python.org/3/ (дата звернення: 02.11.2024)
9. Офіційна карта повітряних тривог України [електронний ресурс] - Режим доступу до ресурсу: https://map.ukrainealarm.com/ (дата звернення: 02.11.2024)

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є сайт повітряних тривог, що надає користувачам актуальну інформацію про ситуацію з тривогами через інтеграцію з зовнішнім API та можливістю перегляду цієї інформації на інтерактивній карті.

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити веб-сайт, який забезпечить доступ до актуальних даних про повітряні тривоги в реальному часі через інтеграцію з API.
Measurable (вимірювана)	Сайт має надавати інформацію про тривоги з точністю до 5 хвилин, на карті повинні бути позначені всі тривоги за останні 24 години.
Achievable (досяжна)	Використовувати Python і Flask для серверної частини, HTML, CSS, JavaScript для клієнтської частини, інтеграція з загальнодоступним API.
Relevant (реалістична)	Розробка сайту є актуальною, зважаючи на необхідність своєчасного інформування населення про повітряні тривоги в реальному часі.
Time-framed (обмежена у часі)	Проект буде реалізовано протягом 3 місяців, з чіткими етапами виконання (планування, розробка, тестування, запуск).

Планування змісту структури робіт ІТ-проекту (WBS). Для проекту розробки сайту повітряних тривог застосовуємо **структуру декомпозиції робіт**

(WBS). Це дозволяє чітко розподілити всі етапи роботи на окремі завдання та підзадачі для зручності управління та контролю. (рис Б.1.)

Основні етапи проекту, включаючи обробку даних API, розробку інтерфейсу, тестування та запуск, будуть розбиті на конкретні пакети робіт.

Структура робіт WBS:

1. **Планування і дизайн**
   - Аналіз вимог до системи.
   - Вибір технологій та інструментів.
   - Проектування бази даних.
   - Створення технічної документації.
2. **Розробка серверної частини (Backend)**
   - Налаштування середовища для Flask.
   - Створення серверної логіки для обробки запитів API.
   - Інтеграція з API для отримання даних про тривоги.
   - Налаштування бази даних для зберігання історії тривог.
3. **Розробка клієнтської частини (Frontend)**
   - Розробка HTML-структури сайту.
   - Стилізація сайту за допомогою CSS.
   - Реалізація інтерактивної карти для відображення тривог.
   - Додавання JavaScript для динамічного оновлення даних на карті.
4. **Тестування**
   - Тестування серверної частини.
   - Тестування клієнтської частини.
   - Тестування інтеграції з API.
   - Тестування зручності користувача та відгуку системи.
5. **Запуск та підтримка**
   - Розгортання сайту на сервері.
   - Моніторинг роботи сайту після запуску.
   - Підтримка та оновлення даних.

.

Побудова календарного графіка виконання ІТ-проекту. Для того, щоб мати реальне уявлення про тривалість виконання робіт над проєктом "Розробка сайту повітряних тривог", з урахуванням обмеженості ресурсів, вихідних та святкових днів, необхідно створити календарний графік робіт. Цей графік визначає послідовність та часові рамки всіх завдань, що входять у процес розробки проєкту. На першому етапі буде проведено планування, аналіз вимог до системи, вибір технологій, проектування бази даних і розробка технічної документації. На другому етапі розпочнеться розробка серверної частини, включаючи налаштування середовища для обраної серверної платформи (наприклад, Flask), створення API для обробки запитів, інтеграцію із зовнішніми джерелами даних про повітряні тривоги та проєктування бази даних для зберігання інформації про тривоги. Третій етап охоплює створення клієнтської частини сайту: верстку HTML-структури, стилізацію (CSS), розробку інтерактивних компонентів та інтеграцію функціональності для користувачів, включаючи інтерактивну карту для відображення актуальної інформації. Діаграма Ганта допоможе чітко візуалізувати всі завдання проєкту на горизонтальній шкалі часу. Кожне завдання або підзадача буде розміщено відповідно до плану, із зазначенням тривалості, початку та кінця виконання. Такий підхід до організації робіт дає змогу оптимізувати використання ресурсів і забезпечити досягнення кінцевої мети проєкту в заплановані терміни. За допомогою програми Microsoft Project Professional побудовано діаграму Ганта, що демонструє повний календарний графік виконання завдань проєкту.. Усі етапи проекту, включно з їх тривалістю та послідовністю виконання, детально зображено на діаграмі Ганта (рис. Б.2).

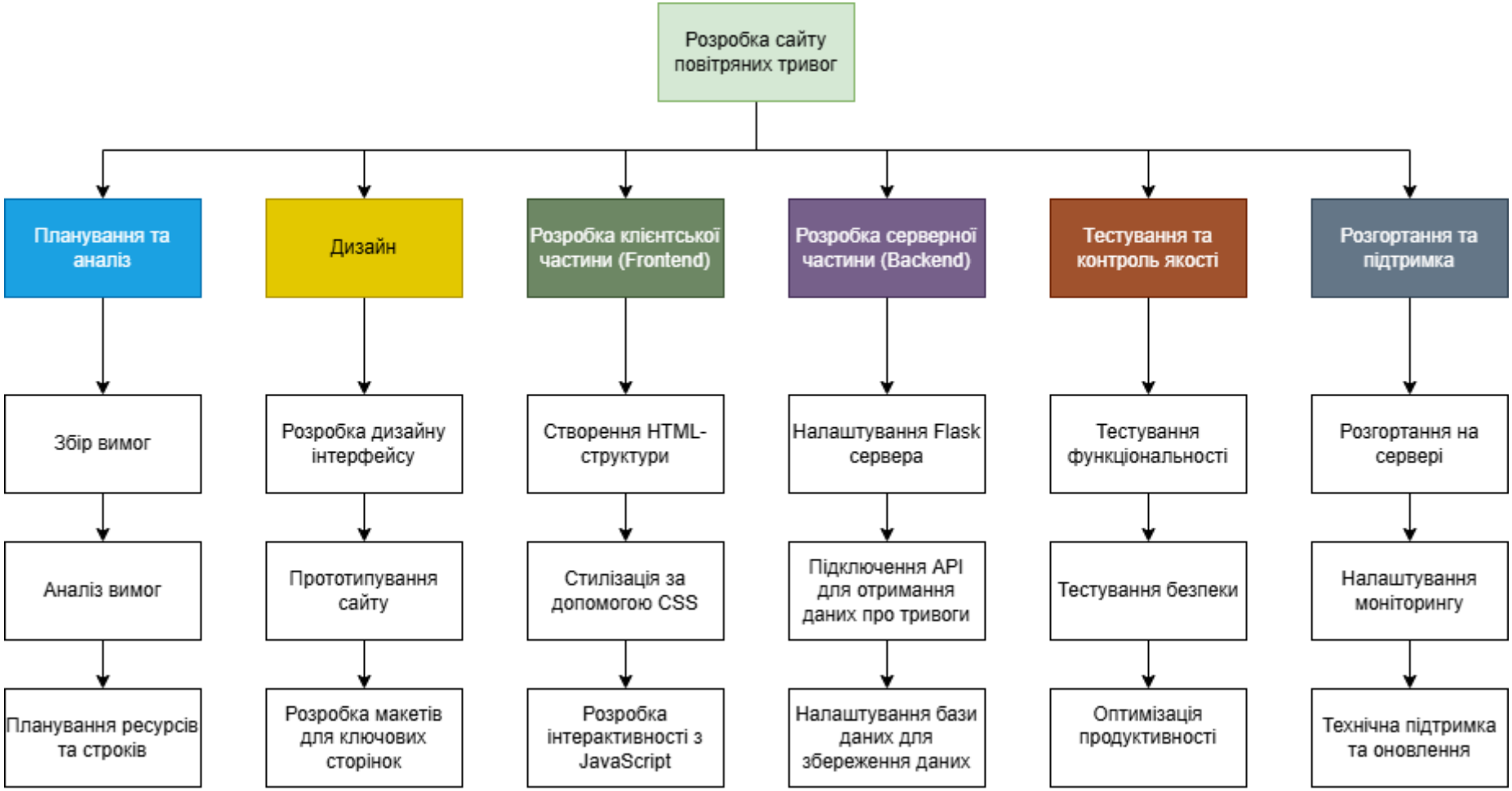

Рисунок Б.1 – таблиця WBS (work breakdown structure)

Джерело: побудовано автором

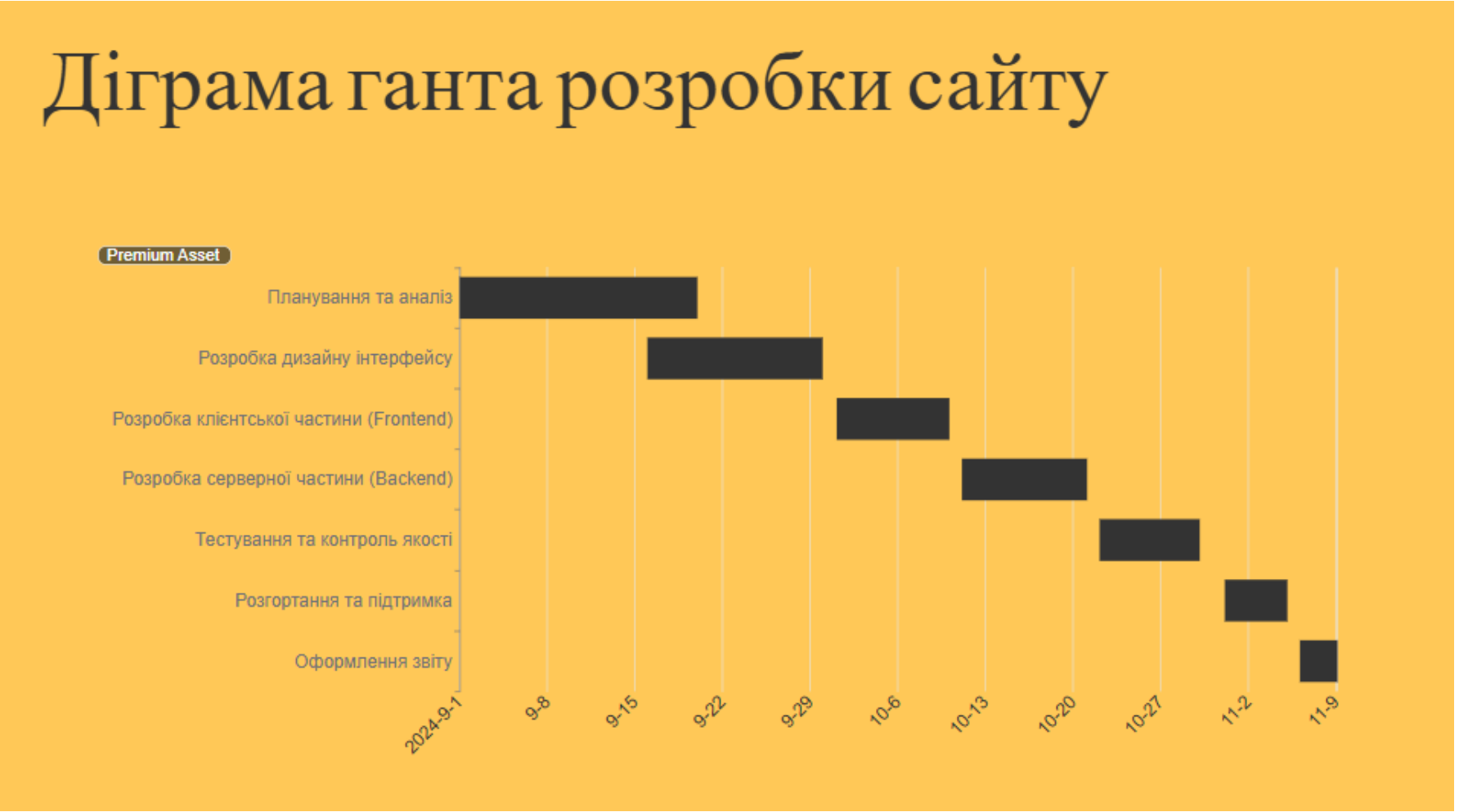

Рисунок Б.2 – Діаграма Ганта

Джерело: побудовано автором

Управління ризиками. До основних ризиків розробки Web-додатку для сайту повітряних тривог є:

- неглибоке вивчення предметної області;
- збільшення навантаження під час реалізації проекту;
- зміна цілей у ході реалізації проекту;
- людський фактор;
- відмова обладнання;
- відсутність кваліфікованого програміста;
- зміна строків виконання роботи;
- незнаходження спільної мови між керівником і виконавцем;
- зростання вимог до проекту.

Таблиця Б.2. Ймовірність виникнення і величина ризику

№	Ризики	Виникнення	Втрати
1	Відсутність досвіду	3	2
2	Збільшення навантаження під час реалізації проекту	3	2
3	Зміна цілей у ході реалізації проекту	4	4
4	Людський фактор	2	3
5	Відмова обладнання	2	4
6	Відсутність кваліфікованого програміста	2	5
7	Зміна строків виконання роботи	2	4
8	Складнощі при впровадженні на місці	3	3
9	Зростання вимог до проекту	3	3

Матриця «Ймовірність – Втрати»

Ймовірність							
-	**5**	5	10	15	20	25
Зміна цілей у ході реалізації проекту	**4**	4	8	12	16	20
Збільшення навантаження під час реалізації проекту Зростання вимог до проекту	**3**	3	6	9	12	15
Відсутність досвіду Людський фактор Відмова обладнання Відсутність кваліфікованого програміста Зміна строків виконання роботи Складнощі при впровадженні на місці	**2**	2	4	6	8	10
-	**1**	1	2	3	4	5
		1	**2**	**3**	**4**	**5**
		-	Збільшення навантаження під час реалізації проекту	Людський фактор	Складнощі при впровадженні на місці Зростання вимог до проекту	Відсутність досвіду	Зміна цілей у ході реалізації проекту Відмова обладнання Зміна строків виконання роботи	Відсутність кваліфікованого програміста

Втрати

Аналізуючи ризики за ймовірністю їх виникнення, можемо їх розділити на:

- Ігноровані
- Відсутні
- Незначні

- Збільшення навантаження під час реалізації проекту
- Людський фактор
- Відмова обладнання
- Зміна строків виконання роботи

- Помірні
  - Відсутність кваліфікованого програміста
  - Зростання вимог до проекту
  - Відсутність досвіду
  - Складнощі при впровадженні на місці
- Істотні
  - Зміна цілей у ході реалізації проекту
- Критичні
  - Відсутні

Класифікація ризиків за рівнем впливу:

- Прийнятні
  - Відсутні
- Виправдані
  - Неглибоке вивчення предметної області
  - Збільшення навантаження під час реалізації проекту
  - Людський фактор
  - Відмова обладнання
  - Відсутність кваліфікованого програміста
  - Зміна строків виконання роботи
  - Не знаходження спільної мови між керівником і виконавцем

- Зростання вимог до проекту

- Неприпустимі

- Зміна цілей у ході реалізації проекту.

ДОДАТОК Б

```
from flask import Flask, render_template, jsonify
from flask_caching import Cache
from alerts_in_ua import Client as AlertsClient
import ast
from datetime import datetime, timedelta
import json
import re
import pytz
# http://127.0.0.1:5000
app = Flask(__name__)
app.config['CACHE_TYPE'] = 'SimpleCache'
app.config['CACHE_DEFAULT_TIMEOUT'] = 120
cache = Cache(app)

alerts_client                                                    =
AlertsClient(token="59bdf80233698f3b73b2cdb0e065b7d906726fb4ab2203")
@app.route('/')
def index():
    # Кешуємо запит статусів тривог по областях
    @cache.cached(timeout=120, key_prefix='alert_statuses')
    def get_alert_statuses():
        return alerts_client.get_air_raid_alert_statuses_by_oblast()

    # Кешуємо запит активних тривог
    @cache.cached(timeout=120, key_prefix='active_alerts')
    def get_active_alerts():
        return alerts_client.get_active_alerts()

    # Отримуємо дані з кешу
    active_alerts = get_alert_statuses()
    active_alerts_more = get_active_alerts()

    # Форматуємо дані для передачі до шаблону
    alert_list = [str(alert) for alert in active_alerts]
    alert_more_list = [str(alert) for alert in active_alerts_more]
    alert_more_list_processed = []
    for data in alert_more_list:
        temp_mass = []
        location_title   =   re.search(r"'location_title':   '([^']+)'",
data).group(1)
        started_at              =              re.search(r"'started_at':
datetime\.datetime\(([^)]+)\)", data).group(1)
        date_str = started_at.split(", tzinfo")[0]
        date_parts = [int(part) for part in date_str.split(",")]
        time_obj = datetime(*date_parts)
        tz = pytz.timezone("Europe/Kyiv")
        time_obj = tz.localize(time_obj)
        formatted_start_time = time_obj.strftime('%Y-%m-%d %H:%M:%S %Z')
```

```
        updated_at                =                re.search(r"'updated_at':
datetime\.datetime\(([^)]+)\)", data).group(1)
        date_str = updated_at.split(", tzinfo")[0]
        date_parts = [int(part) for part in date_str.split(",")]
        time_obj = datetime(*date_parts)
        tz = pytz.timezone("Europe/Kyiv")
        time_obj = tz.localize(time_obj)
        formatted_updated_time  =  time_obj.strftime('%Y-%m-%d  %H:%M:%S
%Z')

        alert_type     =     re.search(r"'alert_type':     '([^']+)'",
data).group(1)

        temp_mass.append(location_title)
        temp_mass.append(formatted_start_time)
        temp_mass.append(formatted_updated_time)
        temp_mass.append(alert_type)
        alert_more_list_processed.append(temp_mass)


    return     render_template('index.html',     alert_data=alert_list,
alert_more_data=alert_more_list_processed)

# @app.route('/')
# def index():
#                alert_list   =['active:Автономна   Республіка   Крим',
'no_alert:Волинська     область',     'no_alert:Вінницька     область',
'no_alert:Дніпропетровська   область',   'no_alert:Донецька   область',
'active:Житомирська    область',    'no_alert:Закарпатська    область',
'no_alert:Запорізька область', 'no_alert:Івано-Франківська область',
'no_alert:м. Київ', 'active:Київська область', 'partly:Кіровоградська
область', 'active:Луганська область', 'no_alert:Львівська область',
'no_alert:Миколаївська     область',     'no_alert:Одеська     область',
'active:Полтавська     область',     'no_alert:Рівненська     область',
'no_alert:м.       Севастополь',       'active:Сумська       область',
'no_alert:Тернопільська    область',    'partly:Харківська    область',
'no_alert:Херсонська    область',    'no_alert:Хмельницька    область',
'no_alert:Черкаська    область',    'no_alert:Чернівецька    область',
'active:Чернігівська область']
#       alert_more_list_processed = [['Луганська область', '2022-04-04
19:45:39 EEST', '2023-10-29 20:22:37 EET', 'air_raid'], ['Автономна
Республіка Крим', '2022-12-11 00:22:00 EET', '2023-10-29 18:56:12 EET',
'air_raid'], ['Липецька територіальна громада', '2024-05-12 11:37:29
EEST',   '2024-05-12   11:37:34   EEST',   'air_raid'],   ['Вовчанська
територіальна громада', '2024-05-12 11:37:57 EEST', '2024-05-12 11:37:58
EEST', 'air_raid'], ['Вовчанська територіальна громада', '2024-05-20
12:31:37  EEST',  '2024-05-20  12:31:39  EEST',  'artillery_shelling'],
['Вовчанська територіальна громада', '2024-05-20 12:31:41 EEST', '2024-
05-20 12:31:44 EEST',]]
#         return  render_template('index.html',  alert_data=alert_list,
alert_more_data=alert_more_list_processed)

@app.route('/alerts_data')
```

```
def alerts_data():
    with open('alerts.json', 'r', encoding='utf-8') as f:
        alerts = json.load(f)

    current_time = datetime.now()
    start_time = current_time - timedelta(hours=24)
    hourly_data = {}

    for entry in alerts:
        region = entry['region']
        if region not in hourly_data:
            hourly_data[region] = [0] * 24

        for alert in entry.get('alerts', []):
            # Видаляємо тимчасову зону та парсимо дату та час
            alert_time_str = alert['start_time'].split(' ')[0:2]
            alert_time  =  datetime.strptime('  '.join(alert_time_str),
'%Y-%m-%d %H:%M:%S')

            if start_time <= alert_time <= current_time:
                hour_diff         =         int((current_time         -
alert_time).total_seconds() // 3600)
                hourly_data[region][23 - hour_diff] += 1
    return jsonify(hourly_data)

@app.route('/alerts_summary')
def alerts_summary():
    # Завантажуємо дані з файлу JSON
    with open('alerts.json', 'r', encoding='utf-8') as f:
        alerts_data = json.load(f)

    # Словник для підрахунку кількості попереджень за типами для кожного
регіону
    region_alerts = {}

    for entry in alerts_data:
        region = entry['region']
        if region not in region_alerts:
            region_alerts[region] = {
                'air_raid': 0,
                'urban_fights': 0,
                'artillery_shelling': 0
            }

        for alert in entry.get('alerts', []):
            alert_type = alert.get('alert_type')
            if alert_type in region_alerts[region]:
                region_alerts[region][alert_type] += 1
    print(region_alerts)
    return jsonify(region_alerts)

@app.route('/page1')
def page1():
```

```
    return "<h1>Страница 1</h1>"

@app.route('/page2')
def page2():
    return "<h1>Страница 2</h1>"

@app.route('/page3')
def page3():
    return "<h1>Страница 3</h1>"

@app.route('/page4')
def page4():
    return "<h1>Страница 4</h1>"

@app.route('/page5')
def page5():
    return "<h1>Страница 5</h1>"

if __name__ == '__main__':
    app.run(debug=True)

import time
from alerts_in_ua import Client as AlertsClient
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Application, CommandHandler, CallbackContext,
CallbackQueryHandler

# Токен бота та ініціалізація клієнта для API
telegram_token = "7779604508:AAGc8iFkv8pyoxS9wM24rVTeXYGPMbyL8kQ"
alerts_client                                                    =
AlertsClient(token="59bdf80233698f3b73b2cdb0e065b7d906726fb4ab2203")
areas_alerts = {}
regions = [
    "Автономна Республіка Крим", "Волинська область", "Вінницька
область", "Дніпропетровська область", "Донецька область",
    "Житомирська область", "Закарпатська область", "Запорізька область",
"Івано-Франківська область", "м. Київ",
    "Київська область", "Кіровоградська область", "Луганська область",
"Львівська область", "Миколаївська область",
    "Одеська область", "Полтавська область", "Рівненська область", "м.
Севастополь", "Сумська область", "Тернопільська область",
    "Харківська область", "Херсонська область", "Хмельницька область",
"Черкаська область", "Чернівецька область", "Чернігівська область"
]

# Функція для перевірки тривог
def check_alerts():
    active_alerts                                                =
alerts_client.get_air_raid_alert_statuses_by_oblast()
    alerts_status = {}

    # Парсинг статусу тривог
    for alert in active_alerts:
```

```
        status, area = str(alert).split(":")
        alerts_status[area] = status

    return alerts_status

# Функція для оновлення інформації та надсилання повідомлення
async def update_alerts(update: Update, context: CallbackContext):
    user_id = update.message.chat_id
    if 'area' not in context.chat_data:
        await update.message.reply_text("Виберіть вашу область.
Використовуйте команду /setarea.")
        return

    area = context.chat_data['area']
    alerts_status = check_alerts()

    # Перевіряємо статус тривоги для обраної області
    if area in alerts_status:
        status = alerts_status[area]
        if status == 'active':
            message = f"Тривога в: {area}. Будь-ласка, будьте в
безпеці!"
        else:
            message = f"Тривога знята в: {area}."

        # Надсилаємо повідомлення користувачу
        await update.message.reply_text(message)
    else:
        await update.message.reply_text(f"Не вдалось знайти інформацію
про: {area}. Спробуйте знову.")

# Функція для встановлення області
async def set_area(update: Update, context: CallbackContext):
    keyboard = [
        [InlineKeyboardButton(region, callback_data=region)] for region
in regions
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text("Виберіть область для відстеження
тривоги:", reply_markup=reply_markup)

# Функція обробки вибраної області
async def button(update: Update, context: CallbackContext):
    query = update.callback_query
    user_region = query.data
    context.chat_data['area'] = user_region

    # Перевірка активної тривоги для вибраної області
    alerts_status = check_alerts()
    if user_region in alerts_status and alerts_status[user_region] ==
'active':
        message = f"Увага! Тривога триває в {user_region}. Будь-
```

```
ласка, будьте в безпеці!"
    else:
        message = f"Ви обрали: {user_region}.  Зараз тривоги немає."

    await query.answer()
    await query.edit_message_text(text=message)

# Функція для запуску бота
async def start(update: Update, context: CallbackContext):
    await update.message.reply_text("Привіт! Я бот для відстеження
тривог. Використовуйте /setarea для вибору області.")

# Головна функція для запуску бота
def main():
    application = Application.builder().token(telegram_token).build()

    # Додавання обробників команд
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("setarea", set_area))
    application.add_handler(CommandHandler("checkalert",
update_alerts))
    application.add_handler(CallbackQueryHandler(button))

    # Запуск бота
    application.run_polling()

    # Періодична перевірка тривог щохвилини
    while True:
        print("Перевірка тривог...")
        alerts_status = check_alerts()
        for area, status in alerts_status.items():
            print(f"{area}: {status}")

        time.sleep(60)  # Перевірка щохвилини

if __name__ == '__main__':
    main()

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
    <title>Повітряні тривоги в Україні</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        .map2 {
           position: relative;
           width: 900px;
           height: 800px;
```

```
        }
        #map-canvas {
            position: absolute;
            top: 0;
            left: 0;
            pointer-events: none; /* Щоб миша взаємодіяла лише з картою
*/
        }
    </style>
</head>
<body>
   <div id="sidebar">

      <img   src="static/images/home.png"   alt="Home"   id="home-img"
class="active">
      <img src="static/images/page1-icon.png" id="page1-img" alt="Page
1" >
      <img src="static/images/page3-icon.png" id="page2-img" alt="Page
1">
      <img src="static/images/sun_icon.png" id="page3-img" alt="Page 3">
      <a       href="https://web.telegram.org/k/#@UkraineAirAlerts_bot"
target="_blank">
      <img src="static/images/page4-icon.png" id="page4-img" alt="Page
1">
        </a>
  </div>
   <!-- <div id="tooltip" style="position: absolute; display: none;
background: #fff; border: 1px solid #ccc; padding: 5px; border-radius:
4px; pointer-events: none;"></div> -->
    <div class="main-block" id="main-block">
      <div id="arrow" class="arrow"></div>
      <div  id="tooltip2"  class="tooltip2">Сайт  розроблен  Кошманом
Владом</div>
      <div class="info_block" id = "info_block">
         <div class="item">
             <div class="colored-square_no_alarm"></div>
             <span>Тривоги немає</span>
         </div>
         <div class="item">
             <div class="colored-square_alarm"></div>
             <span>Тривога в області</span>
         </div>
         <div class="item">
             <div class="colored-square_partly"></div>
             <span>Часткова тривога в області</span>
         </div>
     </div>
        <h1 style="font-size: 34px; color: rgb(59, 59, 59); font-family:
Duke_Fill;     margin-left:78%;     margin-top:     1%;     position:
absolute;">Повітряна тривога в Україні</h1>
        <div class="map" style="padding-top: 4%;">
            <svg id="ukraine-map"
            xmlns:mapsvg="http://mapsvg.com"
```

```
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:svg="http://www.w3.org/2000/svg"
            xmlns="http://www.w3.org/2000/svg"
            viewBox="0 0 712.47321 408.0199"
            width="1496.8"
            height="771.2">
</svg>
            <canvas id="map-canvas" width="896" height="771"></canvas>
            <script>
                const alertData2 = [
                    { id: "Автономна Республіка Крим", alerts: 0,
center: { x: 635, y: 507 } },
                    { id: "Волинська область", alerts: 0, center: { x:
241, y: 110 } },
                    { id: "Вінницька область", alerts: 0, center: { x:
396, y: 287 } },
                    { id: "Дніпропетровська область", alerts: 0, center:
{ x: 652, y: 313 } },
                    { id: "Донецька область", alerts: 0, center: { x:
782, y: 343 } },
                    { id: "Житомирська область", alerts: 0, center: { x:
391, y: 160 } },
                    { id: "Закарпатська область", alerts: 0, center: {
x: 186, y: 317 } },
                    { id: "Запорізька область", alerts: 0, center: { x:
691, y: 380 } },
                    { id: "Івано-Франківська область", alerts: 0,
center: { x: 235, y: 300 } },
                    { id: "м. Київ", alerts: 0, center: { x: 481, y: 175
} },
                    { id: "Київська область", alerts: 0, center: { x:
481, y: 200 } },
                    { id: "Кіровоградська область", alerts: 0, center:
{ x: 581, y: 300 } },
                    { id: "Луганська область", alerts: 0, center: { x:
842, y: 293 } },
                    { id: "Львівська область", alerts: 0, center: { x:
191, y: 220 } },
                    { id: "Миколаївська область", alerts: 0, center: {
x: 551, y: 380 } },
                    { id: "Одеська область", alerts: 0, center: { x:
481, y: 410 } },
                    { id: "Полтавська область", alerts: 0, center: { x:
622, y: 213 } },
                    { id: "Рівненська область", alerts: 0, center: { x:
321, y: 110 } },
                    { id: "м. Севастополь", alerts: 0, center: { x: 609,
y: 547 } },
                    { id: "Сумська область", alerts: 0, center: { x:
642, y: 133 } },
                    { id: "Тернопільська область", alerts: 0, center: {
x: 281, y: 230 } },
```

```
                    { id: "Харківська область", alerts: 0, center: { x:
742, y: 233 } },
                    { id: "Херсонська область", alerts: 0, center: { x:
605, y: 407 } },
                    { id: "Хмельницька область", alerts: 0, center: { x:
341, y: 230 } },
                    { id: "Черкаська область", alerts: 0, center: { x:
521, y: 270 } },
                    { id: "Чернівецька область", alerts: 0, center: { x:
265, y: 330 } },
                    { id: "Чернігівська область", alerts: 0, center: {
x: 542, y: 113 } },

                ];
                fetch('/alerts_data')
  .then(response => response.json())
  .then(data => {
    // Перетворимо об'єкт даних
    const result = Object.entries(data).map(([region, numbers]) => {
      return {
        region: region,
        sum: numbers.reduce((acc, num) => acc + num, 0) // Підсумовуємо
числа у масиві
      };
    });

    console.log("Регіони:");
    console.log(result); // Виводимо перетворені дані

    // Обновляємо значення alerts в alertData2 залежно від суми
    result.forEach(({ region, sum }) => {
      // Находим объект региона в alertData2
      const regionData = alertData2.find(item => item.id === region);
      if (regionData) {
        regionData.alerts = sum; // Оновлюємо поле alerts
      }
    });

    console.log("Оновлюємо alertData2:");
    console.log(alertData2);
    drawCircles();
  })
  .catch(error => {
    console.error("помилка:", error);
  });


                // Canvas для відображення кіл
                const canvas = document.getElementById('map-canvas');
                const ctx = canvas.getContext('2d');

                // Малюємо кола
                function drawCircles() {
```

```
                    ctx.clearRect(0, 0, canvas.width, canvas.height);

                    alertData2.forEach(area => {
                        const { x, y } = area.center;
                        const radius = area.alerts*5;
                        ctx.beginPath();
                        ctx.arc(x, y, radius, 0, 2 * Math.PI);
                        ctx.fillStyle = 'rgba(255, 0, 0, 0.5)';
                        ctx.fill();
                        ctx.strokeStyle = 'red';
                        ctx.stroke();
                    });
                }

                drawCircles();
            </script>
        </div>
    </div>

        <div class="mb2">

        <div class="alerts_graph">
        <canvas id="alertsChart2" width="800" height="1100"></canvas>
    <script>
        // Список усіх областей
        const allRegions = [
            'Автономна Республіка Крим', 'Волинська область', 'Вінницька
область', 'Дніпропетровська область',
            'Донецька область', 'Житомирська область', 'Закарпатська
область', 'Запорізька область',
            'Івано-Франківська область', 'м. Київ', 'Київська область',
'Кіровоградська область', 'Луганська область',
            'Львівська область', 'Миколаївська область', 'Одеська
область', 'Полтавська область', 'Рівненська область',
            'м. Севастополь', 'Сумська область', 'Тернопільська
область', 'Харківська область', 'Херсонська область',
            'Хмельницька область', 'Черкаська область', 'Чернівецька
область', 'Чернігівська область'
        ];

        // Завантажуємо дані із сервера
        fetch('/alerts_summary')
            .then(response => response.json())
            .then(data => {
                // Масиви для даних
                const regions = allRegions;
                const airRaidData = [];
                const urbanFightsData = [];
                const artilleryShellingData = [];

                // Для кожного регіону перевіряємо наявність даних, якщо
ні, присвоюємо 0
                regions.forEach(region => {
```

```
                    if (data[region]) {
                        airRaidData.push(data[region].air_raid);
urbanFightsData.push(data[region].urban_fights);
artilleryShellingData.push(data[region].artillery_shelling);
                    } else {
                        airRaidData.push(0);
                        urbanFightsData.push(0);
                        artilleryShellingData.push(0);
                    }
                });

                // Налаштування графіка
                const ctx = document.getElementById('alertsChart2').getContext('2d');
                const chart = new Chart(ctx, {
                    type: 'bar', // Тип графіка
                    data: {
                        labels: regions, // Мітки по регіонах
                        datasets: [
                            {
                                label: 'Повітряні тривоги',
                                data: airRaidData, // Дані для air_raid
                                backgroundColor: 'rgba(121, 36, 36, 255)', // колір
                                stack: 'Stack 0' // Стек для першого типа тривог
                            },
                            {
                                label: 'Міські бої',
                                data: urbanFightsData, // Дані для urban_fights
                                backgroundColor: 'rgba(0,55,78,255)', // колір для urban_fights
                                stack: 'Stack 0' // Стек для 2 типа тривог
                            },
                            {
                                label: 'Артилерійський обстріл',
                                data: artilleryShellingData, // Данные для artillery_shelling
                                backgroundColor: 'rgba(181,128,0,255)',
                                stack: 'Stack 0'
                            }
                        ]
                    },
                    options: {
                        responsive: true,
                        indexAxis: 'y', // Горизонтальний графік
                        scales: {
                            x: {
                                stacked: true, // Включаємо стек для осі
```

```
X
                                title: {
                                    display: true,
                                    text: 'Number of Alerts'
                                }
                            },
                            y: {
                                stacked: true, // Включаємо стек для осі
Y
                                title: {
                                    display: true,
                                    text: 'Regions'
                                }
                            }
                        },
                        plugins: {
                            legend: {
                                position: 'top',
                            },
                            tooltip: {
                                callbacks: {
                                    label: function(tooltipItem) {
                                        const region =
tooltipItem.label;
                                        const type =
tooltipItem.dataset.label;
                                        const value = tooltipItem.raw;
                                        return `${type}: ${value} in
${region}`;
                                    }
                                }
                            }
                        }
                    }
                });
            })
            .catch(error => {
                console.error('Error loading data:', error);
            });
    </script>
        </div>

        </div>

    </div>
```

```
      </div>


</body>

<script>
   // Передаємо дані з Python у JavaScript через JSON
   const alertData = JSON.parse('{{ alert_data | tojson | safe }}');
   const alertData_more = JSON.parse('{{ alert_more_data | tojson | safe
}}');

   const svg = document.getElementById('ukraine-map');
   const alertIcons = {
    'urban_fights': 'static/images/urban_fights_icon.png',
    'air_raid': 'static/images/air_raid_icon.png',
    'artillery_shelling': 'static/images/artillery_shelling_icon.png',
      };
      const alertTypes = {
    urban_fights: 'Міські бої',
    air_raid: 'Повітряна тривога',
    artillery_shelling: 'Артилерійський обстріл'
      };
   console.log(alertData_more)
   console.log(alertData)
   // Створюємо спливаюче блок
   const tooltip = document.createElement('div');
   tooltip.id = 'tooltip';
   tooltip.style.position = 'absolute';
   tooltip.style.backgroundColor = '#242424';
   tooltip.style.color = 'white';
   tooltip.style.textAnchor = "center"
   tooltip.style.padding = '10px';
   tooltip.style.borderRadius = '8px';
   tooltip.style.fontSize = '14px';
   tooltip.style.fontFamily = 'Duke_Fill'
   tooltip.style.border = '1px solid #000';
   tooltip.style.display = 'none';
   tooltip.style.pointerEvents = 'none';
   tooltip.style.boxShadow = '0px 4px 6px rgba(0, 0, 0, 0.3)';
   document.body.appendChild(tooltip);

   // Функція для отримання додаткової інформації за назвою регіону
   function getAlertInfo(region) {
       const  alertInfo  =  alertData_more.find(item  =>  item[0]  ===
region);
       if (alertInfo) {
           return `
               <strong>Початок тривоги:</strong> ${alertInfo[1]}<br>
               <strong>Останнє оновлення:</strong> ${alertInfo[2]}<br>
               <strong>Тип тревоги:</strong> ${alertInfo[3]}<br>
           `;
       }
       return '<strong>В обалсті немає тривоги.</strong>';
```

```
}

// Функція зміни кольору області в залежності від статусу тривоги
function updateMapColors() {
    alertData.forEach(alert => {
        const [status, ...oblastArr] = alert.split(":");
        const oblast = oblastArr.join(' ');

        const path = svg.getElementById(oblast);

        if (path) {
            // Встановлюємо колір залежно від стану тривоги
            if (status === 'no_alert') {
                path.classList.add('alert-green');
            } else if (status === 'active') {
                path.classList.add('alert-red');
            } else if (status === 'partly') {
                path.classList.add('alert-yellow');
            }

            //  Додаємо  події  для  відображення  інформації  при
наведенні
            path.addEventListener('mouseover', (e) => {
                const info = getAlertInfo(oblast);
                tooltip.innerHTML = `
                    <strong                          style="font-size:
24px;">${oblast}</strong> <br>
                    <strong     style="font-size:     16px;">Статус
тривоги:</strong> ${status}<br>
                    ${info}
                `;
                tooltip.style.display = 'block';
                tooltip.style.left = `${e.pageX + 15}px`;
                tooltip.style.top = `${e.pageY + 15}px`;
                path.classList.add('hover-blue');
            });

            path.addEventListener('mousemove', (e) => {
                tooltip.style.left = `${e.pageX + 15}px`;
                tooltip.style.top = `${e.pageY + 15}px`;
            });

            path.addEventListener('mouseout', () => {
                tooltip.style.display = 'none';
                path.classList.remove('hover-blue');
            });
        }
    });
}

// Оновлюємо карту під час завантаження сторінки
window.onload = updateMapColors;
```

```
   function toggleScroll() {
  const page1 = document.getElementById('page1-img');
  if (page1.classList.contains('active')) {
    // Дозволити прокручування
    document.body.style.overflow = 'auto';
  } else {
    // Заблокувати прокручування
    document.body.style.overflow = 'hidden';
  }
}

// Обробники для навігації між блоками
document.getElementById('home-img').addEventListener('click',
function() {
  document.getElementById('main-block').style.display = 'block';
  document.getElementById('main-block2').style.display = 'none';
  document.getElementById('main-block3').style.display = 'none';
  document.getElementById('page1-img').classList.remove('active');
  document.getElementById('page2-img').classList.remove('active');
  document.getElementById('home-img').classList.add('active');
  toggleScroll();
});

document.getElementById('page1-img').addEventListener('click',
function() {
  document.getElementById('main-block2').style.display = 'block';
  document.getElementById('main-block3').style.display = 'none';
  document.getElementById('main-block').style.display = 'none';
  document.getElementById('home-img').classList.remove('active');
  document.getElementById('page2-img').classList.remove('active');
  document.getElementById('page1-img').classList.add('active');
  toggleScroll();
});

document.getElementById('page2-img').addEventListener('click',
function() {
  document.getElementById('main-block3').style.display = 'block';
  document.getElementById('main-block2').style.display = 'none';
  document.getElementById('main-block').style.display = 'none';
  document.getElementById('home-img').classList.remove('active');
  document.getElementById('page1-img').classList.remove('active');
  document.getElementById('page2-img').classList.add('active');
  toggleScroll();
});

document.getElementById('page3-img').addEventListener('click', function
() {
    const mainBlock = document.getElementById('main-block');
    const mainBlock2 = document.getElementById('main-block2');
    const infoBlock = document.getElementById('info_block');
    const img = document.getElementById('page3-img');
    const svg = document.getElementById('ukraine-map');
```

```
    if (mainBlock.classList.contains('light-mode')) {
        mainBlock.classList.remove('light-mode');
        mainBlock2.classList.remove('light-mode');
        infoBlock.classList.remove('light-mode');
        svg.classList.remove('light-mode');
        img.src = 'static/images/sun_icon.png';
    } else {
        mainBlock.classList.add('light-mode');
        mainBlock2.classList.add('light-mode');
        infoBlock.classList.add('light-mode');
        svg.classList.add('light-mode');
        img.src = 'static/images/moon_icon.png';
    }
});

function formatDate(dateString) {
    // Перетворимо рядок дати на об'єкт Date (видаляємо зайвий часовий
пояс і парсим)
    const dateObj = new Date(dateString.split(" ")[0] + 'T' +
dateString.split(" ")[1]);


    const day = dateObj.getDate().toString().padStart(2, '0');
    const month = (dateObj.getMonth() + 1).toString().padStart(2, '0');
// Месяц начинается с 0
    const year = dateObj.getFullYear();
    const hours = dateObj.getHours().toString().padStart(2, '0');
    const minutes = dateObj.getMinutes().toString().padStart(2, '0');


    return `${day}.${month}.${year} о ${hours}:${minutes}`;
}
function displayAlerts(alertData) {
    const mainBlock = document.getElementById('main-block2');
    mainBlock.style.display = 'block';

    alertData.forEach(data => {
        const [location, alertTime, alertlastTime, alertType] = data;


        const alertBlock = document.createElement('div');
        alertBlock.classList.add('alert-block');


        const img = document.createElement('img');
        img.src          =          alertIcons[alertType]          ||
'static/images//default_icon.png';
        alertBlock.appendChild(img);


        const locationDiv = document.createElement('div');
        locationDiv.classList.add('location');
```

```
        locationDiv.textContent = location;
        alertBlock.appendChild(locationDiv);


        const alertTypeDiv = document.createElement('div');
        alertTypeDiv.classList.add('alert-type');
        alertTypeDiv.textContent = alertTypes[alertType] || 'Невідомий
тип тривоги';
        locationDiv.appendChild(alertTypeDiv);

        const timeDiv = document.createElement('div');
        timeDiv.classList.add('time');
        timeDiv.textContent = formatDate(alertlastTime);;
        alertBlock.appendChild(timeDiv);
        const mapIcon = document.createElement('img');
        mapIcon.classList.add('google_set')
        mapIcon.src = 'static/images/map-icon.png';
        mapIcon.alt = 'Open in Google Maps';
        mapIcon.title = 'Open in Google Maps';
        mapIcon.addEventListener('click', function() {
            const query = encodeURIComponent(location);
            const mapsUrl = `https://www.google.com/maps?q=${query}`;
            window.open(mapsUrl, '_blank');
        });

        alertBlock.appendChild(mapIcon);




        mainBlock.appendChild(alertBlock);
    });
}


displayAlerts(alertData_more.reverse());

document.getElementById('main-block2').style.display = 'none';
document.getElementById('main-block3').style.display = 'none';



const arrow = document.getElementById('arrow');
const tooltip2 = document.getElementById('tooltip2');


function showTooltip() {
    tooltip2.style.display = 'block';
    tooltip2.style.opacity = 1;
    tooltip2.style.transform = 'translateX(0)';
}
```

```
function hideTooltip() {
    tooltip2.style.opacity = 0;
    tooltip2.style.transform = 'translateX(20px)';
}


arrow.addEventListener('mouseenter', () => {

    showTooltip();
});

arrow.addEventListener('mouseleave', () => {

    hideTooltip();
});
</script>



</html>
```