БАКАЛАВРСЬКА РОБОТА

БР. ІП - 05.00.00.000 ПЗ

Група ІП-21-4

Гуменів Віталій

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Гуменів Віталій Володимирович

(прізвище, ім'я, по батькові)

УДК 004.4

(індекс)

БАКАЛАВРСЬКА РОБОТА

Проектування та розробка функціонального сповіщувача енергоефективності

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

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

Здобувач освітнього рівня Гуменів В.В.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Тимків Дмитро Федорович, д.т.н., професор

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Гуменіву Віталію Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Проектування та розробка функціонального сповіщувача енергоефективності”

керівник проекту (роботи) Тимків Д.Ф., професор

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 11 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Аналіз області застосування новітніх технологій для забезпечення енергоефективності
2. Представлення фреймворків та платформи для розробки сповіщувача енергоефективності
3. Розробка адаптивних представлень інтерфейсу користувача
4. Реалізація модуля керування обліковими записами
5. Оцінка стратегій розгортання додатку

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Схема HVAC (рис. 1.1)
2. Схема типової системи кондиціонування повітря змінного об'єму (VAV) (рис. 1.2)
3. Блок-схема моделі для системи HVAC за допомогою генетичних алгоритмів (рис. 1.3)
4. Представлення параметрів контролю (рис. 1.4)
5. Архітектурний шаблон MVVM (Model-View-ViewModel) (рис. 2.1)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз області застосування новітніх технологій для забезпечення енергоефективності	15.05.2025	виконано
2	Представлення фреймворків та платформи для розробки сповіщувача енергоефективності	29.05.2025	виконано
3	Розробка адаптивних представлень інтерфейсу користувача	30.05.2025	виконано
4	Реалізація модуля керування обліковими записами	04.06.2025	виконано
5	Оцінка стратегій розгортання додатку	07.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	11.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 79 сторінок, 30 рисунків, список використаних джерел із 30 найменуваннями, 1 додаток.

Мета роботи - розробити програмний додаток для моніторингу та сповіщення про ефективність енергоспоживання з урахуванням сучасних вимог до дизайну, безпеки, користувацького досвіду та можливостей адаптивного управління.

Об'єкт дослідження - процеси управління енергоефективністю у системах HVAC на основі програмних засобів.

Предмет дослідження - методи, засоби та технології розробки додатків сповіщення енергоефективності з використанням сучасних фреймворків, алгоритмів оптимізації та підходів до UX/UI дизайну.

В першому розділі у межах проведеного аналізу предметної області було встановлено актуальність застосування новітніх інформаційних технологій у сфері підвищення енергоефективності.

В другому розділі розглянуто ключові інструменти, технології та підходи, необхідні для ефективної реалізації програмного додатку сповіщувача енергоефективності

В третьому розділі здійснено повноцінну програмну реалізацію функціонального сповіщувача енергоефективності з урахуванням принципів адаптивності, ергономіки, безпеки та масштабованості

Висновок: реалізовано програмний прототип системи. Здійснено розробку адаптивного інтерфейсу, модулів аутентифікації, управління обліковими записами та структурними одиницями.

КЛЮЧОВІ СЛОВА: ЕНЕРГОЕФЕКТИВНІСТЬ; СПОВІЩУВАЧ; HVAC-СИСТЕМИ; ІНТЕРФЕЙС КОРИСТУВАЧА; MATERIAL DESIGN; ФРЕЙМВОРК; МОБІЛЬНИЙ ДОДАТОК; МОНІТОРИНГ; ІНФОРМАЦІЙНА БЕЗПЕКА

ANNOTATION

The bachelor's thesis contains 79 pages, 30 figures, a list of used sources with 30 names, 1 appendix.

The purpose of the work is to develop a software application for monitoring and notification of energy consumption efficiency, taking into account modern requirements for design, security, user experience and adaptive management capabilities.

The object of research is energy efficiency management processes in HVAC systems based on software tools.

The subject of research is methods, tools and technologies for developing energy efficiency notification applications using modern frameworks, optimization algorithms and approaches to UX/UI design.

In the first section, within the framework of the analysis of the subject area, the relevance of the use of the latest information technologies in the field of increasing energy efficiency was established.

The second section discusses the key tools, technologies and approaches required for the effective implementation of the energy efficiency detector software application

The third section provides a full-fledged software implementation of a functional energy efficiency detector taking into account the principles of adaptability, ergonomics, security and scalability

Conclusion: a software prototype of the system has been implemented. An adaptive interface, authentication modules, account management and structural units have been developed.

KEY WORDS: ENERGY EFFICIENCY; NOTIFIER; HVAC SYSTEMS; USER INTERFACE; MATERIAL DESIGN; FRAMEWORK; MOBILE APPLICATION; MONITORING; INFORMATION SECURITY

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HVAC - Heating, Ventilation, and Air Conditioning

ECP - Energy Performance Certificates

EPI - Energy performance index

POET - Performance, operation, equipment and technology efficiency

RTU - Remote telemetry units

SCADA - Supervisory control and data acquisition

TOU - Time-of-Use

UPS - Uninterruptable power supply

EE - Energy Efficiency

EES - electrical energy storage

ВСТУП

Зростання енергоспоживання у побутовому, комерційному та промисловому секторах висуває нові вимоги до ефективного використання енергоресурсів. У зв'язку з цим дедалі більшої популярності набувають цифрові інструменти, які дозволяють не лише здійснювати моніторинг енергоспоживання, а й формувати рекомендації щодо його оптимізації. Одним із перспективних напрямів є розробка програмних сповіщувачів, що інформують користувачів про неефективні режими роботи обладнання або відхилення від встановлених норм.

Актуальність роботи

У контексті зростаючого попиту на енергоефективність та сталий розвиток, впровадження сучасних технологій управління енергоспоживанням стає пріоритетним завданням для багатьох галузей, зокрема в управлінні системами HVAC. Розробка програмного забезпечення для моніторингу, аналізу та оптимізації енергоспоживання відіграє ключову роль у досягненні екологічних та економічних цілей. Застосування інтелектуальних алгоритмів, таких як генетичні алгоритми, у поєднанні з сучасними платформами розробки дозволяє створювати ефективні системи сповіщення й керування. Це робить тему дослідження надзвичайно актуальною в умовах цифрової трансформації енергетичних процесів.

Сучасні виклики в енергетичній галузі, пов'язані з глобальним потеплінням, вичерпністю традиційних джерел енергії та зростанням тарифів на енергоносії, потребують впровадження новітніх підходів до управління енергоспоживанням. Особливої ваги набуває концепція енергоефективності, яка передбачає не лише модернізацію технічного обладнання, а й застосування цифрових технологій для оптимізації процесів енергокористування. Інформування користувачів про перевищення встановлених порогів споживання, виявлення нераціональних режимів

роботи обладнання та надання рекомендацій у режимі реального часу дозволяють значно знизити витрати та підвищити енергетичну культуру користувачів.

Разом із тим, успішна реалізація подібних рішень неможлива без надійного, зручного та безпечного програмного забезпечення, що враховує як технічні, так і соціальні аспекти взаємодії користувача з системою. Саме тому актуальною є розробка програмного додатку, який дозволяє здійснювати гнучке керування параметрами енергоспоживання, своєчасно інформує про відхилення та має зрозумілий інтерфейс відповідно до сучасних стандартів дизайну.

Усе це зумовлює практичну значущість запропонованої тематики та підтверджує потребу у створенні функціонального сповіщувача енергоефективності, що може бути застосований як у побутовому, так і в комерційному середовищі.

Мета роботи - розробити програмний додаток для моніторингу та сповіщення про ефективність енергоспоживання з урахуванням сучасних вимог до дизайну, безпеки, користувацького досвіду та можливостей адаптивного управління.

Завдання дослідження

1. Проаналізувати сучасні підходи до забезпечення енергоефективності в системах HVAC.
2. Вивчити можливості використання генетичних алгоритмів для оптимізації енергоспоживання.
3. Обґрунтувати вибір фреймворків, бібліотек та платформ для розробки додатку.
4. Розробити адаптивний інтерфейс користувача з урахуванням концепцій Material Design.
5. Реалізувати функціональні модулі керування обліковими записами, безпеки, та структурних одиниць.

Об’єкт дослідження - процеси управління енергоефективністю у системах HVAC на основі програмних засобів.

Предмет дослідження - методи, засоби та технології розробки додатків сповіщення енергоефективності з використанням сучасних фреймворків, алгоритмів оптимізації та підходів до UX/UI дизайну.

Методи дослідження

- Аналіз літературних джерел і технічної документації.
- Метод моделювання функціональності системи.
- Проєктування архітектури додатку.
- Експериментальна реалізація та тестування компонентів ПЗ.
- Методи UX-досліджень (опитування, прототипування).
- Оцінка ефективності за допомогою індикаторів енерговитрат.

Наукова новизна

У роботі запропоновано інтеграцію генетичних алгоритмів керування в мобільний додаток сповіщення енергоефективності, що дозволяє адаптивно оптимізувати витрати енергії в системах HVAC із врахуванням змінних параметрів середовища.

Практичне застосування

Результати дослідження можуть бути використані для впровадження у реальні енергоменеджмент-системи в житловому, комерційному та промисловому секторах, а також у розробці подібних мобільних рішень для інших галузей інтелектуального керування ресурсами.

Бакалаврська робота містить 79 сторінок, 30 рисунків, 3 розділи список використаних джерел із 30 найменуваннями, 1 додаток.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ НОВІТНІХ ТЕХНОЛОГІЙ В ПРОГРАМАХ ЗАБЕЗПЕЧЕННЯ ЕНЕРГОЕФЕКТИВНОСТІ

1.1. Мотивація програмної розробки в межах програми енергоефективності. HVAC системи

Програма енергоефективності — це комплекс планових заходів, спрямованих на зменшення споживання енергії при збереженні або покращенні рівня комфорту, продуктивності та якості послуг. Програми енергоефективності є важливим інструментом для досягнення сталого розвитку, зменшення негативного впливу на довкілля та забезпечення економічної вигоди для всіх учасників.

Вікна, які можна відкривати та закривати, є в багатьох будівлях і приміщеннях. Проблема полягає в тому, що мешканці будівель, які контролюють ці вікна, можуть відкривати їх у невідповідний час. Якщо температура зовні відрізняється від температури всередині, то система опалення, вентиляції та кондиціонування (HVAC - Heating, Ventilation, and Air Conditioning) буде перевантажена, прагнучи зменшити різницю між двома різними температурами. Це додаткове навантаження на систему HVAC призводить до вищих енергетичних витрат, скорочення терміну служби системи HVAC та несправності термостатів. Мешканці того самого будинку також відчуватимуть дискомфорт, оскільки їхня область стане теплішою або холоднішою, коли система HVAC намагається вирівняти зміну температури через нововведене повітря.

Основні компоненти та функції систем HVAC:

- Опалення (Heating). Забезпечує підвищення температури повітря в холодну пору року для створення комфортних умов. Для цього

використовуються різні джерела тепла, такі як котли (газові, електричні, твердопаливні), теплові насоси, електричні обігрівачі тощо.

- Вентиляція (Ventilation). Забезпечує обмін повітря між приміщенням і зовнішнім середовищем або між різними приміщеннями. Вентиляція необхідна для видалення забрудненого повітря (вуглекислий газ, запахи, волога, пил, шкідливі речовини) та забезпечення припливу свіжого повітря. Може бути природною (через вікна, двері, вентиляційні канали) або примусовою (за допомогою вентиляторів та повітроводів).

- Кондиціонування повітря (Air Conditioning). Забезпечує зниження температури повітря та контроль вологості в теплу пору року. Кондиціонери працюють за принципом охолодження повітря та відведення надлишкової вологи.

Сучасні системи HVAC часто інтегрують усі три функції в одному комплексі, забезпечуючи цілорічний контроль мікроклімату. Вони можуть також включати додаткові функції, такі як фільтрація повітря, іонізація, рекуперація тепла тощо.

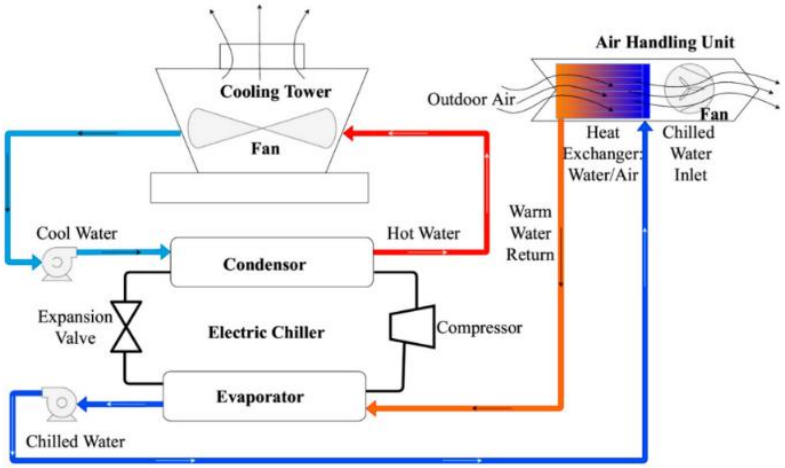

Рисунок 1.1 – Схема HVAC

На рисунку 1.1 зображена схема великомасштабної системи опалення, вентиляції та кондиціонування повітря (HVAC). Схема демонструє цикли охолодження води та повітря, а також основні компоненти системи.

Наведемо опис основних елементів та їх взаємозв'язку:

- Electric Chiller (Електричний чилер). Центральний елемент системи охолодження. Він складається з двох основних теплообмінників:

- Evaporator (Випарник). Холодна вода (Chilled Water) проходить через випарник, де поглинає тепло з повітря або іншого середовища, охолоджуючись ще більше. Холодоагент у випарнику кипить, забираючи тепло.

- Condenser (Конденсатор). Гарячий холодоагент з компресора надходить у конденсатор, де віддає своє тепло охолоджуючій воді (Cool Water) з градирні, конденсуючись у рідину. Гаряча вода (Hot Water) відводиться до градирні.

- Compressor (Компресор). Стискає газоподібний холодоагент, підвищуючи його температуру та тиск перед надходженням у конденсатор.

- Expansion Valve (Розширювальний клапан). Знижує тиск рідкого холодоагенту перед його надходженням у випарник, що призводить до його різкого охолодження.

- Cooling Tower. Використовується для охолодження води, яка забрала тепло в конденсаторі. Гаряча вода (Hot Water) розпилюється у градирні, де охолоджується за рахунок випаровування невеликої кількості води у повітря, що нагнітається вентилятором (Fan). Охолоджена вода (Cool Water) повертається до конденсатора.

- Pumps (Насоси). У схемі зображено два насоси:

Один насос циркулює охолоджену воду (Chilled Water) від випарника до припливно-витяжної установки (Air Handling Unit).

Інший насос циркулює охолоджуючу воду (Cool Water) від градирні до конденсатора та гарячу воду (Hot Water) назад до градирні.

- Air Handling Unit (Припливно-витяжна установка). Пристрій, що обробляє повітря для подачі в приміщення. Він включає:

- Heat Exchangers Water/Air (Теплообмінники Вода/Повітря). Охолоджена вода (Chilled Water Inlet) проходить через теплообмінники, де віддає холод повітрю, що надходить ззовні (Outdoor Air). Тепле вода (Warm Water Return) повертається до чилера.

- Fan (Вентилятор). Забезпечує циркуляцію обробленого (охолодженого) повітря в приміщення.

Розглянемо цикли роботи.

- Цикл охолодження холодоагенту. Холодоагент циркулює через випарник (де поглинає тепло), компресор (де стискається), конденсатор (де віддає тепло) та розширювальний клапан (де знижується його тиск).

- Цикл охолодження води. Вода циркулює між конденсатором (де нагрівається) та градирнею (де охолоджується).

- Цикл охолодження повітря. Охолоджена вода циркулює від чилера до припливно-витяжної установки, де охолоджує повітря, яке потім подається в приміщення.

Схема наочно ілюструє взаємодію різних компонентів у великій системі HVAC, необхідній для забезпечення комфортного клімату у великих будівлях.

Неконтрольована зміна температури — це не єдина проблема. Раптова зміна температури не залишиться непоміченою, але є інші фактори, які залишаться майже непоміченими, наприклад, забрудненість повітря та кількість пилу. Якщо якість повітря погана в певний день або кількість пилу надзвичайно висока, це може бути не відразу очевидно для людини, яка просто відкрила вікно, думаючи, що вона просто пропускає свіже повітря. Додатковий пил і забруднення в приміщенні можуть призвести до того, що персонал буде частіше замінювати фільтри в системі HVAC, а також завдати непотрібні шкоди іншим мешканцям.

Останнім аспектом, що викликає занепокоєння, є питання безпеки та виникнення нерегулярних ситуацій. Вікно, залишене мешканцем відчиненим

без нагляду, може створити потенційну вразливість, забезпечуючи несанкціонований доступ до будівлі, яка в іншому випадку є захищеною. Це може включати як фізичне проникнення сторонніх осіб, так і потрапляння небажаних газоподібних речовин та твердих частинок до внутрішнього середовища будівлі.

1.2. Стратегія керування для систем кондиціонування повітря з використанням генетичних алгоритмів

Системи опалення, вентиляції та кондиціонування повітря (HVAC) складають значну частку енергоспоживання багатьох будівель [1-3]. Дослідники намагалися визначити методологію для зниження такого енергоспоживання HVAC, і серед розроблених методологій генетичні алгоритми (GA) широко використовуються в різних галузях і відомі своєю придатністю для вирішення складних задач оптимізації, особливо коли задіяна велика кількість даних і параметрів. Тому GA (тип методу машинного навчання) можна застосовувати для оптимізації складних конфігурацій систем, таких як будівлі. Проведено значне дослідження для оптимізації теплової продуктивності будівель та зниження енергоспоживання, особливо системи HVAC будівлі. Недавні дослідження показали, що методи оптимізації з використанням GA можуть заощадити енергію в системах HVAC та підвищити їх енергоефективність. Дослідники проводили аналіз змін енергоспоживання систем HVAC залежно від параметрів проектування будівлі та оптимізували проектування системи HVAC на основі моделювання.

Продуктивність будівель та їхніх систем HVAC у реальному часі впливає на такі фактори, як температура та вологість зовнішнього повітря, режими та патерни роботи та інші. Для ефективної експлуатації та ефективного керування системою HVAC будівлі, система HVAC повинна

працювати та керуватися оптимальними керувальними змінними (налаштуваннями) в реальному часі, які відповідають змінам навантаження через зовнішнє середовище. Бажано, щоб таке оптимальне управління та керування можна було реалізувати без додаткових витрат на оновлення системи.

Типові локальні системи керування в системі кондиціонування повітря змінного об'єму (VAV) спираються на керування статичним тиском, температурою подачі повітря та потоком зовнішнього повітря. Ці локальні системи керування впливають на комфорт всередині приміщення та енергоспоживання системою. Відповідне керування можна досягти, адаптуючи внутрішні навантаження та зовнішні умови в реальному часі. Локальне керування в системі VAV традиційно реалізується за допомогою контролерів з використанням значень, отриманих з традиційних стратегій керування, таких як скидання заданої точки, пропорційно-інтегрально-диференційне керування (PID) та модельне передбачувальне керування (MPC).

Системи HVAC складаються з багатьох компонентів і підсистем (наприклад, вентилятори, холодильні машини, насоси, повітряні канали, труби, теплообмінники тощо). У цьому дослідженні офісна будівля була використана як еталонна будівля, в якій моделювалася та симулювалася система HVAC. Оптимальні керувальні змінні, отримані з генетичних алгоритмів (GA), пропонуються для зниження енергоспоживання та оптимізації продуктивності систем HVAC у великомасштабних офісних будівлях. Розраховані оптимальні керувальні змінні/налаштування вводяться в систему кондиціонування повітря змінного об'єму (VAV) з інтервалом один год. GA використовувалися для аналізу змін розрахованих оптимальних керувальних змінних, які потім використовувалися для управління моделлюваною системою. За допомогою цього процесу розраховувалися та

порівнювалися економія енергії з енергоспоживанням при попередніх "нормальних" (у порівнянні з "оптимальними") умовах експлуатації.

Серед стандартних будівель, для оцінки енергії, категорія великих офісних будівель була обрана як тип будівлі для генерування необхідних вхідних даних для цього дослідження. Вхідні значення для еталонної будівлі були змінені, щоб відобразити профіль використання великої офісної будівлі. Для відповідності програмі моделювання дані про погоду були отримані шляхом модифікації даних тестового еталонного року (TRY), які є стандартними погодними даними для району Сеул. У таблиці 1.1 наведено основні граничні умови, використані в моделі для отримання даних.

Таблиця 1.1 - Умови моделювання для еталонної великомасштабної офісної будівлі. HVAC: опалення, вентиляція та кондиціонування повітря

Компонент	**Характеристики**
Дані про погоду та місце розташування	Тестовий еталонний рік (TRY) (широта: 37.57◦N37.57◦N, довгота: 126.97◦E126.97◦E)
Тип будівлі	Великомасштабна офісна будівля
Загальна площа будівлі (m2)(m2)	46,320
Години моделювання (год)	8760
Ізоляція оболонки (m2 K/W)(m2 K/W)	Зовнішня стіна 0.35 , дах 0.213 , зовнішнє вікно 1.5
Співвідношення вікна до стіни (%)	40
Налаштування (◦C)(◦C)	Охолодження 26, опалення 20
Внутрішнє навантаження	Освітлення 10.76(W/m2)10.76(W/m2), люди 18.58(m2/18.58(m2/ особа)), підключення та технологічні процеси 10.76(W/m2)10.76(W/m2)
Розмір HVAC	Автоматичний розрахунок (програмне забезпечення має бути визначене)
Розклад роботи HVAC	7:00−18:00

Будівля та її система HVAC були смоделювані за допомогою EnergyPlus версії 8.9.0. Виходи, включаючи енергоспоживання, були

отримані з EnergyPlus і включають кількість для потоку, температури та тиску в кожній вузлі систем будівлі. Система HVAC, смодельована в цьому дослідженні, має блок обробки повітря, який забезпечує систему VAV для кожної кімнати та фризер, який також може використовуватися як джерело тепла в холодну погоду. Для управління системою VAV потрібно багато керованих налаштувань, починаючи від налаштувань температури нагрівання та охолодження та мінімальної швидкості потоку повітря на рівні зони до мінімального потоку зовнішнього повітря, температури подачі повітря та статичного тиску в повітряних каналах на рівні системи [25]. Серед різних керованих змінних у системі HVAC налаштування температури подачі повітря та статичного тиску в повітряних каналах були спеціально обрані для дослідження як керувальні змінні в системі кондиціонування повітря VAV. На рисунку 1.2 показана типова система HVAC VAV та місця налаштування керування для температури подачі повітря та статичного тиску в повітряних каналах.

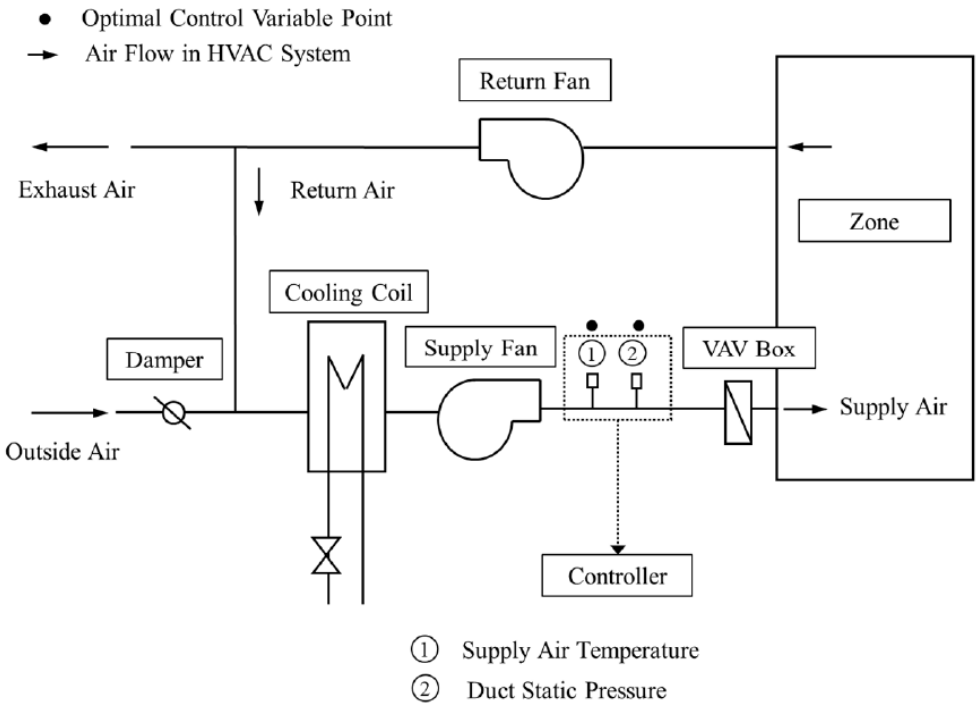

Рисунок 1.2 - Схема типової системи кондиціонування повітря змінного об'єму (VAV) та налаштування керування

У цьому дослідженні GA використовувалися для отримання оптимальних параметрів керування для оптимізованої операції керування. Оптимізаційною функцією є загальне енергоспоживання системи HVAC. Спочатку потрібна модель розрахунку енергії для системи HVAC, щоб реалізувати GA, оптимізаційною функцією якої є енергоспоживання системи HVAC. Чисельна модель для розрахунку енергії системи HVAC посилається на існуючу модель розрахунку та кілька вхідних змінних, включаючи керовані та некеровані змінні.

Вхідні значення, необхідні для розрахунку енергоспоживання системи HVAC, були отримані з виходів моделювання будівлі. Наступним кроком є отримання оптимальних керувальних змінних за допомогою GA, які дають набір оптимальних або потенційних рішень для задачі. Кожне рішення в популяції називається індивідом. Нове покоління індивідів створюється кожного разу; алгоритм оптимізації повторюється для визначення найбільш оптимального рішення. На рисунку 1.3 показана блок-схема, яка описує процес оптимізації для визначення оптимальних налаштувань керування за допомогою GA.

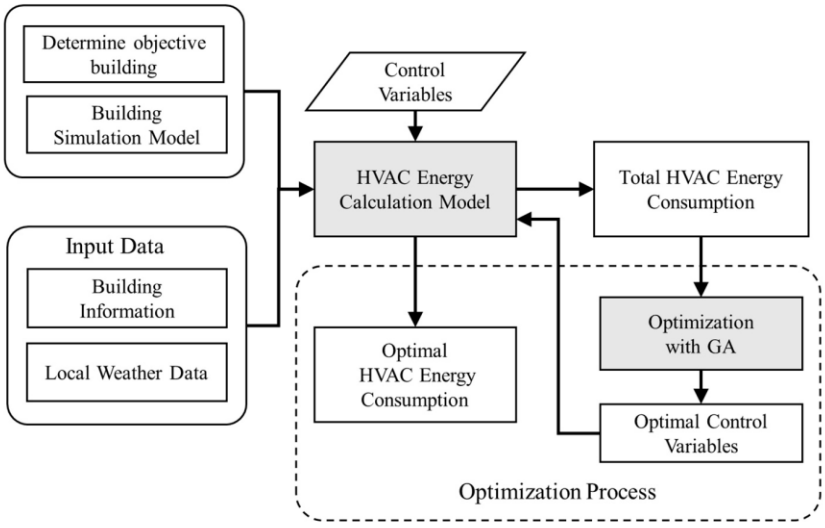

Рисунок 1.3 - Блок-схема моделі аналізу енергії для системи HVAC за допомогою генетичних алгоритмів

GA, використані для моделі розрахунку енергії та оптимізації системи HVAC, були запрограмовані за допомогою MATLAB та відповідного інструментарію в MATLAB. Гарячий сезон (з травня по вересень), був використаний як період аналізу для розрахунку енергоспоживання системи HVAC. Загальне енергоспоживання системи HVAC протягом гарячого сезону є сумою енергоспоживання трьох компонентів (вентилятор, холодильна машина та насос холодної води) у системі HVAC.

Описана оптимальна стратегія керування передбачає використання енергоспоживання як функції оптимізації для генетичного алгоритму (ГА). ГА застосовується для визначення оптимальних значень керуючих параметрів системи змінної витрати повітря (VAV) у режимі реального часу: температури припливного повітря та статичного тиску у повітроводах.

Оцінка ефективності запропонованого методу оптимального керування здійснювалася шляхом аналізу динаміки зміни оптимальних керуючих змінних та інтегральних показників енергозбереження в модельованій системі HVAC типової великомасштабної офісної будівлі (еталонна модель). Встановлено, що в оптимальному режимі керування температура припливного повітря підтримується на рівні нижче 12°C, що є нижчим за проектне значення системи (12.8°C). Аналогічно, статичний тиск у повітроводах оптимізується до значення нижче проектного (474 Па).

При роботі системи за оптимальними параметрами, визначеними ГА (низькі значення температури припливного повітря та статичного тиску), зафіксовано наступні зміни в структурі енергоспоживання: загальне енергоспоживання знизилося на 5,7%; енергоспоживання вентиляційної установки зменшилося на 26,9%; енергоспоживання чилера знизилося на 1,6%; при цьому спостерігалося незначне зростання (на 0,9%) енергоспоживання насоса холодної води. Слід зазначити, що експлуатація системи при низьких температурах припливного повітря та низьких витратах повітря може призвести до виникнення небажаних феноменів, таких як

внутрішня конденсація в компонентах системи або повітроводах, формування зон холодних потоків у приміщеннях та тимчасове погіршення якості внутрішнього повітря. Отже, при оптимізації проектування та виборі режимів експлуатації системи HVAC необхідно враховувати ці потенційні наслідки.

Застосування запропонованого методу є особливо ефективним у конфігураціях систем HVAC, де енергоспоживання вентиляційної установки становить значну частку загального енергоспоживання; у таких випадках досягається суттєвіша економія енергії. Подальші дослідження можуть бути сфокусовані на оптимізації керування джерелом тепло-холодопостачання. Регулювання таких змінних, як температура подачі холодної води, її витрата та холодопродуктивність чилера, відкриває додаткові можливості для підвищення енергоефективності систем HVAC при використанні розширених наборів керуючих параметрів.

1.3. Передумови та цілі виконання проекту

Вирішення проблеми з відкриттям вікна в невідповідний час або забуттям закрити вікно є можливим кількома способами, і ця робота досліджує створення додатку, який допомагає користувачам визначити, чи можна відкрити їхнє вікно.

На момент проведення дослідження та підготовки даної роботи не було виявлено прецедентів існування програмного забезпечення, розповсюджуваного публічно або в рамках приватного використання, що забезпечувало б функціональність реєстрації організаційної структури користувачем, визначення внутрішніх просторів з можливістю конфігурації параметрів мікроклімату та інформування іншого користувача в межах цієї ж організації щодо доцільності відкриття віконних конструкцій на основі заданих критеріїв. Найближчим за функціональним призначенням аналогом

слід вважати метеорологічні додатки, які надають користувачеві інформацію про поточні та прогнозовані погодні умови, що може слугувати опосередкованою основою для прийняття рішення щодо вентиляції приміщення шляхом відкриття вікон.

Створення оригінального додатку з урахуванням інтересів зацікавлених сторін вимагає інтенсивних досліджень та спілкування з клієнтами, щоб гарантувати, що проєкт може бути точно сформований та вчасно завершений.

Метою проєкту є розробка мобільного додатку, який допоможе людям визначити, чи можна відкрити вікно для простору, в якому вони знаходяться, або чи воно повинно залишатися закритим. Набір дослідницьких цілей, які я визначив, наступні:

1. Визначити цільову аудиторію.

2. Розробити список параметрів, які враховуються при розрахунку доцільності відкриття вікна.

3. Визначити, які функції складають функціональний додаток, який вирішує проблему.

4. Дослідити, як користувач буде взаємодіяти з додатком.

Після вирішення вищезазначених цілей було зроблено перехід до вирішення наступного набору цілей розробки:

1. Визначити платформу, на буде розроблено додаток.

2. Вирішити, які фреймворки, бібліотеки або інші технології будуть задіяні в створенні додатку.

3. Знайти найкращий спосіб повідомити користувачеві, чи можна відкрити їхнє вікно або чи воно повинно залишатися закритим.

4. Дослідити можливі канали зв'язку з користувачем та реалізувати найбільш відповідні.

Після успішного розвитку та оцінки всіх функцій додатку було визначено остаточний набір цілей:

1. Розрахувати оціночні витрати на роботу додатку.

2. Уточнити всі функції додатку у співпраці з вимогами зацікавлених сторін.

3. Оцінити різні варіанти для розгортання додатку.

4. Розробити стратегію, щоб інші могли підтримувати додаток.

1.4. Визначення параметрів контролю і необхідних функцій додатку

Після аналізу джерел визначено параметри які мені потрібно враховувати при визначенні доцільності відкриття вікна, а саме це три основні параметри, які необхідно враховувати: температура, вологість та забрудненість повітря (рис. 1.4).

Рисунок 1.4 – Представлення параметрів контролю

Ці параметри найбільше впливають на систему HVAC, оскільки "основні цілі системи опалення, вентиляції та кондиціонування повітря (HVAC) полягають у забезпеченні хорошої якості повітря в приміщенні за допомогою достатньої вентиляції з фільтрацією та забезпечення комфортної температури [1]." Тому ці три змінні мали найбільшу вагу серед потенційних факторів, які я враховував при написанні логіки для розрахунку доцільності відкриття вікна.

Розглянемо необхідні функції додатку. Основна функціональність включає можливість для користувача шукати організацію за її назвою та шукати конкретний простір, у якому знаходиться їхнє вікно, за назвою цього простору. Цей простір може бути або загальним, або конкретним; приклад загальної назви простору — "Лабораторія ІФНТУНГ" для позначення всіх дослідницьких лабораторій в організації; приклад конкретної назви — "Карпатська, 15. Корпус ФАЄ" для позначення коду будівлі. Користувач не зможе шукати, якщо не будуть створені організації та простори, тому додаток також повинен підтримувати спосіб для користувача зареєструвати організацію, створювати простори в організації та керувати ними відповідно.

Природно, аутентифікація користувачів слідує за вимогами функціональності для створення та реєстрації організації, оскільки організацію можна змінювати лише той, хто її створив. Остаточне бажання щодо основної функціональності виникло з первісного опису проєкту, який передбачав, що користувач може зареєструватися для отримання сповіщень щодо конкретного простору, а потім додаток сповістить користувачеві, коли вони можуть відкрити своє вікно. Нарешті, потрібно розгорнути додаток у мобільному режимі, тобто додаток повинен підтримувати мобільний перегляд.

Висновки до розділу

В даному розділі проведено огляд існуючих рішень, що підтвердив впровадження інтелектуальних методів, таких як генетичні алгоритми, дозволяє оптимізувати витрати енергії, забезпечуючи при цьому комфортні мікрокліматичні умови в приміщеннях.

У підрозділі 1.1 висвітлено мотиваційні аспекти створення програмного забезпечення в межах енергоефективних ініціатив, що ґрунтуються на сучасних викликах у сфері сталого розвитку та підвищення

ефективності використання енергоресурсів. Поглиблений розгляд у підрозділі 1.2 показав доцільність застосування стратегій оптимального керування, які базуються на еволюційних підходах, для автоматизованого регулювання параметрів роботи кліматичних систем.

У підрозділах 1.3 та 1.4 сформовано передумови та цілі створення функціонального додатку, що слугуватиме інструментом моніторингу та сповіщення щодо стану енергоефективності. Проведено деталізацію параметрів контролю та окреслено ключові функціональні можливості майбутнього рішення.

Таким чином, результати аналітичного етапу стали підґрунтям для формування технічних вимог до розробки програмного забезпечення, здатного забезпечити інтелектуальне керування енергоспоживанням у сучасних будівлях.

РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ ФРЕЙМВОРКІВ ТА ПЛАТФОРМИ ДЛЯ РОЗРОБКИ ДОДАТКУ СПОВІЩУВАЧА ЕНЕРГОЕФЕКТИВНОСТІ

2.1. Вибір фреймворку та бібліотек для розробки додатку

Після вирішення розробляти веб-додаток, а не мобільний додаток, мені потрібно було визначити, які фреймворки та технології будуть використовуватися для створення додатку. Хоча можливо створити веб-додаток лише з HTML, CSS та JavaScript, більшість розробників схиляються до використання клієнтського JavaScript-фреймворку. Фреймворки JavaScript роблять сучасну веб-розробку швидшою та простішою. Це було б майже неможливо розробити всі вищезазначені функції без фреймворку, і результат був би гіршим.

Більш формальне визначення фреймворку походить від розробників Mozilla, яка зазначає, що "фреймворк — це бібліотека, яка пропонує думки щодо того, як розробляти програмне забезпечення. Ці думки дозволяють передбачувати та однорідність у додатку; передбачуваність дозволяє програмному забезпеченню масштабуватися до надзвичайно великого розміру та залишатися підтримуваним; передбачуваність та підтримуваність є необхідними для здоров'я та довговічності програмного забезпечення [2]."

Мережа проєкту Mozilla, яку співзасновував творець JavaScript Брендан Ейх [3], пропагує використання фреймворків JavaScript, перераховуючи їх як "невід'ємну частину сучасної розробки фронтенду, яка надає розробникам перевірені та протестовані інструменти для створення масштабованих, інтерактивних веб-додатків [4]." Загалом, не було сумнівів щодо використання фреймворку чи ні. Питання полягало в тому, який фреймворк вибрати.

Існує чотири видатні фреймворки JavaScript, серед яких розробник може вибрати: Ember, Angular, Vue та React. Жоден із варіантів не був би

невідповідним для проєкту такого масштабу, але було вирішено використовувати Vue, в основному тому, що у мене був досвід роботи з ним, і тому що мені подобається з ним працювати. Vue також сумісний з Vuetify, фреймворком матеріального дизайну, створеним спеціально для Vue.js. Тому можна використовувати бібліотеку JavaScript для створення додатку швидше, а також з меншими зусиллями, ніж робити це лише з CSS.

Vue.js — це прогресивний JavaScript-фреймворк для створення інтерфейсів користувача та односторінкових додатків (SPA). Він надає розробникам гнучкий інструментарій для побудови сучасних веб-додатків з акцентом на простоту, масштабованість та ефективність.

Наведемо основні характеристики Vue.js:

- Vue.js дозволяє поступово впроваджувати його у проєкти, починаючи з окремих компонентів і розширюючи до повноцінних SPA.

- Інтерфейс розбивається на незалежні, багаторазові компоненти, що полегшує розробку та тестування.

- Завдяки шаблонам Vue.js забезпечує зрозумілий та читабельний код, де структура HTML відображає стан даних.

- Двостороннє зв'язування даних (two-way data binding), тобто зміни в моделі автоматично відображаються у в'ю, і навпаки, що спрощує синхронізацію даних.

- Система реактивності Vue.js відстежує зміни в даних і автоматично оновлює DOM, забезпечуючи високу продуктивність.

Vue.js реалізує архітектурний шаблон MVVM (Model-View-ViewModel):

- Model (Модель). Джерело даних, яке містить бізнес-логіку та стан додатку.

- View (Вигляд). Шаблони, що визначають структуру та вигляд інтерфейсу користувача.

- ViewModel. Об'єкт Vue, який зв'язує модель та в'ю, забезпечуючи реактивність та двостороннє зв'язування даних.

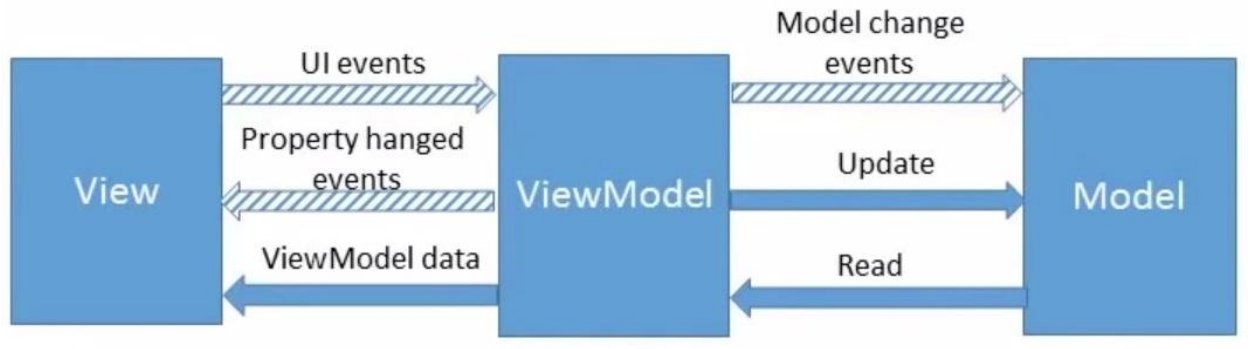

Рисунок 2.1 - Архітектурний шаблон MVVM (Model-View-ViewModel)

Основні принципи MVVM, відображені на схемі:

- Розділення відповідальності (Separation of Concerns), тобто кожен компонент має чітко визначену роль, що полегшує розробку, тестування та підтримку коду.

- Data Binding (Прив'язка даних), механізм, який автоматично синхронізує дані між ViewModel та View. Коли дані у ViewModel змінюються, View автоматично оновлюється, і навпаки. На схемі це відображено двонаправленими стрілками з пунктирною текстурою.

- ViewModel може бути протестована незалежно від View, оскільки вона не має прямого посилання на UI.

Цей підхід сприяє чіткому розділенню відповідальностей, полегшуючи розробку та підтримку додатків.

Vuetify — це бібліотека інтерфейсу користувача Vue [5], яка містить готові компоненти матеріального дизайну. Наявність цих компонентів матеріального дизайну дозволила мені розробляти додаток надзвичайно швидше.

Vuetify — це популярна бібліотека компонентів UI (інтерфейсу користувача) для фреймворку Vue.js. Вона надає розробникам готовий набір багатих, красивих та адаптивних компонентів, які відповідають принципам

Material Design від Google. Vuetify значно спрощує та пришвидшує процес розробки сучасних веб-додатків з інтерактивним та візуально привабливим інтерфейсом.

Дослідимо основні характеристики та переваги бібліотеки Vuetify. Vuetify надає велику кількість готових до використання компонентів, таких як кнопки, форми, навігаційні панелі, картки, діалоги, таблиці, сітки, іконки тощо. Кожен компонент є гнучким та налаштовуваним за допомогою пропсів (props) та слотів (slots).

Бібліотека повністю відповідає специфікаціям Material Design, що забезпечує консистентний та сучасний вигляд додатків, а також покращує їхню зручність використання (UX). Компоненти Vuetify розроблені з урахуванням різних розмірів екранів та орієнтацій (десктопи, планшети, мобільні пристрої), що дозволяє створювати додатки, які чудово виглядають та працюють на будь-якому пристрої.

Vuetify легко встановлюється та інтегрується в проєкти, побудовані на Vue.js. Вона використовує знайомий синтаксис Vue.js, що робить її легкою для вивчення та використання. Vuetify надає потужні можливості для налаштування зовнішнього вигляду компонентів за допомогою тем. Розробники можуть легко змінювати кольори, шрифти, тіні та інші стилі відповідно до брендингу свого проєкту.

Vuetify підтримує відображення тексту справа наліво, що є важливим для мов, які використовують цей напрямок письма. Компоненти Vuetify розробляються з урахуванням вимог доступності (WCAG), що робить створені додатки більш інклюзивними для користувачів з обмеженими можливостями. Бібліотека оптимізована для забезпечення високої продуктивності веб-додатків. Vuetify сумісна з серверним рендерингом Vue.js, що може покращити SEO та час першого завантаження сторінки.

Основні переваги використання Vuetify:

- Забезпечує єдиний та професійний вигляд усіх елементів інтерфейсу.

- Готові компоненти значно скорочують час, необхідний для створення інтерфейсу користувача.

- Компоненти Vuetify добре протестовані та відповідають сучасним стандартам веб-розробки.

- Material Design принципи роблять інтерфейс інтуїтивно зрозумілим для користувачів.

Завдяки своїм численним перевагам та широкому набору компонентів, Vuetify є одним з найпопулярніших виборів для розробників, які використовують Vue.js та прагнуть швидко створювати красиві та функціональні веб-додатки.

2.2. Вибір платформи для розробки додатку

Було вирішено, що додаток повинен підтримувати мобільні пристрої. Це дозволяє вибрати між розробкою веб-додатку з підтримкою мобільного перегляду або окремого мобільного додатку. Було вирішено розробляти веб-додаток, а не мобільний додаток. Веб-додаток пропонує кілька переваг порівняно з мобільним додатком:

- Користувач може взаємодіяти з веб-додатком на будь-якому пристрої з браузером;

- Користувачу не потрібно завантажувати або встановлювати веб-додаток;

- Користувачу не потрібно оновлювати веб-додаток;

- Веб-додаток буде дешевшим, оскільки потрібно платити за публікацію додатків.

Мені потрібно розробляти веб-додаток лише для однієї платформи, а не для багатьох мобільних платформ, які існують. Хоча мені потрібно було розробляти веб-додаток для сумісності з різними браузерами, тобто він має ідеально відображатися у усіх відомих браузерах.

Остаточний елемент технології, який було використано — це Google Firebase, платформа з широким спектром обчислювальних та розробницьких інструментів. Основна послуга, яку було використано — це їхня повністю керована серверна інфраструктура. Мені потрібно було зберігати організації, простори, налаштування облікових записів та сповіщень, які створюють користувачі, а також API-ключі, які використано. Google Firebase пропонує послугу під назвою Cloud Firestore, яка є "безсерверною документо-орієнтованою базою даних, яка легко масштабується для задоволення будь-якого попиту без обслуговування [6]."

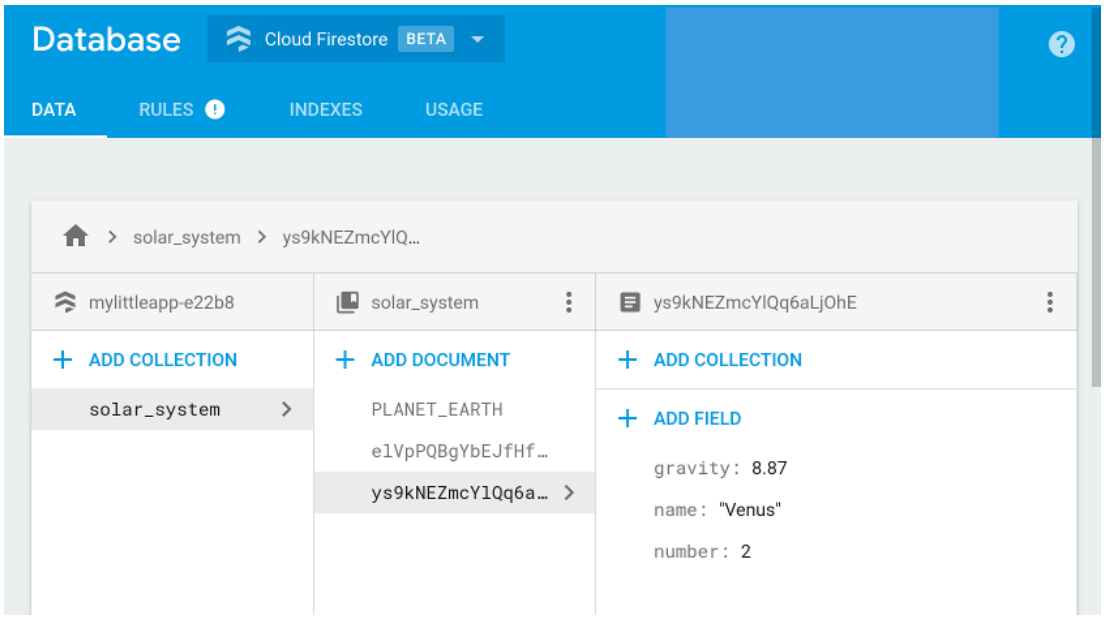

Рисунок 2.2 – Інтерфейс Cloud Firestore

Cloud Firestore — це гнучка, масштабована NoSQL-база даних, що розміщується в хмарі, від Google Firebase. Вона призначена для зберігання та синхронізації даних для мобільних, веб- та серверних розробок. Cloud Firestore є наступним поколінням Cloud Datastore і має ряд покращень, зокрема підтримку запитів у реальному часі та кращу структуру даних.

Розглянемо ключові характеристики та особливості Cloud Firestore.

Cloud Firestore є документо-орієнтованою базою даних NoSQL. Дані зберігаються як документи, які є наборами пар ключ-значення. Ці документи групуються в колекції. Документи можуть містити різні типи даних, включаючи рядки, числа, булеві значення, масиви, об'єкти та двійкові дані. Документи також можуть містити вкладені об'єкти та масиви.

Cloud Firestore підтримує потужні запити, які можуть отримувати дані в реальному часі. Клієнтські додатки можуть підписуватися на зміни в даних і отримувати оновлення автоматично при їх виникненні. Cloud Firestore автоматично масштабується для обробки зростаючого обсягу даних та трафіку без необхідності ручного керування інфраструктурою.

Google Cloud забезпечує високу доступність та надійність Cloud Firestore, автоматично реплікуючи дані в кількох зонах. Cloud Firestore тісно інтегрована з іншими сервісами Firebase (наприклад, Authentication, Cloud Functions) та Google Cloud Platform. Клієнтські SDK Firebase забезпечують кешування даних на пристрої та синхронізацію змін після відновлення підключення до мережі. Cloud Firestore підтримує транзакції, що відповідають вимогам ACID (Atomicity, Consistency, Isolation, Durability), для забезпечення цілісності даних при складних операціях запису.

Основні концепції:

- Колекції (Collections). Групи документів. Колекції використовуються для організації даних. Наприклад, колекція може містити інформацію про всіх користувачів або всі продукти.

- Документи (Documents). Окремі записи в колекції. Документ містить дані у вигляді пар ключ-значення.

- Поля (Fields). Пари ключ-значення в документі.

- Підколекції (Subcollections). Колекції, вкладені в документи. Підколекції допомагають організувати ієрархічні дані.

Переваги використання Cloud Firestore полягають в тому, що клієнтські SDK Firebase надають простий та інтуїтивно зрозумілий API для роботи з

базою даних, а запити в реальному часі спрощують створення інтерактивних та динамічних додатків. Також немає необхідності керувати серверами, що дозволяє розробникам зосередитися на написанні коду.

Cloud Firestore є потужним інструментом для розробників, які шукають масштабовану, гнучку та просту у використанні хмарну NoSQL-базу даних з підтримкою запитів у реальному часі. Її інтеграція з екосистемою Firebase та Google Cloud робить її привабливим вибором для широкого спектру завдань розробки.

Керування серверною частиною — це непроста задача, і тому допомога Google у керуванні нею для мене прискорила розробку ще більше. З фреймворками, які було обрано, мені потрібно було визначити послугу для управління залежностями, які ці фреймворки стануть для мого проєкту, і залежності, які ці фреймворки використовують.

Щоб гарантувати, що залежності, задіяні в моєму веб-додатку, залишаються оновленими та сумісними з версією, яку від них очікують, то було використано менеджер пакетів node.js.

Node.js — це "менеджер пакетів для Node.js. Node.js був створений у 2009 році як проект з відкритим вихідним кодом, щоб допомогти розробникам JavaScript легко ділитися пакуваними модулями коду [7]." Vue, Vuetify та Google Firebase існують як модулі node, тому з менеджером пакетів node я можу об'єднати їх усіх у свій веб-додаток для підтримки їх поруч з будь-якими іншими залежностями, які можна додати.

Node.js — це середовище виконання JavaScript, яке дозволяє запускати JavaScript-код поза браузером. Воно побудоване на рушії V8 від Google (той самий, що використовується в Chrome), і призначене для створення швидких, масштабованих серверних додатків.

Основні характеристики Node.js:

1. Використовується цикл подій (event loop), який реагує на асинхронні події (наприклад, запити до бази даних чи HTTP-запити).

2. Node.js працює в одному потоці, але використовує неблокуючу модель вводу/виводу, що дозволяє обробляти тисячі одночасних з'єднань без створення нових потоків.

3. Менеджер пакетів npm (Node Package Manager) — це найбільший реєстр JavaScript-бібліотек і модулів, що значно полегшує розробку.

4. Можливість створювати як серверні, так і інструментальні скрипти, оскільки Node.js часто використовують для розробки веб-серверів, REST API, чатів, WebSocket-додатків, а також для створення інструментів командного рядка.

Приклад простого HTTP-сервера показано в лістингу 2.1.

Лістинг 2.1. Код простого HTTP-сервера

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

const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Привіт з Node.js!');
});

server.listen(3000, () => {
  console.log('Сервер працює на порту 3000');
});
```

Застосування Node.js:

- Реалізація веб-серверів та API
- Чат-додатки в реальному часі (наприклад, із використанням WebSocket)
- Системи обробки потокових даних
- Інструменти автоматизації (наприклад, Gulp, Webpack)
- Серверна частина для SPA (React, Angular, Vue)

Архітектура Node.js побудована на однопоточній подієво-орієнтованій моделі з неблокуючим ввід/виводом (I/O). Це дозволяє ефективно обробляти

велику кількість одночасних з'єднань без створення нових потоків для кожного запиту.

Розглянемо основні компоненти архітектури Node.js:

- V8 Engine. Використовується для виконання JavaScript-коду. Цей рушій, розроблений Google, компілює JavaScript у машинний код, забезпечуючи високу продуктивність.

- libuv. Бібліотека, написана на C, яка забезпечує кросплатформену абстракцію для асинхронного вводу/виводу. libuv реалізує цикл подій (event loop) і керує пулом потоків для обробки операцій, які не можуть бути виконані асинхронно на рівні ядра ОС.

- Цикл подій (Event Loop). Це механізм, який дозволяє Node.js обробляти численні запити без блокування. Цикл подій постійно перевіряє чергу подій і виконує відповідні зворотні виклики (callback functions).

- Пул потоків (Thread Pool). Для деяких операцій, таких як файловий ввід/вивід або криптографічні обчислення, Node.js використовує пул потоків, щоб виконувати ці завдання у фоновому режимі, не блокуючи основний потік.

- Node.js API. Надає набір модулів для роботи з файловою системою, мережею, буферами та іншими ресурсами. Ці API побудовані на основі libuv і забезпечують асинхронну взаємодію з системними ресурсами.

Коли надходить запит, Node.js додає його до черги подій. Цикл подій обробляє кожен запит, викликаючи відповідні зворотні виклики. Якщо запит вимагає операції, яка може бути виконана асинхронно (наприклад, читання файлу), вона передається до пулу потоків. Після завершення операції результат повертається до циклу подій, який виконує відповідний зворотний виклик.

На рисунку 2.3 зображена архітектура Node.js. Діаграма ілюструє неблокуючу, подієво-орієнтовану модель, яку використовує Node.js для обробки запитів.

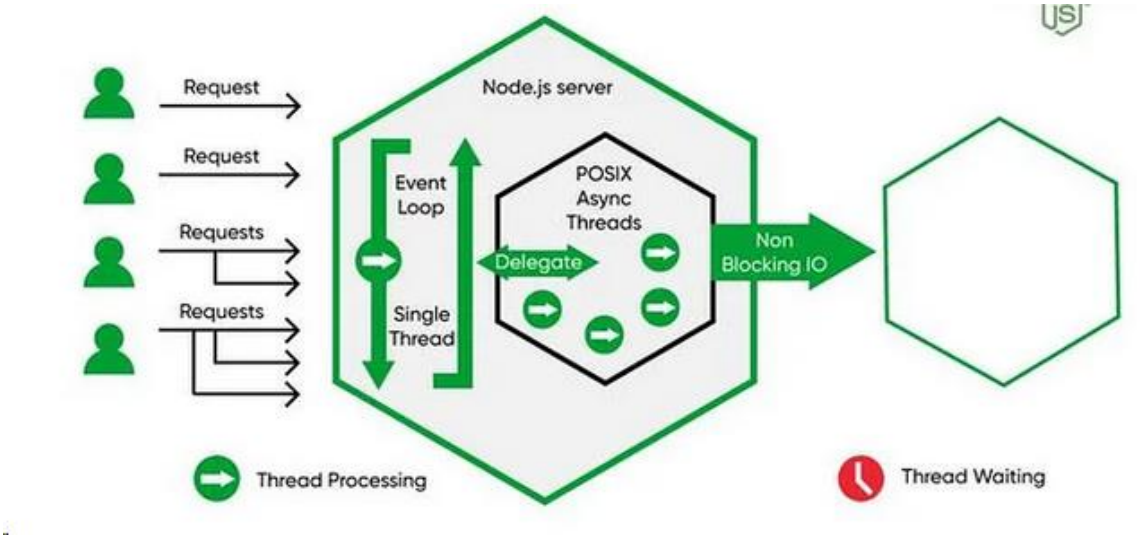

Рисунок 2.3 – Архітектура Node.js

Основні компоненти архітектури, представлені на схемі:

- Клієнти (ліворуч), що генерують запити (Requests) до Node.js сервера. Один клієнт надсилає одиничний запит, а інший — множинні запити.

Node.js Server, який є однопотоковим (Single Thread), але здатен обробляти безліч одночасних запитів завдяки неблокуючому вводу-виводу та циклу подій.

Event Loop (Цикл подій) рображений у вигляді зеленої стрілки, що циркулює навколо однопотокового процесу. Цикл подій відповідає за отримання нових запитів та делегування їх на обробку, а також за обробку результатів асинхронних операцій.

Single Thread (Один потік) позначає основний потік виконання JavaScript коду в Node.js.

POSIX Async Threads (Асинхронні потоки POSIX) представлені у вигляді меншого шестикутника всередині сервера Node.js, що містить кілька зелених кіл, які символізують асинхронні потоки. Node.js використовує пул асинхронних потоків для виконання блокуючих операцій вводу-виводу (наприклад, операції з файловою системою, мережеві запити, операції з базами даних) без блокування основного однопотокового процесу.

Delegate (Делегування), тобто зелена стрілка показує, як запити, що потребують блокуючих операцій, делегуються з основного потоку в пул асинхронних потоків.

Non Blocking IO (Неблокуючий ввід-вивід) – велика зелена стрілка, що виходить з пулу асинхронних потоків та прямує до порожнього шестикутника праворуч (який може символізувати зовнішні ресурси або зворотні виклики). Після завершення асинхронної операції результат передається назад у цикл подій через механізм неблокуючого вводу-виводу.

- Thread Processing (Обробка потоку) - Зелена стрілка внизу ліворуч ілюструє процес обробки завдань в асинхронних потоках.

Thread Waiting (Очікування потоку) - Червоний символ у вигляді годинника внизу праворуч позначає стан очікування потоку, коли основний потік не блокується, чекаючи завершення асинхронної операції.

Отже, архітектура Node.js, як показано на рисунку 2.3, є однопотоковою для виконання JavaScript коду, але використовує неблокуючий ввід-вивід та цикл подій для ефективної обробки великої кількості одночасних запитів. Коли Node.js отримує запит, який потребує блокуючої операції, він делегує цю операцію в пул асинхронних потоків. Основний потік продовжує обробляти інші запити. Після завершення асинхронної операції результат передається назад у цикл подій, який потім обробляє відповідний зворотний виклик. Ця неблокуюча, подієво-орієнтована модель робить Node.js особливо ефективним для розробки мережевих застосунків з високою пропускною здатністю та низькою затримкою, таких як веб-сервери та API.

Разом ці три елементи технології, які було обрано, дозволили мені розробляти додаток у багато разів швидше та краще, ніж якби було вирішено відмовитися від залежностей повністю, так що це було б неможливо розробити його без них, враховуючи обмежений термін, що був на програмну розробку.

2.3. Сутність концепції “Material Design”

Material Design — це візуальна мова та система компонентів, розроблена Google, яка прагне поєднати принципи класичного дизайну з інноваціями науки і техніки. Її ключові аспекти включають використання метафори матеріалу, що імітує властивості фізичного світу (такі як тіні, світло, поверхні), сміливу та інтенсивну графіку, а також осмислену анімацію, що надає сенс взаємодії. Material Design фокусується на створенні інтуїтивно зрозумілих, візуально привабливих та послідовних інтерфейсів користувача на різних пристроях.

Vuetify, зі свого боку, є популярним фреймворком компонентів для Vue.js, який глибоко інтегрує принципи Material Design. Сутність Material Design у Vuetify полягає в наданні розробникам готового набору високоякісних, адаптивних та настроюваних компонентів інтерфейсу користувача, які відповідають специфікаціям Material Design.

Розглянемо ключові прояви сутності Material Design у Vuetify:

- Vuetify надає компоненти (кнопки, картки, поля вводу, навігаційні панелі тощо), які візуально імітують фізичні матеріали, використовуючи тіні для позначення висоти та важливості, а також реагуючи на дії користувача за допомогою продуманої анімації та візуальних ефектів (наприклад, ефект брижів при натисканні).

- Фреймворк втілює принципи сміливої графіки через типографіку, сітки, простір та використання кольору згідно з рекомендаціями Material Design, допомагаючи створити чітку ієрархію та фокус.

- Vuetify пропонує широкий спектр попередньо розроблених компонентів, що значно прискорює розробку, оскільки не потрібно створювати елементи інтерфейсу з нуля, водночас забезпечуючи їх відповідність стандартам Material Design.

- Компоненти Vuetify розроблені з урахуванням адаптивності, що дозволяє створювати інтерфейси, які чудово виглядають та функціонують на різних розмірах екранів та пристроях, що є важливою частиною філософії Material Design.

- Хоча Vuetify базується на Material Design, він надає широкі можливості для налаштування тем, кольорів, типографіки та окремих властивостей компонентів, дозволяючи адаптувати зовнішній вигляд до потреб конкретного проєкту, не відходячи від основних принципів Material Design.

- Vuetify приділяє значну увагу доступності, включаючи функції та атрибути, що допомагають створювати інклюзивні інтерфейси відповідно до рекомендацій Material Design щодо доступності.

Таким чином, сутність Material Design у Vuetify полягає в наданні потужного та зручного інструментарію для розробників на Vue.js, який дозволяє легко створювати сучасні, естетичні та функціональні користувацькі інтерфейси, повністю інтегруючи візуальну мову та принципи Google Material Design. Це спрощує процес розробки, забезпечує послідовність дизайну та покращує загальний користувацький досвід.

Висновки до розділу

В даному розділі розглянуто ключові інструменти, технології та підходи, необхідні для ефективної реалізації програмного додатку сповіщувача енергоефективності. Було враховано критерії сумісності з мобільними платформами, доступності документації, активності спільноти розробників, а також підтримки інтеграції з IoT-пристроями.

Здійснено порівняльну оцінку популярних платформ для розробки — таких як Android Studio, Flutter та React Native — з метою визначення найбільш придатної для цільового середовища розгортання. Враховано

потребу в кросплатформеності, продуктивності, а також відповідності вимогам до інтерфейсу користувача та зручності сповіщення.

Підрозділ 2.3 було присвячено дослідженню концепції “Material Design” як основи візуального та інтерфейсного оформлення додатку. Її застосування дозволяє забезпечити зрозумілий, інтуїтивно доступний та естетично привабливий користувацький досвід, що є важливим чинником підвищення ефективності взаємодії користувача з додатком.

Загалом, результати цього розділу сформували технічну основу для практичної реалізації додатку, забезпечивши обґрунтований вибір інструментів, що відповідають функціональним і візуальним вимогам системи моніторингу енергоефективності.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛЬНОГО СПОВІЩУВАЧА ЕНЕРГОЕФЕКТИВНОСТІ

3.1. Розробка адаптивних представлень інтерфейсу користувача

У сфері розробки програмного забезпечення підходи до процесу створення додатків суттєво варіюються залежно від індивідуальної філософії кожного розробника. Відсутність універсальної методології, яка б була однаково придатною для усіх проєктів, зумовлює необхідність адаптації підходів відповідно до контексту конкретного завдання. Власна стратегія автора полягає в наданні пріоритету створенню високоякісної логіки для функціональності, яка з великою ймовірністю залишатиметься складовою додатку протягом усього його життєвого циклу. Такий підхід сприяє підвищенню якості програмного продукту, полегшує його супровід третіми сторонами, а також забезпечує чистоту, читабельність і масштабованість коду.

У випадках, коли стабільність функціоналу викликає сумніви, доцільним є застосування швидкого прототипування. Це дозволяє більш глибоко дослідити досвід користувача та, за потреби, оперативно внести зміни до концепції. Такий підхід дає змогу досягти оптимального балансу між швидкістю реалізації та якістю, за умови подальшого рефакторингу прототипного коду у разі наявності відповідного часу.

У процесі розробки веб-додатку початково використовувалося середовище розробки Visual Studio Code. Однак, згодом відбулося перемикання на спеціалізовану інтегровану середу розробки JetBrains WebStorm, яка забезпечила розширені можливості для роботи з JavaScript. Управління залежностями здійснювалося за допомогою інструментарію Node.js, а контроль версій реалізовано через систему Git із розміщенням репозиторію на платформі GitHub, що також дозволяло підтримувати

резервні копії проєкту. З метою дотримання професійних стандартів написання коду, було впроваджено інструмент статичного аналізу ESLint, налаштований відповідно до стилістичних рекомендацій. Це дозволило не лише забезпечити технічну коректність коду, а й сприяти підтримці високого рівня його стилістичної узгодженості.

Для розробки графічного інтерфейсу користувача було використано бібліотеку Vuetify, що являє собою комплексний набір попередньо розроблених компонентів, які відповідають принципам Material Design. Vuetify надає розробникам широкий спектр інструментарію для проектування та побудови інтерактивних інтерфейсів, включаючи різноманітні елементи керування (від базових кнопок до складних таблиць даних), а також засоби стилізації та анімації. Застосування готових компонентів та стилів дозволило здійснити швидке прототипування альтернативних макетів з метою оцінки ергономічних характеристик та створити візуально сучасний інтерфейс. Враховуючи часові обмеження проєкту, детальна кастомізація теми оформлення не була пріоритетною; проте, було здійснено цілеспрямовану модифікацію кольору панелі інструментів, обрано червоний відтінок.

Рисунок 3.1 – Панель навігації

Хоча детальна кастомізація візуальної теми не була першочерговим завданням, значна увага була приділена розробці адаптивної структури інтерфейсу. Бібліотека Vuetify надає розробникам ряд інструментів для керування компонуванням елементів на сторінці. З метою оптимізації користувацького досвіду на мобільних пристроях, було прийнято рішення розмістити основний вміст у центральній частині екрана. Крім того, була реалізована програмна логіка для динамічної зміни розмірів компонентів та

адаптації макету сторінки залежно від розмірних характеристик екрана мобільного пристрою. При розробці мобільної версії інтерфейсу враховувалися діапазони розмірів viewport, що охоплюють як найменші (320 x 568 пікселів для Apple iPhone 5), так і найбільші (428 x 926 пікселів для Apple iPhone 12 Pro Max) моделі iPhone, згідно з довідковим джерелом. Такий підхід забезпечує коректне відображення інтерфейсу на широкому спектрі мобільних пристроїв.

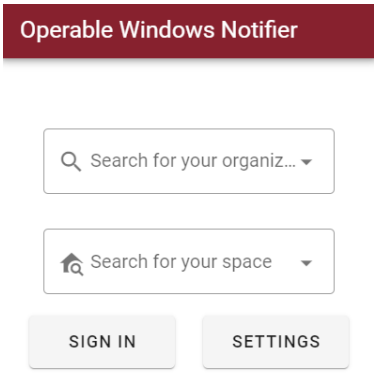

Рисунок 3.2 - Попереднє відображення з розмірами viewport 320 x 568

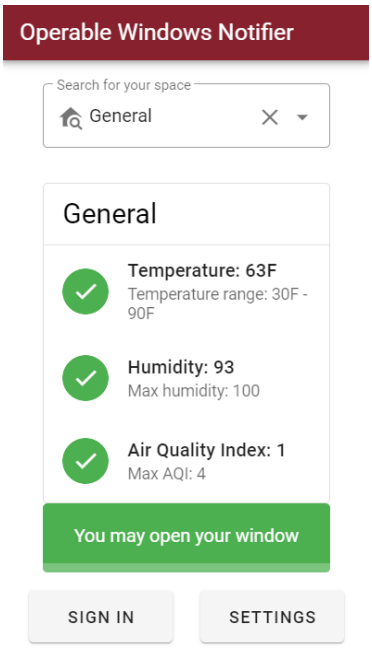

Рисунок 3.3 - Пост-відображення з розмірами viewport 320 x 568

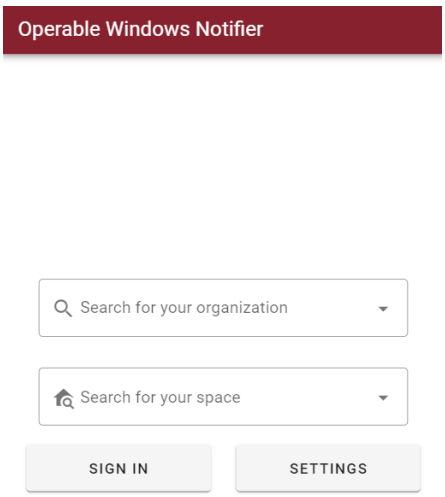

Рисунок 3.4 - Попереднє відображення з розмірами viewport 428 x 926

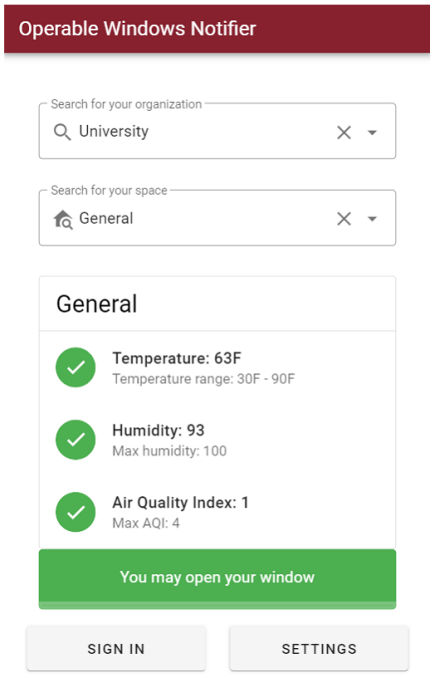

Рисунок 3.5 - Пост-відображення з розмірами viewport 428 x 926

Хоча цей додаток був розроблений з урахуванням мобільного вигляду, макети були налаштовані з урахуванням усіх розмірів пристроїв від iPhone SE до повнорозмірного настільного монітора.

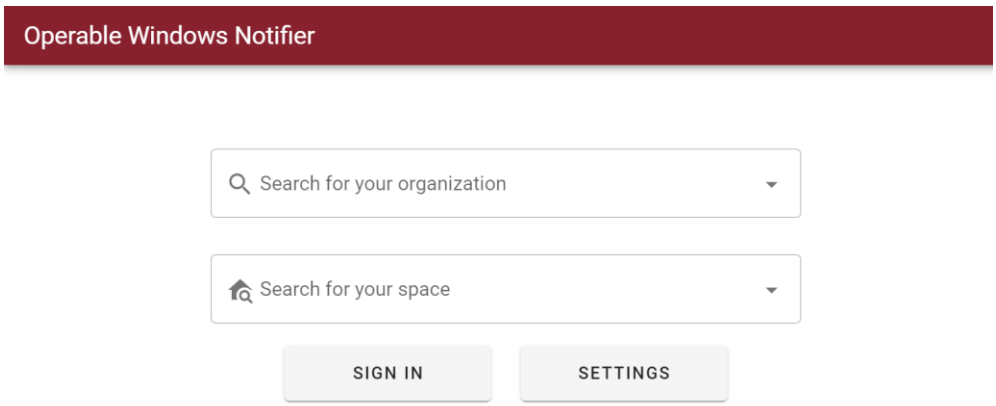

Рисунок 3.6 – Попередній перегляд на екрані монітора

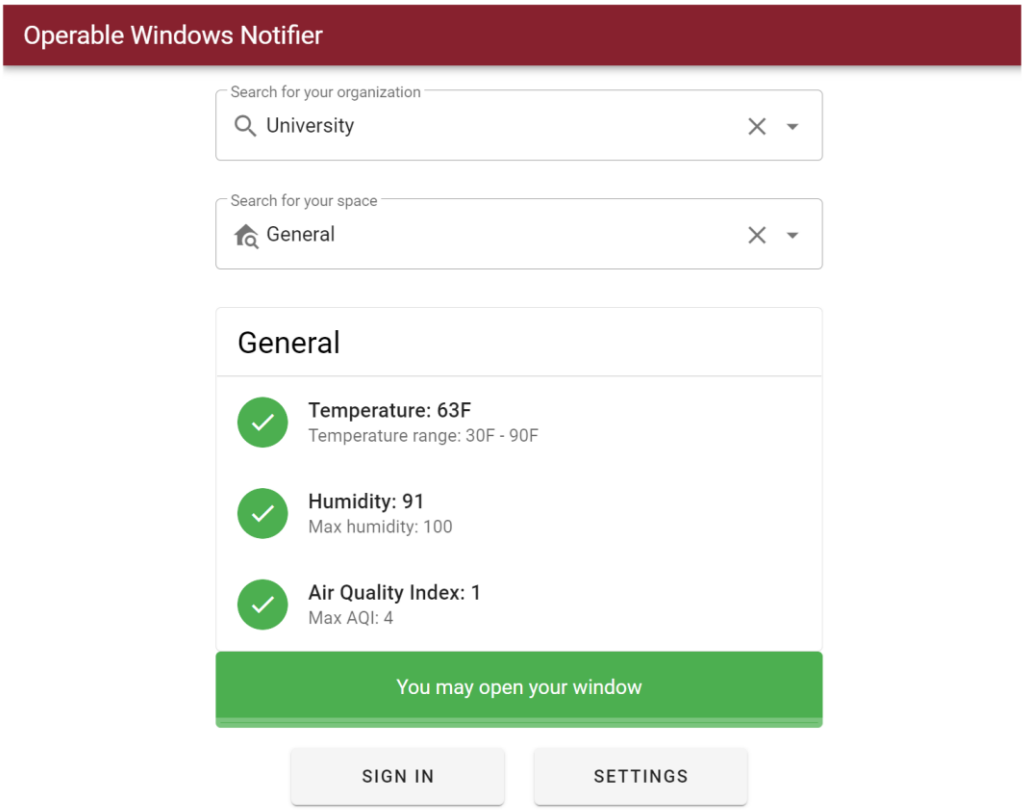

Рисунок 3.7 – Пост-відображення на екрані монітора

У процесі розробки було успішно імплементовано адаптивні представлення інтерфейсу, що забезпечують коректне відображення функціональності на широкому спектрі роздільних здатностей екранів. Проте, аналіз ергономічних характеристик виявив потенційні складнощі у взаємодії з додатком для користувачів, що використовують дисплеї з

мінімальними фізичними розмірами. Для оптимізації досвіду даної категорії користувачів рекомендується впровадження механізму динамічного масштабування типографіки при досягненні певних порогових значень розміру вікна перегляду. Подальша ітеративна оптимізація дизайну інтерфейсу передбачає збір та аналіз зворотного зв'язку від кінцевих користувачів після розгортання програмного забезпечення.

3.2. Досвід користувача

Концептуальна модель "медового коміра досвіду користувача", запропонована П. Морвіллем, є цінним інструментом для систематизації та аналізу ключових аспектів User Experience [9].

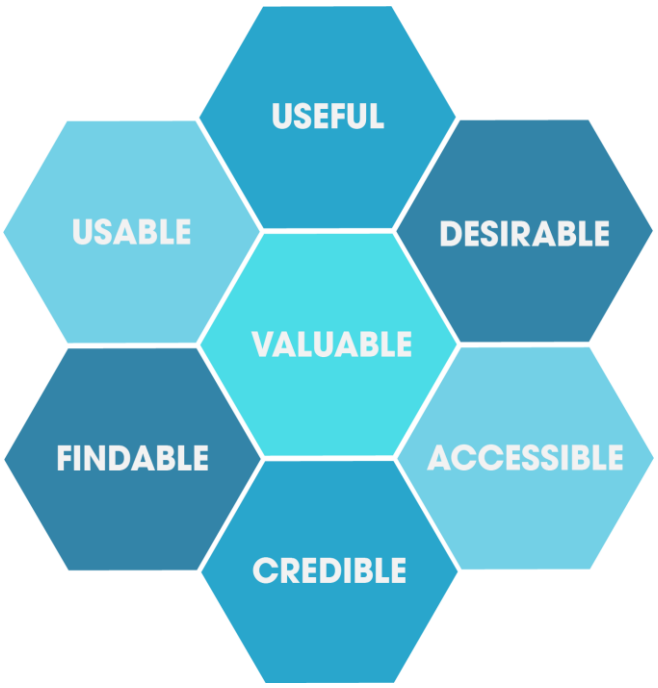

Рисунок 3.8 - Медовий комір досвіду користувача (Honeycomb of User Experience)

У ході проектування та розробки даного застосунку було враховано сім компонентів, що складають зазначену модель.

3.2.1. Ергономічні аспекти: зручність, знаходження та доступність

При оцінці ергономічних властивостей розробленого застосунку було застосовано евристичні принципи зручності, сформульовані в Стенфордській парадигмі [10]. Перелік зазначених принципів включає наступні положення:

1. Контроль користувача. Функціональність застосунку розроблена таким чином, щоб ініціювання навігаційних переходів здійснювалося виключно за явною дією користувача. Автоматичне перенаправлення передбачено лише у випадках спроби доступу до неавторизованого контенту (наприклад, перехід до інтерфейсу адміністрування організації за відсутності відповідних прав доступу). Таким чином, користувач зберігає повний контроль над траєкторією навігації.

2. Розпізнавання проти згадування. З метою мінімізації когнітивного навантаження на оперативну пам'ять користувача реалізовано механізм збереження параметрів пошукових запитів. Дана функція усуває необхідність відтворення в пам'яті точної номенклатури об'єктів пошуку (наприклад, назви приміщення або організації), оскільки збережені параметри доступні для повторного використання. Вимога до згадування обмежується лише процедурою автентифікації користувача.

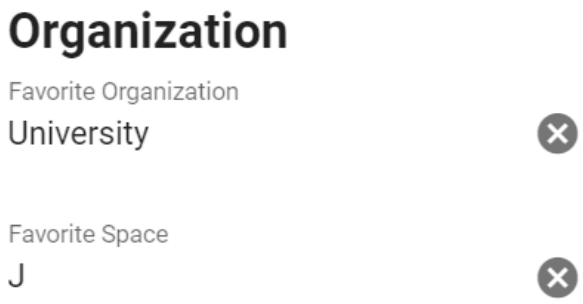

Рисунок 3.9 - Збережена організація та приміщення

3. Ментальна модель. Для забезпечення когнітивної відповідності інтерфейсу моделі сприйняття користувача, було реалізовано послідовний

процес пошуку, що передбачає спочатку ідентифікацію організації, а потім конкретного приміщення в її межах. Такий підхід дозволяє уникнути перевантаження користувача надмірним та неструктурованим переліком потенційних локацій для вибору.

4. Чіткість. Комунікація з користувачем здійснюється з використанням максимально лаконічної та однозначної лексики. Навігаційні елементи інтерфейсу представлені одно- або двослівними позначеннями (наприклад, "налаштування", "вихід", "скинути пароль"). Додатково, поля введення супроводжуються контекстними підказками (placeholder text), що інформують користувача про очікуваний формат даних.

5. Естетична цілісність. Застосування повторно використовуваних компонентів Material Design Vuetify забезпечило уніфікацію візуального стилю застосунку. Інтерфейс характеризується мінімалістичним дизайном та центрованим розташуванням елементів на всіх сторінках, що сприяє візуальній узгодженості.

6. Простота. Як було зазначено в контексті "Чіткості" та "Естетичної цілісності", інтерфейс користувача відрізняється лаконічністю та чіткістю комунікації.

7. Передбачуваність. Функціональність інтерактивних елементів (кнопок) відповідає очікуваній користувачем поведінці. Перед виконанням критичних дій, таких як видалення облікового запису, користувачеві надається попереджувальне повідомлення, що забезпечує повне розуміння наслідків його дій (рис. 3.10).

8. Інтерпретація. Реалізовано підтримку стандартних механізмів браузерів та менеджерів паролів для автоматичного розпізнавання полів введення облікових даних та їх автозаповнення. Таким чином, інтерфейс адаптується до намірів користувача.

9. Точність. У всіх формах та текстових полях застосовано атрибути типу введення (input type), що дозволяє обмежити формат введених даних та

запобігти некоректному введенню (наприклад, введення емодзі в поле для оновлення номера телефону).

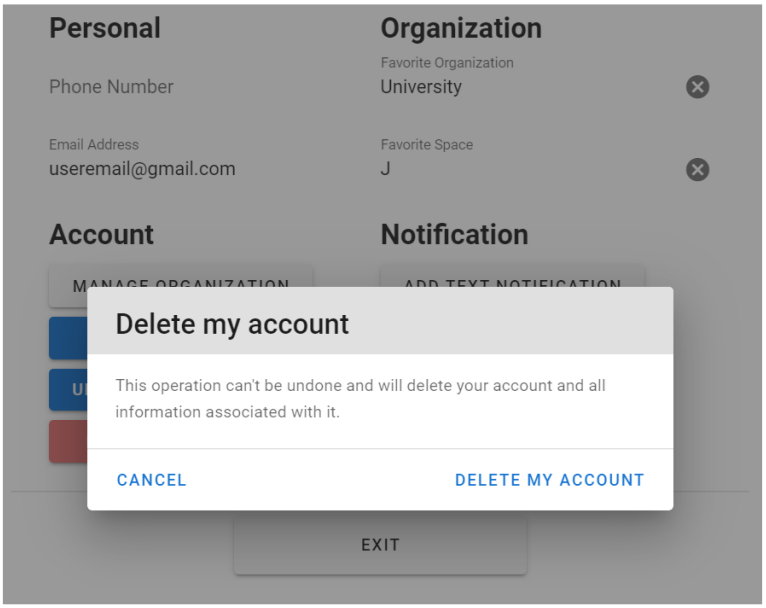

Рисунок 3.10 - Діалогове вікно видалення облікового запису

10. Обробка помилок та запобігання. Розроблено кастомні правила валідації для всіх форм застосунку, що запобігають введенню невідповідних даних та надають користувачеві інформативний зворотний зв'язок щодо причин некоректності введених значень. У випадку виявлення недійсних полів, система інформує користувача про необхідність їх виправлення для успішного надсилання форми (рис. 3.11).

11. Кастомізація. Обмежена складність даних та однорідність типів інформації знизили емпірично визначену доцільність або необхідність реалізації функціоналу деталізованої користувацької кастомізації інтерфейсу.

12. Швидкі клавіші/Навігаційні прискорювачі. Забезпечено прямий навігаційний шлях (наприклад, "швидка клавіша" або постійне посилання) до головної сторінки, доступний з будь-якого іншого розділу додатку.

13. Послідовність та Стабільність Інтерфейсу. Панель інструментів (тулбар), що містить ідентифікатор поточного розділу (назву сторінки), має

фіксоване та незмінне розташування у верхній частині вікна перегляду на всіх екранах додатку, забезпечуючи структурну послідовність.

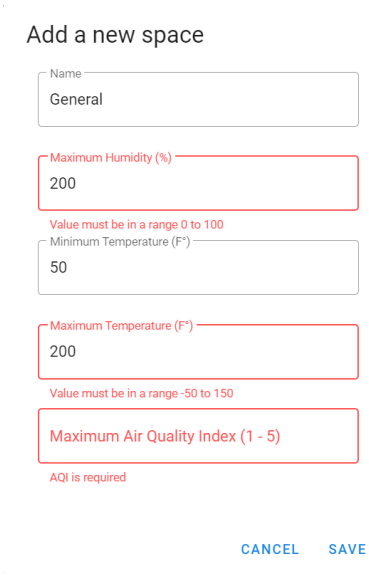

Рисунок 3.11 - Недійсна форма додавання нового приміщення

14. Підтримка Користувачів. Модель та імплементація системи підтримки користувачів (User Support) визначається виключно вимогами та рішеннями замовника (клієнта проєкту).

15. Ефективність та Надійність Взаємодії. Функціонал додатку спроєктовано таким чином, що користувачі можуть виконувати необхідні операції без необхідності застосування неочевидних "обхідних шляхів" або використання субоптимальних методів. Очікуваний результат дії досягається детерміновано при кожній коректній взаємодії з інтерфейсом.

16. Толерантність до Помилок (Forgiveness) та Система Зворотного Зв'язку (Feedback): Реалізовано механізми, спрямовані на підвищення стійкості системи до помилок користувача та надання адекватного зворотного зв'язку. Зокрема надано можливість користувачам скасовувати або коригувати попередні дії, такі як видалення та повторне створення сповіщень, а також модифікація параметрів сутностей (наприклад,

організацій, приміщень) у випадку початкового некоректного введення даних.

17. Система зворотного зв'язку. Інтегровано централізований компонент (панель сповіщень) на кожній сторінці для уніфікованого відображення системних повідомлень. Ці повідомлення включають індикацію успішного виконання операцій та деталізацію причин (наприклад, відсутність необхідних даних) у випадку виникнення помилок при спробі виконання дії, що дозволяє користувачеві ідентифікувати та усунути проблему перед повторною спробою.

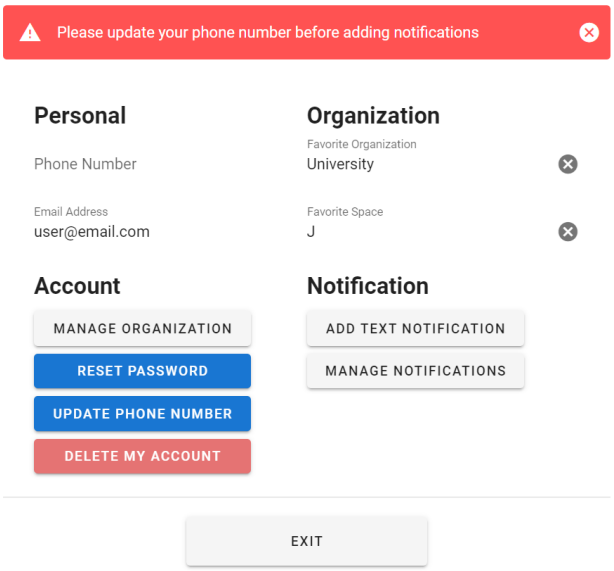

Рисунок 3.12 – Приклад повідомлення помилки

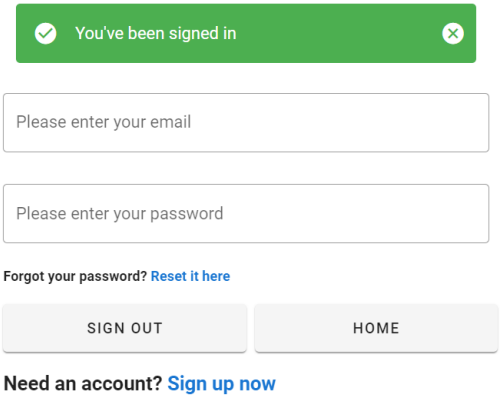

Рисунок 3.12 – Приклад повідомлення успішності дії

3.2.2. Оцінка атрибутів якості: корисність, бажаність, достовірність та придатність до використання

Оцінка цінності даного програмного продукту безпосередньо залежить від його ефективності у вирішенні або зменшенні проблем, ідентифікованих у першому розділі (мотивація). Пост-імплементаційний аналіз впровадження додатку продемонструє покращення або усунення зазначених проблем, тим самим підтверджуючи корисність додатку для цільової організації та кінцевих користувачів. На момент підготовки даного документа, розгортання додатку ще не завершено, що унеможливлює емпіричну оцінку його кінцевої корисності на даному етапі.

Дизайн інтерфейсу користувача був реалізований з акцентом на мінімалізм та простоту. Візуально привабливий дизайн веб-інтерфейсу є значущим фактором, що впливає на користувацьку бажаність. Враховуючи обмеженість часових ресурсів, пріоритет при розробці функціоналу надавався тим можливостям, що були визначені як найбільш бажані для кінцевого користувача. Всі інтегровані функції (наприклад, система сповіщень, збереження параметрів пошуку) спрямовані на забезпечення їх активного використання та відповідності потребам користувачів.

Критичним аспектом забезпечення достовірності додатку є точність даних, які він надає. Наприклад, якщо інформація про температуру, що відображається для конкретного віконного отвору, не відповідає фактичним показникам у відповідній зоні, це миттєво підриває довіру користувача до достовірності системи, особливо щодо такої базової функції, як агрегація точних метеорологічних даних. Некоректна робота будь-якого іншого функціоналу додатку також призводить до зниження його достовірності в сприйнятті користувачів. Для забезпечення високого рівня достовірності було використано API з верифікованого та надійного джерела метеорологічних даних. Додатково проведено всебічне тестування функціоналу для мінімізації збоїв. У випадках, коли збій все ж відбувається,

система передбачає інформування користувача про причину виникнення помилки.

Оцінка загальної придатності додатку до використання (корисності) охоплює перспективи ключових груп зацікавлених сторін: мешканців будівель, що використовують додаток для прийняття рішень щодо стану віконних прорізів. Оскільки поточний функціонал додатку повністю відповідає початковому технічному завданню, подальше визначення ступеня його корисності є прерогативою замовника та базуватиметься на досвіді та оцінках кінцевих користувачів після етапу експлуатації.

3.3. Реалізація модуля керування обліковими записами

3.3.1. Огляд

Користувач може створити обліковий запис, заповнивши форму реєстрації з своєю електронною адресою, паролем, підтвердженням пароля та потім надіславши пароль. Якщо користувач успішно надішле форму, додаток створить обліковий запис для користувача та увійде в систему.

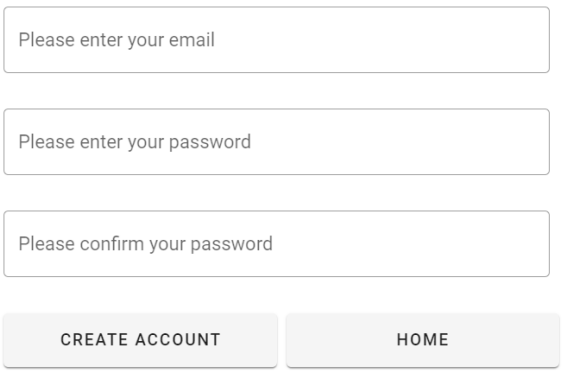

Рисунок 3.14 - Форма створення облікового запису

Після того, як користувач створить обліковий запис і додаток увійде в систему, або вони увійдуть вручну, вони зможуть отримати доступ до своїх

налаштувань і отримати доступ до всіх функцій для аутентифікованого користувача. Ці функції включають:

- Збереження вибору пошуку
- Автозавантаження вибору пошуку
- Керування сповіщеннями
- Керування організацією

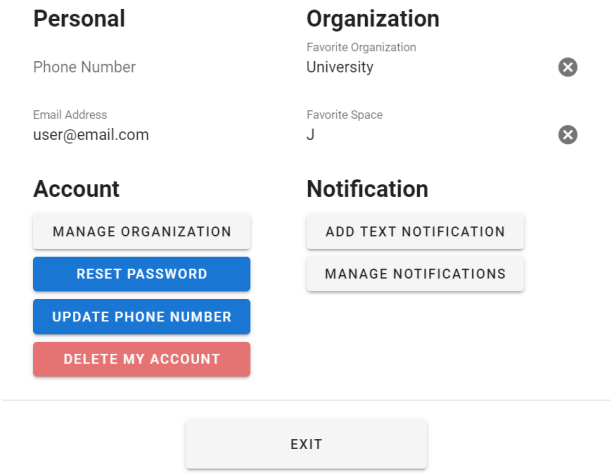

Рисунок 3.15 - Налаштування користувача

Аутентифікований користувач також може оновити свою особисту інформацію та налаштування облікового запису. Це включає:

- Скидання та повторне введення свого пароля.
- Оновлення свого номера телефону.
- Очищення своїх налаштувань пошуку.
- Видалення свого облікового запису та всієї пов'язаної з ним інформації.

Ці функції дозволяють користувачеві видалити свій цифровий слід, якщо він цього бажає. Нарешті, якщо користувач втрачає свою аутентифікацію, він може повторно увійти, ввівши свою електронну адресу та пароль.

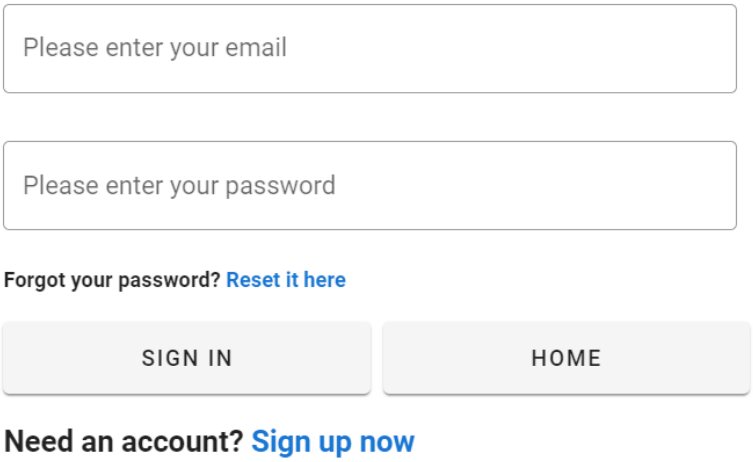

Рисунок 3.16 - Екран входу в систему

3.3.2. Реалізація аутентифікації і налаштувань облікового запису

Керування обліковими записами вимагає аутентифікації облікового запису. Аутентифікація облікового запису не може керуватися на клієнтській стороні та вимагає використання backend-сервісів. Backend-сервіси, що були використані, походять від Google Firebase Authentication [11]. Google Firebase Authentication має детальну документацію щодо налаштування аутентифікації користувача. Аутентифікація, яку було обрано — це стандартна електронна адреса та парольна аутентифікація. Стандартна електронна адреса та парольна аутентифікація узгоджує баланс між безпекою, ефективністю та зручністю використання. Користувач міг би керувати своїм обліковим записом лише з електронною адресою та паролем після того, як я реалізував наступні п'ять функцій, які надали б користувачеві можливість керувати своїм обліковим записом без втручання адміністратора. Користувач принаймні повинен мати можливість:

- Зареєструватися для облікового запису з електронною адресою та паролем.
- Увійти в свій обліковий запис з електронною адресою та паролем.
- Скинути свій пароль, якщо він вийшов з системи.
- Скинути свій пароль, якщо він увійшов в систему.
- Видалити свій обліковий запис.

Імплементація представленого функціоналу виконана згідно з офіційною документацією для веб-версії 9 бібліотеки Google Firebase [12]. Для управління асинхронними операціями замість традиційних ланцюжків обіцянок (promise chaining) застосовано синтаксис async/await, що підвищує читабельність та спрощує логіку асинхронного коду. Обробка винятків на рівні додатку реалізована з використанням конструкцій try...catch. Загальна стратегія обробки помилок передбачає перехоплення та логування системних повідомлень про помилки та їх кодів, при цьому деталі цих помилок не відображаються кінцевому користувачеві з міркувань безпеки та запобігання потенційному зловживанню інформацією про внутрішні помилки системи. Виняток становить процес реєстрації облікового запису, де користувач отримує специфічне повідомлення про існування облікового запису з вказаною електронною адресою, що є необхідним для коректної взаємодії користувача із системою реєстрації.

При успішному створенні облікового запису та ініціації процесу реєстрації організації, користувачеві пропонується заповнити структуровану форму. Ця форма вимагає введення назви організації, міста та штату. Після надсилання даних форми системою виконується валідація на предмет дублювання – перевіряється наявність організації з ідентичними параметрами. У разі унікальності введених даних, додатком автоматично виконується запит до зовнішнього API метеорологічних сервісів (OpenWeatherMap) з метою отримання відповідних погодних даних. При успішному отриманні даних від API, система створює новий документ у базі даних Cloud Firestore, що представляє організацію. Цей документ містить назву організації, місто та штат як пари ключ-значення. Власницькі права користувача на створену організацію фіксуються шляхом оновлення документа, що відповідає профілю користувача, у Cloud Firestore.

Після успішного завершення реєстрації організації користувач отримує доступ до спеціалізованого інтерфейсу для управління нею. Навігація до

цього розділу здійснюється через меню налаштувань користувача, де елемент інтерфейсу, призначений для реєстрації організації, динамічно змінюється на посилання для управління організацією за умови, що користувач є її власником. Система верифікує статус власності при першому завантаженні сторінки налаштувань, використовуючи функціонал аутентифікації Google Firebase для отримання даних про поточного користувача та моніторингу змін його стану аутентифікації. Інтерфейс управління організацією додатково перевіряє автентичність користувача при доступі до нього, підтверджуючи, що користувач є законним власником організації, яка була асоційована з його обліковим записом під час створення. У випадку зміни статусу аутентифікації користувача або втрати прав власності на організацію, система автоматично ініціює перенаправлення користувача з інтерфейсу управління організацією. Механізм перевірки та перенаправлення ідентичний тому, що використовується на сторінці налаштувань. Всередині інтерфейсу управління власник організації має можливість здійснювати операції оновлення, додавання та видалення об'єктів "приміщення", дані про які персистентно зберігаються у Cloud Firestore як елементи підколекції, вкладеної у документ відповідної організації.

На поточному етапі розробки реалізовано базовий набір інструментів та функціональності, необхідних для забезпечення аутентифікації користувачів у системі. Подальший розвиток додатку передбачає можливість розширення цієї підсистеми з метою покращення користувацького досвіду та збільшення гнучкості методів входу.

Потенційні напрямки модернізації включають інтеграцію альтернативних механізмів аутентифікації, таких як аутентифікація за номером телефону. Використання номера телефону може слугувати не тільки для верифікації особи користувача, але й як канал для доставки сповіщень у форматі текстових повідомлень.

З метою усунення потреби для користувачів створювати специфічний обліковий запис у даній системі, може бути імплементована підтримка федеративних постачальників ідентичності через протокол OAuth. Інтеграція з популярними постачальниками, такими як Google Sign-in або Microsoft Login, передає відповідальність за управління обліковими даними зовнішнім суб'єктам, що знижує адміністративне навантаження на власника додатку.

3.4. Реалізація функціональності сповіщення

Відповідно до технічного завдання проєкту, передбачалася реалізація функціоналу сповіщень. Система надає користувачам можливість конфігурувати параметри сповіщень та отримувати релевантну інформацію щодо умов експлуатації віконних прорізів у визначених просторових одиницях, на які оформлено підписку. На етапі проєктування розглядався механізм доставки сповіщень засобами електронної пошти та SMS. Відмова від використання електронної пошти як основного каналу доставки була аргументована аналізом користувацьких патернів (перевантаженість поштових скриньок) та оцінкою ресурсних витрат на імплементацію відносно очікуваної користі.

Зрештою, фокус реалізації було зміщено на функціонал текстових сповіщень. Система дозволяє користувачам створювати множинні екземпляри текстових сповіщень. Конфігурація кожного сповіщення вимагає визначення наступних шести обов'язкових атрибутів:

- Ідентифікатор організації: Посилання на структурну одиницю (організацію), якій належить відповідний простір (наприклад, University).

- Ідентифікатор простору: Посилання на конкретний простір (приміщення), що містить цільовий віконний проріз (наприклад, Research Lab).

- Час ініціації сповіщення. Вказаний час доби для обчислення та доставки користувачеві інформації про статус можливості відкриття вікна (наприклад, 13:00).

- Розклад повторень. Набір днів тижня, протягом яких сповіщення має надсилатися на регулярній основі (наприклад, понеділок, середа, п'ятниця).

- Дата активації. Календарна дата, починаючи з якої система розпочинає надсилання запланованих сповіщень (наприклад, 25 травня 2025 року).

- Дата деактивації. Календарна дата, після якої надсилання сповіщень припиняється (наприклад, 30 червня 2025 року). Система передбачає автоматичне архівування або видалення записів про сповіщення з терміном дії, що минув, з персистентного сховища даних.

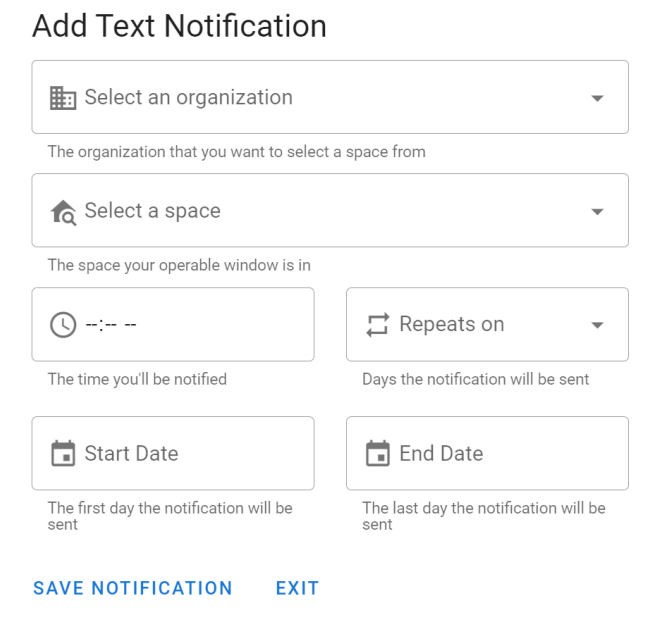

Рисунок 3.17 – Налаштування сповіщувача

Нарешті, користувачі можуть керувати своїми сповіщеннями, що дозволяє їм переглядати список усіх своїх сповіщень, а також увімкнути, вимкнути або видалити будь-яке з них.

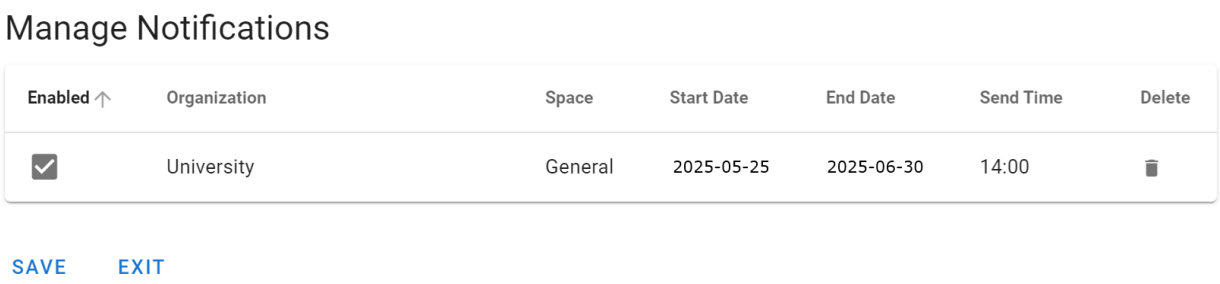

Рисунок 3.18 – Керування сповіщеннями

Підсистема сповіщень є найбільш складною підсистемою в межах даного проєкту. Процес доставки сповіщення від моменту його ініціації до отримання кінцевим користувачем реалізовано у вигляді багатоетапного конвеєра. Користувачі мають повний контроль над конфігурацією та життєвим циклом власних сповіщень. Структура зберігання даних передбачає асоціацію сповіщень з індивідуальним документом кожного користувача у персистентному сховищі.

Серверний планувальник завдань (cron job), що виконується на базі функцій безсерверної інфраструктури (Cloud Functions), здійснює періодичну перевірку (з інтервалом в одну хвилину) статусу всіх запланованих сповіщень та ініціює доставку тих, що відповідають встановленим критеріям [13]. Алгоритм обробки сповіщень включає серію перевірок, зокрема валідацію терміну дії сповіщення шляхом порівняння поточної дати з датою завершення. Сповіщення, термін дії яких вичерпано, автоматично видаляються із відповідного документа користувача в базі даних. Сповіщення, які пройшли всі необхідні валідаційні перевірки та відповідають критеріям для надсилання, ставляться у чергу на доставку. Для здійснення безпосередньої доставки текстових повідомлень інтегровано зовнішній програмний інтерфейс (API) MessageBird. Інтеграція даного API в структуру проєкту спрощена за рахунок використання відповідних розширень Firebase [14]. Це розширення відповідає за створення запису про факт надсилання

сповіщення у спеціалізованій колекції документів, що містить метадані щодо відправленого повідомлення.

При розробці архітектури системи сповіщень було враховано ці регуляторні аспекти, що забезпечило її високу розширюваність та дозволяє легко інтегрувати інші API для доставки повідомлень. Ця архітектурна гнучкість стосується не тільки інтеграції SMS API, але й підтримки інших каналів комунікації, таких як електронна пошта та push-сповіщення, якщо таке рішення буде прийнято замовником.

На ранніх етапах проєктування обговорювалася можливість імплементації функціоналу, який дозволив би адміністраторам системи ініціювати масове сповіщення всіх користувачів з метою надання рекомендацій щодо закриття віконних прорізів у конкретній організації та просторі. Необхідність такої функції аргументувалася потенційною потребою оперативного інформування користувачів про непередбачені події (як природного, так і антропогенного походження), що вимагають уникнення зовнішнього впливу (наприклад, витоки газу, неналежне поводження з відходами, поява диких тварин, проникнення сторонніх осіб). Реалізація зазначеного функціоналу була відкладена через часові обмеження проєкту, а також внаслідок того, що вибір між внутрішнім або зовнішнім розгортанням системи суттєво впливає на архітектурний підхід до імплементації цієї можливості.

3.5. Модуль управління структурними одиницями

Центральною функціональною можливістю даної програмної системи є надання користувачеві інструментарію для пошуку та ідентифікації організацій і пов'язаних з ними просторових одиниць (приміщень). Функції створення та управління цими сутностями доступні користувачам, проте їх активація залежить від наявності асоційованої організації. Веб-додаток надає

користувачеві механізм первинної реєстрації організації за її відсутності. Після успішного завершення процесу реєстрації користувач отримує доступ до спеціалізованого інтерфейсу управління організацією, що дозволяє виконувати операції створення, модифікації та видалення підпорядкованих приміщень, а також деактивації (видалення) самої організації. При створенні нового запису про приміщення застосовуються параметри, визначені у першому розділі, які є визначальними для обчислення можливості відкриття відповідних віконних прорізів. Власник організації має можливість встановлювати специфічні порогові значення для кожного приміщення, включаючи максимальний рівень вологості, діапазон мінімальної та максимальної температури, а також максимальне значення індексу якості повітря [17].

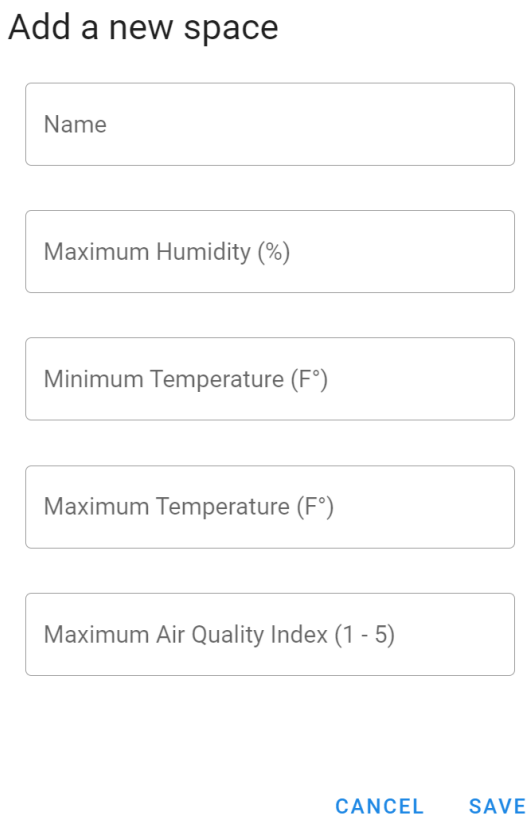

Рисунок 3.19 - Форма додавання нового приміщення

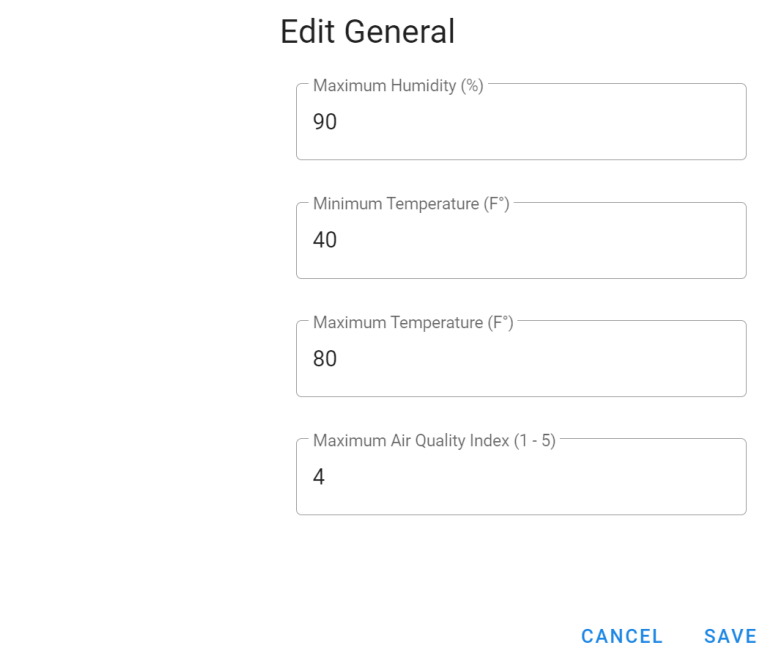

Рисунок 3.19 - Форма редагування показників в приміщенні

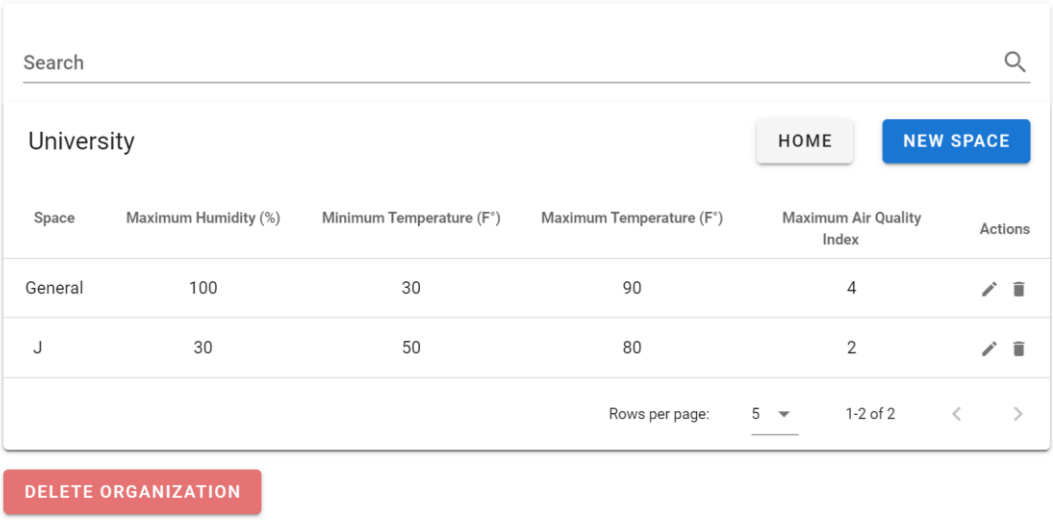

Рисунок 3.20 – Керування приміщеннями

Ці параметри в подальшому використовуються системою для виконання розрахунків стану вікон у вказаному приміщенні. Після реєстрації організації її стає доступною для пошуку та вибору будь-яким користувачем системи. Користувач може обрати будь-яке приміщення, асоційоване з вибраною організацією, для перегляду поточного статусу можливості відкриття вікон у цьому приміщенні.

Процес реєстрації організації ініціюється після створення облікового запису користувача та його переходу до відповідного розділу додатку. Користувачеві пропонується заповнити форму, що вимагає введення назви організації, міста та штату. Після надсилання форми система виконує перевірку на наявність дублюючих записів організації в базі даних. У разі відсутності існуючої організації з ідентичними параметрами, система ініціює запит до зовнішнього програмного інтерфейсу метеорологічних сервісів (OpenWeatherMap). Успішне отримання даних від OpenWeatherMap дозволяє створити новий документ організації у сховищі Cloud Firestore, де назва, місто та штат зберігаються як атрибути. Право власності користувача на створену організацію фіксується шляхом оновлення відповідного документа користувача.

Після успішного завершення процедури створення організації, користувач отримує можливість перейти до окремого інтерфейсу управління. Навігація до цього інтерфейсу здійснюється через меню налаштувань користувача, де елемент інтерфейсу, призначений для реєстрації, динамічно змінюється на посилання управління за умови підтвердження власності. Верифікація автентичності користувача та його прав власності виконується при початковому завантаженні сторінки налаштувань з використанням функціоналу Google Firebase Authentication та моніторингу зміни стану автентифікації [18]. Інтерфейс управління організацією додатково перевіряє аутентифікацію користувача при доступі, підтверджуючи його статус як власника організації, асоційованої з обліковим записом під час створення. У випадку зміни стану аутентифікації або втрати прав власності, система автоматично здійснює перенаправлення користувача з інтерфейсу управління організацією. Механізм цієї обробки ідентичний тому, що застосовується у розділі налаштувань. В межах інтерфейсу управління власник організації має можливість виконувати операції модифікації, додавання або видалення

підпорядкованих приміщень, дані про які персистентно зберігаються у Cloud Firestore в межах підколекції, інкапсульованої у документ організації.

Функціонал, пов'язаний з реєстрацією та управлінням організаціями, а також операціями з приміщеннями (створення, оновлення, видалення), є повністю реалізованим. Однак існують можливості для подальших вдосконалень, спрямованих на підвищення зручності використання як для кінцевих користувачів, так і для адміністраторів. Початкові вимоги до додатку включали створення користувацького інтерфейсу, оптимізованого для мобільних пристроїв. Мобільна версія інтерфейсу пошуку та перегляду статусу вікон реалізована, проте інтерфейс для управління організаціями, хоча і є функціональним, може бути оптимізований для покращення користувацького досвіду, зокрема на мобільних платформах.

3.6. Аспекти безпеки

Згідно з оцінками фахівців у галузі безпеки веб-додатків, "для компрометації системи достатньо виявити лише одну вразливість, тоді як для забезпечення її захисту необхідно ідентифікувати та усунути всі потенційні слабкі місця [19]". З огляду на відкриту природу мережі Інтернет, веб-додатки апріорі є об'єктами підвищеного ризику. Аудит безпеки, проведений Positive Technologies, засвідчив виявлення високоризикових вразливостей у 100% проаналізованих банківських та фінансових веб-додатків [20]. Незважаючи на високі вимоги до захисту критичних систем, таких як банківські, наявність єдиної експлуатованої вразливості може призвести до повної компрометації.

3.6.1. Огляд політики безпеки

Забезпечення комплексного захисту веб-додатків є надзвичайно складним завданням, і досягнення абсолютної невразливості може вважатися

теоретично неможливим [21]. Створений веб-додаток не обробляє дані, що класифікуються як високочутливі (на відміну від фінансових або державних систем), тому потенційний витік даних у найгіршому випадку становитиме лише незначну незручність для користувачів. Незважаючи на це, було вжито максимально можливих заходів для підвищення рівня захищеності системи.

Архітектура веб-додатку базується на платформі Google Firebase. При розробці було дотримано рекомендацій зі списку перевірки безпеки, опублікованого Google для забезпечення веб-додатків. Оскільки розгортання проєкту замовником ще не відбулося, було імплементовано всі доступні на даному етапі заходи захисту. Перелік верифікованих та впроваджених кроків включає:

- Локальне тестування функціоналу Cloud Functions за допомогою емуляторів.

- Захист серверної логіки, реалізованої у вигляді єдиної функції Cloud Functions, від несанкціонованого доступу та зловживання за рахунок її активації виключно за розкладом (cron job).

- Конфігурація правил безпеки (Security Rules) для кожного ресурсу в базі даних Cloud Firestore, що визначають права доступу до даних.

- Встановлення лімітів та квот на операції аутентифікації за допомогою електронної пошти та пароля для запобігання перебору.

- Заборона передачі токенів аутентифікації в відкритому вигляді.

- Відмова від використання анонімної аутентифікації.

Вищезазначені заходи забезпечують захист серверної частини (backend). Для гарантування того, що користувачі мають дозволи лише на читання, запис, оновлення та видалення даних відповідно до їх ролі, регулярно виконується верифікація стану аутентифікації користувача – як при його зміні, так і при переході на нову сторінку. Це дозволяє переконатися, що дії користувача відповідають його поточним правам доступу. Крім того, з метою надання користувачам адекватного зворотного

зв'язку щодо успішності чи невдачі їхніх дій, було прийнято рішення відображати загальні повідомлення про помилки, уникаючи специфічних кодів помилок, які могли б бути потенційно використані зловмисниками для експлуатації системи.

3.6.2. Імплементація заходів безпеки

Побудова захищеного додатку вимагає як проактивних дій, спрямованих на підвищення стійкості системи до атак, так і уникнення типових помилок розробки, що можуть створити вразливості (наприклад, розміщення секретних ключів доступу до backend-сервісів на клієнтській стороні).

Для забезпечення безпеки серверної частини системи було реалізовано кілька ключових заходів. Першим з них стало розміщення API-ключа, необхідного для взаємодії із сервісом OpenWeatherMap, виключно на стороні сервера. Розміщення цього ключа на клієнтській стороні могло б дозволити зловмиснику перехопити його та автоматизувати надмірну кількість запитів до сервісу, що призвело б до блокування ключа та, як наслідок, до повної нефункціональності основного призначення додатку, оскільки запити до постачальника послуг стали б неможливими.

Другим важливим заходом є використання інтегрованих сервісів Firebase Authentication та правил безпеки Cloud Firestore для реалізації механізмів безсерверної аутентифікації, авторизації та валідації даних. Завдання полягає у захисті даних, що зберігаються на backend, як від цілеспрямованих зловмисних атак, так і від ненавмисного доступу чи модифікації даних користувачами, які не мають відповідних дозволів. Первинним кроком у цьому процесі є надання користувачам можливості створення облікових записів, що слугують основою для їх аутентифікації. Наявність аутентифікації користувача дозволяє системі виконувати авторизацію перед ініціацією дій, які вимагають певних прав доступу. Це

досягається шляхом використання обчислюваного властивості в інтерфейсній частині додатку (написаній на Vue), яке динамічно отримує поточний стан аутентифікації користувача за допомогою Firebase Authentication [23]. Замість передачі стану аутентифікації, додаток отримує його актуальну версію безпосередньо перед виконанням критичних дій, що унеможливлює маніпуляції зі статусом авторизації користувача та гарантує його відповідність поточним правам.

На основі отриманих даних аутентифікації та авторизації, були розроблені та впроваджені правила безпеки у Cloud Firestore для перевірки наявності у користувача необхідних дозволів для виконання запиту. При спробі користувача отримати доступ до документа або колекції в Cloud Firestore, ці правила аналізують низку параметрів, включаючи статус аутентифікації користувача та його унікальний ідентифікатор, для визначення дозволу на виконання запиту. Наприклад, всім користувачам (як аутентифікованим, так і неаутентифікованим) надано право на читання даних про всі організації та приміщення, проте право на запис (створення, оновлення, видалення) даних організації надано виключно її власнику. Доступ до читання та запису даних власних сповіщень обмежено лише користувачем, який їх створив. Важливою особливістю є взаємодія із серверною логікою (Cloud Functions). Оскільки Cloud Functions потребують доступу для читання та запису даних сповіщень користувачів, але не є "користувачами" в стандартному розумінні, їх функціонування забезпечується за рахунок використання серверних клієнтських бібліотек Firebase, які обходять правила безпеки Cloud Firestore [25], надаючи привілейований доступ.

Однак багато аспектів подальшого посилення безпеки залежать від рішень замовника щодо конфігурації розгорнутої системи. Існує кілька ключових заходів безпеки, які рекомендовано імплементувати майбутнім

розробникам або виконати перед етапом промислового розгортання. До них належать:

- активація функціоналу App Check для забезпечення захисту backend-ресурсів від несанкціонованого використання та зловживань.

- налаштування системи моніторингу та сповіщень про використання ресурсів, що наближається до встановлених порогових значень та квот.

- встановлення лімітів використання для всіх платних ресурсів Firebase для контролю витрат та запобігання атакам типу "відмова в обслуговуванні".

- обмеження використання API-ключа додатку Firebase виключно для доступу з домену розгорнутого веб-сайту.

- моніторинг квот та аналіз спроб входу до системи для виявлення та запобігання атакам перебору паролів.

- імплементація підтримки постачальників ідентичності за протоколом OAuth 2.0.

З імплементацією всіх вищезазначених заходів, система досягне значно вищого рівня захищеності, що є критично важливим, особливо враховуючи потенційні загрози з боку зловмисних суб'єктів.

3.7. Оцінка стратегій розгортання додатку

Прийняття рішення щодо стратегії розгортання потребує визначення замовником ряду ключових аспектів та надання необхідних ресурсів, як зазначено в розділах, що стосуються розробки додатку. Ці аспекти включають питання конфігурації безпеки, зокрема встановлення лімітів витрат та порогів використання ресурсів, а також надання ресурсів, таких як дійсний API-ключ для сервісів комунікації з користувачами та забезпечення фінансування ресурсів Google Firebase. Після завершення розробником попередніх етапів підготовки, замовник може обрати оптимальну стратегію розгортання додатку.

3.7.1. Стратегія зовнішнього розгортання

Початкова архітектура додатку розроблялася з орієнтацією на публічну доступність. Однак, наявність певних невирішених питань становить підставу для занепокоєння стосовно розгортання у відкритому доступі. Основною проблемою є відсутність формалізованих механізмів модерації організацій. На поточному етапі будь-який користувач може зареєструвати організацію, використовуючи дійсні географічні дані (місто) та унікальну назву. Це створює ризик можливості імітації легітимних організацій зловмисниками з метою введення користувачів в оману та перенаправлення їх на пошук приміщень у фейкових структурах. Потенційним рішенням цієї проблеми може бути впровадження ролі суперадміністратора для здійснення модерації та затвердження реєстрації організацій. Однак, цей підхід пов'язаний зі значними операційними витратами, зумовленими необхідністю залучення людських ресурсів для ручної модерації запитів на створення організацій та управління адміністративними обліковими записами. Додатковою проблемою є необхідність покриття замовником усіх експлуатаційних витрат для всіх зареєстрованих у системі організацій. Незважаючи на зазначені виклики, публічне розгортання має вагомі переваги. Зокрема, значний обсяг користувацького зворотного зв'язку сприятиме швидкому ітераційному покращенню функціоналу та підвищенню загальної корисності додатку. Крім того, початкова орієнтація розробки на публічне розгортання означає, що значні інвестиції часу в архітектуру та функціонал будуть виправдані.

3.7.2. Стратегія внутрішнього розгортання

Альтернативною стратегією є внутрішнє розгортання (on-premise або приватне хмарне розгортання). Цей підхід дозволяє усунути більшість проблем, асоційованих з публічним розгортанням. Однак, поточна архітектура додатку не оптимізована для такого сценарію. Імплементація

внутрішнього розгортання вимагатиме значних часових витрат на рефакторинг кодової бази (оціночно десятки годин). Зокрема, необхідно буде модифікувати логіку, що стосується пошуку, створення та управління організаціями, оскільки користувачі всередині однієї організації не потребуватимуть пошуку своєї організації або вибору її для отримання сповіщень. Для забезпечення коректної роботи в умовах внутрішнього розгортання, необхідно буде розробити та інтегрувати нові модулі для обробки аутентифікації та безпеки, адаптовані до специфіки внутрішнього середовища. Незважаючи на значні технічні виклики, внутрішнє розгортання має певні переваги. До них належать значне підвищення рівня безпеки за умови належної реалізації підсистем аутентифікації та контролю доступу, а також можливість створення більш цілеспрямованого та оптимізованого користувацького досвіду, оскільки функціонал додатку буде обмежений контекстом конкретної організації.

3.7.3. Гібридна стратегія: внутрішнє розгортання з відкритим вихідним кодом

Третій варіант, який поєднує переваги двох попередніх підходів, передбачає внутрішнє розгортання системи при одночасній публікації вихідного коду у відкритому доступі. Ця стратегія зберігає переваги внутрішнього розгортання, такі як підвищена безпека та оптимізований користувацький досвід, і водночас надає можливість іншим організаціям адаптувати та розгорнути додаток на власних ресурсах, використовуючи наданий вихідний код. Це потенційно може призвести до ширшого впровадження додатку різними організаціями, що, у свою чергу, збільшить обсяг користувацького зворотного зв'язку.

Основним недоліком цієї гібридної стратегії є найдовший термін реалізації. Вона вимагає не тільки значного обсягу рефакторингу коду, аналогічного тому, що потрібен для чисто внутрішнього розгортання, але й

додаткових зусиль на підготовку вичерпної технічної документації, необхідної для організацій, які зацікавлені в самостійному розгортанні додатку.

Висновки до розділу

В даному розділі було здійснено повноцінну програмну реалізацію функціонального сповіщувача енергоефективності з урахуванням принципів адаптивності, ергономіки, безпеки та масштабованості. Реалізація інтерфейсу користувача (підрозділ 3.1) передбачала створення адаптивного дизайну, що забезпечує комфортне використання додатку на різних типах пристроїв, враховуючи розмір екрана, особливості сенсорного керування та візуальної навігації.

Особливу увагу приділено користувацькому досвіду (3.2), де ергономічні аспекти були реалізовані через зручну навігацію, швидкий доступ до ключових функцій та інтуїтивно зрозумілий інтерфейс. Оцінка атрибутів якості (3.2.2) підтвердила відповідність системи очікуванням кінцевого користувача за такими критеріями, як корисність, достовірність і придатність до щоденного використання.

У підрозділі 3.3 описано модуль керування обліковими записами, зокрема, реалізовано систему аутентифікації, а також інтерфейси для налаштування персональних даних користувача. Модуль управління структурними одиницями (3.5) забезпечує логічне групування об’єктів енергоспоживання, що сприяє масштабованості системи в межах багатокористувацького середовища.

У підрозділі 3.6 проаналізовано ключові аспекти інформаційної безпеки, включаючи політику безпеки та технічну реалізацію захисту даних — від механізмів шифрування до контролю доступу.

Нарешті, у підрозділі 3.7 були оцінені стратегії розгортання додатку. Переваги та недоліки зовнішнього, внутрішнього та гібридного розгортання дозволили обґрунтувати вибір моделі, яка найкраще відповідає вимогам надійності, доступності та підтримки відкритості коду.

Загалом, результати цього розділу демонструють цілісність реалізованого рішення, його готовність до практичного застосування, а також потенціал до подальшого вдосконалення та масштабування.

ВИСНОВКИ

В дипломній роботі було проведено всебічне дослідження теоретичних і практичних аспектів проєктування та розробки програмного засобу — функціонального сповіщувача енергоефективності. Робота охоплює всі етапи створення інноваційного інструменту для моніторингу енергоспоживання з орієнтацією на сучасні виклики у сфері сталого розвитку та цифрової трансформації.

У першому розділі здійснено ґрунтовний аналіз предметної області застосування новітніх технологій у програмах забезпечення енергоефективності. Розглянуто мотиваційні фактори для створення подібного програмного забезпечення, зокрема у контексті HVAC-систем. Обґрунтовано використання генетичних алгоритмів для оптимального керування кліматичними системами. Визначено функціональні цілі розробки та ключові параметри, які підлягають моніторингу, що забезпечило формування чітких технічних вимог до майбутнього застосунку.

Другий розділ був присвячений вибору інструментів розробки. Проаналізовано переваги різних фреймворків, бібліотек і платформ, що дозволило сформувати оптимальний стек технологій. Особливу увагу приділено принципам візуального дизайну відповідно до концепції Material Design, що забезпечує доступний та привабливий інтерфейс користувача.

У третьому розділі було реалізовано програмний прототип системи. Здійснено розробку адаптивного інтерфейсу, модулів аутентифікації, управління обліковими записами та структурними одиницями. Проведено аналіз користувацького досвіду, що підтвердив відповідність розробки сучасним вимогам до ергономіки, зручності та якості інтерфейсу. Реалізовано механізми захисту даних та розглянуто можливі стратегії розгортання додатку в реальному середовищі.

Загалом, результати бакалаврської роботи підтверджують доцільність створення програмного сповіщувача енергоефективності як ефективного інструменту для вчасного реагування на відхилення в енергоспоживанні та забезпечення більш раціонального використання ресурсів. Отримані результати можуть бути використані як основа для впровадження інтелектуальних систем моніторингу у розумних будівлях, комерційних установах і побутовому секторі.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. “Heating, Ventilation and Air-Conditioning Systems, Part of Indoor Air Quality Design Tools for Schools,” EPA: https://www.epa.gov/iaq-schools/heating-ventilation-and-air-conditioningsystems-part-indoor-air-quality-design-tools.
2. Pérez-Lombard, L.; Ortiz, J.; Pout, C. A review on buildings energy consumption information. Energy Build. 2008, 40, 394–398.
3. Allouhi, A.; El Fouih, Y.; Kousksou, T.; Jamil, A.; Zeraouli, Y.; Mourad, Y. Energy consumption and efficiency in buildings: Current status and future trends. J. Clean. Prod. 2015, 109, 118–130.
4. Cunningham, A. Designing Energy Efficient Hospitals? First Let’s Give You the Facts on Existing Performance; Western Cape Government Health Department: Sandton, South Africa, 2015.
5. “Introduction to client-side frameworks,” Learn web development: https://developer.mozilla.org/en-US/docs/Learn/Tools and testing/Clientside JavaScript frameworks/Introduction.
6. “About JavaScript,” JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About JavaScript.
7. “Understanding client-side JavaScript frameworks,” Learn web development, https://developer.mozilla.org/en-US/docs/Learn/Tools and testing/Client-side JavaScript frameworks.
8. Amowine, N.; Ma, Z.; Li, M.; Zhou, Z.; Azembila Asunka, B.; Amowine, J. Energy Efficiency Improvement Assessment in Africa: An Integrated Dynamic DEA Approach. Energies 2019, 12, 3915.
9. Aydin, Y.C.; Mirzaei, P.A.; Akhavannasab, S. On the relationship between building energy efficiency, aesthetic features and marketability: Toward a novel policy for energy demand reduction. Energy Policy 2019, 128, 593–606.

10. Javied, T.; Rackow, T.; Franke, J. Implementing energy management system to increase energy efficiency in manufacturing companies. Procedia CIRP 2015, 26, 156–161.
11. Understanding Vue.js Architecture and How It Works | by Kiran Bhagannavar | Medium - https://hashtagkiran.medium.com/understanding-vue-js-architecture-and-how-it-works-553f93e2d1d6
12. Node.js Architecture: Understanding Node.js Architecture | by Ibrahim Lanre Adedimeji | Medium - https://medium.com/@ibrahimlanre1890/node-js-architecture-understanding-node-js-architecture-5fb32879b994
13. "A material design framework for vue.js," Vuetify: https://vuetifyjs.com/en/.
14. "Firestore," Google Cloud: https://cloud.google.com/firestore.
15. P. Morville, "User experience design," Semantic Studios: http://semanticstudios.com/user experience design/.
16. "Usability principles," Stanford Improvement, Analytics, and Innovation Services: https://improvement.stanford.edu/resources/usability-principles.
17. "Firebase Authentication," Firebase Documentation: https://firebase.google.com/docs/auth.
18. "Get Started with Firebase Authentication on Websites ," Firebase Documentation: https://firebase.google.com/docs/auth/web/start.
19. "Manage Users in Firebase," Firebase Documentation: https://firebase.google.com/docs/auth/web/manage-users.
20. R. Abela, "The dangerous complexity of web application security," Web Application Security can Become Dangerously Complex: https://www.invicti.com/blog/web-security/dangerouscomplexity-of-web-application-security/.
21. Aghenta, L. O., & Iqbal, M. T. (2019). Design and implementation of a smart energy monitoring system. Energy Reports, 5, 1410–1422. https://doi.org/10.1016/j.egyr.2019.09.025

22. Ali, A., Rehman, A. U., & Khan, M. A. (2020). IoT-based energy management system for smart buildings. Sustainable Cities and Society, 61, 102231. https://doi.org/10.1016/j.scs.2020.102231
23. Aman, S., Simmhan, Y., & Prasanna, V. (2013). Energy management systems: State of the art and emerging trends. IEEE Communications Magazine, 51(1), 114–119. https://doi.org/10.1109/MCOM.2013.6400447
24. Arjunan, R., & Jawahar, P. (2022). Energy consumption monitoring and notification system using IoT. Journal of Ambient Intelligence and Humanized Computing, 13, 225–237. https://doi.org/10.1007/s12652-021-02940-7
25. Batra, N., Singh, A., & Srivastava, M. (2014). Data-driven energy efficiency in buildings. IEEE Data Eng. Bull., 37(4), 56–63.
26. Behl, M., Jain, A., & Mangharam, R. (2016). Data-driven modeling, control and tools for cyber-physical energy systems. Proceedings of the IEEE, 104(5), 998–1007. https://doi.org/10.1109/JPROC.2015.2510201
27. Chen, Y., Ahmad, I., & Shabbir, F. (2021). IoT-based smart energy management system for home applications. Energies, 14(7), 1932. https://doi.org/10.3390/en14071932
28. Darby, S. (2006). The effectiveness of feedback on energy consumption. Environmental Change Institute, University of Oxford.
29. Debnath, K. B., & Mourshed, M. (2018). Designing an intelligent energy monitoring system for residential buildings. Energy Procedia, 153, 229–234. https://doi.org/10.1016/j.egypro.2018.10.010
30. Erol-Kantarci, M., & Mouftah, H. T. (2015). Energy-efficient information and communication infrastructures in the smart grid: A survey on interactions and open issues. IEEE Communications Surveys & Tutorials, 17(1), 179–197. https://doi.org/10.1109/COMST.2014.2357071

ДОДАТКИ

Додаток А

Лістинг форми додавання повідомлення

```
interface NotificationSettings {
  organization: string;
  space: string;
  time: string;
  repeatsOn: string;
  startDate: string;
  endDate: string;
}

export default function AddTextNotification() {
  const [notificationSettings, setNotificationSettings] =
useState<NotificationSettings>({
    organization: '',
    space: '',
    time: '',
    repeatsOn: '',
    startDate: '',
    endDate: '',
  });

  const handleInputChange = (field: keyof NotificationSettings, value:
string) => {
    setNotificationSettings((prevSettings) => ({ ...prevSettings,
[field]: value }));
  };

  const handleSaveNotification = () => {
    console.log('Notification settings:', notificationSettings);
  };

  return (
    <div className="max-w-lg mx-auto p-6 bg-white rounded-lg shadow-
md">
      <h1 className="text-2xl font-bold mb-6">Add Text
Notification</h1>
      <div className="mb-4">
        <div className="flex items-center justify-between border
border-gray-300 rounded-lg p-3 cursor-pointer">
          <div className="flex items-center">
            <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
```

```
strokeLinejoin="round" strokeWidth="2" d="M3 7v10a2 2 0 002 2h14a2 2 0
002-2V9a2 2 0 00-2-2h-6l-2-2H5a2 2 0 00-2 2z"></path></svg>
            <span>Select an organization</span>
          </div>
          <svg className="w-5 h-5" fill="none" stroke="currentColor"
viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path
strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M19
9l-7 7-7-7"></path></svg>
        </div>
        <p className="text-sm text-gray-600 mt-1">The organization
that you want to select a space from</p>
      </div>
      <div className="mb-4">
        <div className="flex items-center justify-between border
border-gray-300 rounded-lg p-3 cursor-pointer">
          <div className="flex items-center">
            <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
strokeLinejoin="round" strokeWidth="2" d="M3 12l2-2m0 0l7-7 7 7M5
10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1 1h-3m-6 0a1 1 0 001-
1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6"></path></svg>
            <span>Select a space</span>
          </div>
          <svg className="w-5 h-5" fill="none" stroke="currentColor"
viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path
strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M19
9l-7 7-7-7"></path></svg>
        </div>
        <p className="text-sm text-gray-600 mt-1">The space your
operable window is in</p>
      </div>
      <div className="flex mb-4 space-x-4">
        <div className="flex-1">
          <div className="border border-gray-300 rounded-lg p-3
cursor-pointer">
            <div className="flex items-center">
              <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
strokeLinejoin="round" strokeWidth="2" d="M12 8v4l3 3m6-3a9 9 0 11-18
0 9 9 0 0118 0z"></path></svg>
              <span>--:-- --</span>
            </div>
          </div>
          <p className="text-sm text-gray-600 mt-1">The time you'll be
notified</p>
```

```
        </div>
        <div className="flex-1">
          <div className="flex items-center justify-between border
border-gray-300 rounded-lg p-3 cursor-pointer">
            <div className="flex items-center">
              <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
strokeLinejoin="round" strokeWidth="2" d="M4 4v5h.582m15.356 2A8.001
8.001 0 004.582 9m0 0H9m11 11v-5h-.581m0 0a8.003 8.003 0 01-15.357-
2m15.357 2H15M14.856 17.958l-3.536-3.536m0 0l3.536-3.536M14.856
14l3.536 3.536M14.856 14l-3.536 3.536"></path></svg>
              <span>Repeats on</span>
            </div>
            <svg className="w-5 h-5" fill="none" stroke="currentColor"
viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path
strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M19
9l-7 7-7-7"></path></svg>
          </div>
          <p className="text-sm text-gray-600 mt-1">Days the
notification will be sent</p>
        </div>
      </div>
      <div className="flex mb-6 space-x-4">
        <div className="flex-1">
          <div className="flex items-center border border-gray-300
rounded-lg p-3 cursor-pointer">
            <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
strokeLinejoin="round" strokeWidth="2" d="M8 7V3m8 4V3m-9 8h10M5
21h14a2 2 0 002-2V7a2 2 0 00-2-2H5a2 2 0 00-2 2v12a2 2 0 002
2z"></path></svg>
            <span>Start Date</span>
          </div>
          <p className="text-sm text-gray-600 mt-1">The first day the
notification will be sent</p>
        </div>
        <div className="flex-1">
          <div className="flex items-center border border-gray-300
rounded-lg p-3 cursor-pointer">
            <svg className="w-5 h-5 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round"
strokeLinejoin="round" strokeWidth="2" d="M8 7V3m8 4V3m-9 8h10M5
21h14a2 2 0 002-2V7a2 2 0 00-2-2H5a2 2 0 00-2 2v12a2 2 0 002
2z"></path></svg>
```

```
            <span>End Date</span>
          </div>
          <p className="text-sm text-gray-600 mt-1">The last day the
notification will be sent</p>
        </div>
      </div>
      <div className="flex justify-start space-x-4">
        <button className="text-blue-600 hover:text-blue-800 font-
medium" onClick={handleSaveNotification}>SAVE NOTIFICATION</button>
        <button className="text-blue-600 hover:text-blue-800 font-
medium">EXIT</button>
      </div>
    </div>
  );
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Проектування та розробка функціонального сповіщувача енергоефективності ”

Обсяг пояснювальної записки: 79 аркушів.

Дата закінчення роботи: 11 червня 2025 р.

Підпис студента _____