Вищий навчальний заклад

«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Система збору та обробки інформації «розумного будинку»

Виконав _____
(Підпис)

___Морозов Владислав Сергійович_____
(прізвище, ім'я, по батькові)

_____

Науковий керівник
____Добришин Юрій Євгенович_____ __
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Голова циклової комісії _____ Леонід УВАРОВ

(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Морозов Владислав Сергійович

1. Тема роботи Розробка системи збору та обробки інформації «розумного будинку» затверджена наказом по університету від «_» _____ 202_ р. № _____
2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:

1) цільова аудиторія – мешканці приватних будинків, квартир, адміністратори житлових систем;

2) функціональність – збір показників температури, вологості, стану освітлення, управління світлом, перегляд історії змін;

3) технічні вимоги – вебінтерфейс на js, бекенд (Node.js або ASP.NET Core), база даних jwt xss;

4) можливість розширення – додавання сенсорів руху, диму, протікання, підтримка мобільної версії;

5) використання сучасних принципів безпеки: XSS-захист, автентифікація, збереження бази окремо від вебресурсу.

4. Зміст пояснювальної записки:

1) **Розділ 1. Теоретична частина.** Аналіз сучасних платформ у сфері «розумного дому» (Home Assistant, Domoticz тощо), порівняння, обґрунтування вибору технологій для реалізації власного рішення.

2) **Розділ 2. Проєктування та розробка.**

Створення архітектури вебзастосунку, опис бази даних, реалізація функціоналу: моніторинг, керування, авторизація користувачів, безпека, інтерфейс.

3) **Розділ 3. Експериментальна частина.**

Тестування системи на різних пристроях, аналіз роботи сайту, опис ключових функцій, інструкція користувача, оцінка ефективності.

5. **Перелік графічного матеріалу:**

- Скріншоти інтерфейсу розробленої системи
- Схеми структури бази даних
- Діаграми класів, компонентів, розгортання
- Блок-схеми алгоритмів роботи з даними

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ ___Добришин.Ю.Є___

(підпис) (прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____ ___Морозов.В.С___

(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Об’єкт дослідження – система збору та обробки інформації «розумного будинку».

Мета роботи – розробка програмного забезпечення, яке дозволяє здійснювати збір, збереження, обробку та відображення параметрів середовища в системі «розумного дому», а також керування освітленням у реальному часі.

Кваліфікаційна робота містить результати проєктування та реалізації вебзастосунку для моніторингу температури, вологості, стану світла та історії змін параметрів. Проведено аналіз сучасних платформ (Home Assistant, Domoticz тощо), визначено оптимальний стек технологій (React, Next.js, PostgreSQL, REST API). Реалізовано модулі реєстрації, авторизації, управління пристроями та збереження історії в окремій базі даних. Здійснено тестування працездатності системи на різних пристроях.

Результати можуть бути впроваджені в системи автоматизації житла, офісних приміщень або освітніх установ.

РОЗУМНИЙ ДІМ, СЕНСОРИ, БАЗА ДАНИХ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, РЕАКТИВНИЙ ІНТЕРФЕЙС, ІСТОРІЯ ПАРАМЕТРІВ

ABSTRACT

The object of the study is a smart home information collection and processing system.

The purpose of the work is to develop a software solution that collects, stores, processes and displays temperature, humidity and lighting data in real time, as well as provides remote control over lighting.

The qualification work contains the results of designing and implementing a web-based system for smart home monitoring. The system includes modules for user authentication, real-time control, secure data transmission, and history logging. The technology stack includes React, Next.js, PostgreSQL and REST API. Comparative analysis of modern smart home platforms (Home Assistant, Domoticz) was conducted. Functionality was tested on multiple devices.

The results can be applied in residential, office or educational smart automation environments.

SMART HOME, SENSORS, DATABASE, CLIENT-SERVER ARCHITECTURE, REACTIVE INTERFACE, PARAMETER HISTORY

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

1. **SmartHome** – Вебзастосунок для збору, зберігання та обробки інформації в системі розумного будинку.
2. **Користувач** – Людина, яка керує параметрами розумного будинку через вебінтерфейс.
3. **Історія параметрів** – Журнал змін температури, вологості та стану світла.
4. **API** – Інтерфейс програмування додатків, що забезпечує взаємодію між фронтендом, бекендом і базою даних.
5. **Система моніторингу** – Програмний модуль для зчитування та відображення даних із сенсорів.
6. **База даних** – Сховище інформації про користувачів і показники датчиків.
7. **Температура** – Один із основних параметрів середовища, що контролюється в системі.
8. **Вологість** – Параметр середовища, що визначає відносну вологість у приміщенні.
9. **Освітлення** – Стан освітлювальних пристроїв, яким можна керувати через систему.
10. **Датчик** – Пристрій для зчитування фізичних параметрів (температури, вологості, руху, диму).
11. **Керування** – Функція вмикання/вимикання пристроїв через вебінтерфейс.
12. **Реєстрація** – Створення облікового запису користувача в системі.
13. **Авторизація** – Вхід до системи з перевіркою облікових даних.
14. **XSS** – Тип атаки, проти якого реалізований захист у системі.
15. **Інтерфейс користувача (UI)** – Вебсторінка, через яку здійснюється взаємодія з системою.
16. **Інтеграція** – Поєднання системи з базою даних, сенсорами або іншими сервісами.

17. **Логування** – Збереження дій користувача та змін у параметрах.

18. **Безпека** – Сукупність технічних заходів для захисту даних користувачів.

19. **Хмара** – Сервер або віртуальне середовище, де може зберігатися база даних.

20. **Інтервальна перевірка** – Автоматичне оновлення показників кожні кілька секунд.

21. **Резервна база** – Копія основної бази даних, яка захищає від втрати даних.

22. **Мобільна адаптація** – Здатність системи працювати на смартфонах та планшетах.

23. **Фронтенд** – Клієнтська частина системи, реалізована на React або Next.js.

24. **Бекенд** – Серверна частина системи, яка обробляє запити (наприклад, ASP.NET Core, Node.js).

ВСТУП

Актуальність завдання. У сучасному світі стрімкий розвиток цифрових технологій суттєво змінює стиль життя людей, проникаючи в усі сфери діяльності — від побуту до промисловості. Однією з ключових тенденцій останнього десятиліття є впровадження концепції «розумного будинку» (smart home), яка забезпечує новий рівень автоматизації, комфорту, безпеки та енергоефективності. Розумний будинок — це не просто набір підключених пристроїв, а інтегрована система, яка дає змогу централізовано контролювати важливі параметри середовища, зокрема температуру, вологість, освітлення, вентиляцію тощо. Завдяки автоматизованому управлінню користувач отримує змогу керувати життєвими процесами в оселі або офісі з будь-якої точки світу за допомогою вебінтерфейсу або мобільного застосунку.

Окремо варто зазначити важливість забезпечення зворотного зв’язку в таких системах — моніторинг змін параметрів у реальному часі дозволяє виявляти аномалії, оперативно реагувати на відхилення від норми та вживати необхідних заходів. Таким чином, розробка вебзастосунку для керування розумним будинком є надзвичайно актуальною задачею, оскільки вона поєднує в собі можливості дистанційного керування, візуалізації даних, захищеного доступу та інтеграції з різноманітними сенсорними пристроями. Подібні рішення мають широкі перспективи застосування не лише в житлових приміщеннях, а й у комерційній сфері — бізнес-центрах, навчальних установах, лікарнях, готелях та на об'єктах критичної інфраструктури.

Мета роботи. Основна мета цієї роботи полягає у створенні функціонального вебзастосунку, призначеного для збору, обробки та візуалізації даних із сенсорів у системі «розумного будинку». Застосунок повинен підтримувати інтерактивне керування пристроями, такими як освітлення або клімат-контроль, а також забезпечувати збереження історичних даних про зміну параметрів середовища (температура, вологість, рівень освітленості тощо). Додатково планується впровадити систему реєстрації та авторизації користувачів із захищеним доступом, що унеможливить несанкціонований

перегляд або зміну налаштувань. Реалізація такої системи стане вагомим внеском у розвиток технологій домашньої автоматизації та підвищення якості життя.

Завдання роботи:

- Здійснити комплексний аналіз сучасних рішень у сфері інтелектуальних систем для автоматизації житлових і комерційних приміщень, виявити їхні ключові характеристики, переваги, а також обмеження та недоліки.
- Розробити логічну структуру вебзастосунку, включно з проєктуванням архітектури програмного забезпечення, логіки взаємодії з пристроями та сервісами, а також розробкою бази даних для зберігання поточних і історичних даних.
- Створити функціональні модулі, що відповідають за збір інформації з сенсорів, управління виконавчими пристроями (освітленням, реле тощо), а також за обробку запитів користувача.
- Реалізувати систему реєстрації та авторизації користувачів з використанням сучасних механізмів захисту даних.
- Забезпечити надійне збереження історії зміни параметрів, а також реалізувати зручну систему візуалізації таких даних.
- Провести тестування системи на різних пристроях та в різних умовах з метою перевірки стабільності, продуктивності, адаптивності інтерфейсу та відмовостійкості.

Об’єкт дослідження. Процес створення інтелектуальної системи керування середовищем у розумному будинку з використанням вебтехнологій та сенсорної інфраструктури. Особливу увагу приділено інтеграції пристроїв, збору та обробці даних, взаємодії користувача з інтерфейсом та безпеці.

Предмет дослідження. Механізми та технічні рішення для реалізації функціональних можливостей програмного забезпечення, орієнтованого на дистанційний контроль та моніторинг параметрів середовища в системі розумного будинку.

Методи дослідження. У роботі застосовано низку методів: теоретичний аналіз і порівняння вже існуючих платформ для розумного будинку, що дозволяє виявити кращі практики; структурне та об'єктно-орієнтоване проєктування архітектури системи; моделювання та розробка бази даних; створення інтерфейсу користувача на основі сучасних фреймворків (наприклад, React, ASP.NET Core); тестування на етапі розробки та після завершення для перевірки стабільності та відповідності вимогам.

Практичне значення одержаних результатів. Розроблена система демонструє реальні можливості автоматизації керування побутовими процесами з використанням вебтехнологій. Її можна адаптувати для застосування в будь-яких приміщеннях, де необхідно забезпечити моніторинг стану середовища. Завдяки підтримці збереження історичних даних та надійному захисту інформації, вебзастосунок може бути використаний як базовий компонент для створення масштабованих і більш складних систем автоматизації. Запропоноване рішення здатне функціонувати стабільно навіть за наявності непередбачених збоїв на стороні клієнта або підключених пристроїв, що підтверджує його життєздатність у реальних умовах експлуатації.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

1.1. Аналіз існуючих рішень у сфері систем «розумного будинку»

У сучасних умовах стрімкого розвитку цифрових технологій та зростання попиту на автоматизацію побутових і комерційних процесів, концепція «розумного будинку» (Smart Home) набула особливої популярності. Її мета полягає у створенні інтелектуального середовища, яке здатне автономно або під контролем користувача керувати основними аспектами комфорту, безпеки й енергоефективності. Подібні системи вже не є лише інноваційною забаганкою — вони поступово стають стандартом у новобудовах, житлових комплексах, офісних приміщеннях та приватних будинках.

Системи «розумного будинку» дозволяють здійснювати моніторинг та управління параметрами середовища — зокрема температурою, вологістю, рівнем освітлення, станом вентиляції, безпекою приміщення тощо. Для реалізації цих функцій використовуються різноманітні цифрові платформи, які забезпечують обробку сигналів від сенсорів, активацію виконавчих пристроїв та зручну взаємодію з користувачем за допомогою вебінтерфейсів або мобільних застосунків. У результаті це сприяє підвищенню рівня автоматизації, зменшенню енергоспоживання, оптимізації витрат та загальному комфортному середовищу проживання.

На глобальному ринку вже існує велика кількість рішень у сфері SmartHome, кожне з яких має власні особливості, переваги та обмеження. Серед найбільш популярних можна виділити такі:

- Home Assistant — одна з найвідоміших платформ із відкритим вихідним кодом. Вона підтримує інтеграцію з тисячами пристроїв та сервісів (освітлення, термостати, камери, голосові помічники тощо). Система активно розвивається завдяки сильній спільноті, має широку документацію, а також підтримує гнучку автоматизацію через YAML-конфігурації або візуальний редактор. Її головною перевагою є можливість пристосування до будь-яких умов і сценаріїв використання, проте інтерфейс та налаштування можуть бути складними для новачків.

- Domoticz — легка та ресурсозберігаюча система, яка чудово підходить для локального розгортання на пристроях з обмеженими можливостями (наприклад, Raspberry Pi Zero). Платформа має простий вебінтерфейс, підтримує базову автоматизацію та може бути розширена за рахунок скриптів. Domoticz орієнтований на користувачів із мінімальними технічними знаннями, однак має менш активну спільноту та обмежені функції порівняно з іншими рішеннями.
- OpenHAB (Open Home Automation Bus) — платформа, що вирізняється високою масштабованістю і гнучкою модульною архітектурою. Вона підтримує інтеграцію з великою кількістю пристроїв і протоколів, дозволяє налаштовувати складні правила автоматизації, має розвинену екосистему плагінів. Система орієнтована на досвідчених користувачів або розробників, і часто використовується у великих об’єктах із високим рівнем складності.

Попри численні переваги вищезгаданих рішень, у кожної з платформ є і певні недоліки або обмеження, які варто враховувати при виборі технології для реалізації власної системи:

- Високий поріг входу для пересічного користувача — налаштування, конфігурація та підтримка часто вимагають технічної підготовки, знання терміналів, конфігураційних файлів та мережевих принципів.
- Необхідність наявності окремого серверного обладнання (локального або хмарного), що потребує додаткових ресурсів, часу на налаштування та підтримку стабільної роботи системи.
- Обмежена локалізація інтерфейсів, зокрема відсутність української мови в інтерфейсах або документації, що ускладнює користування для широкого загалу в Україні.
- Залежність від сторонніх сервісів або API, які можуть змінюватися, обмежуватися або взагалі припиняти роботу. Це створює ризики нестабільності, а також питання конфіденційності й безпеки даних.

У зв’язку з цим усе більше користувачів виявляють зацікавленість у простіших і доступніших рішеннях, які не потребують складної інсталяції, конфігурації або спеціального обладнання. Попит зростає на незалежні

вебзастосунки, які можна швидко розгорнути, налаштувати та використовувати без глибоких технічних знань.

Такі рішення повинні:

- забезпечувати базовий набір функцій (моніторинг, управління, сповіщення);
- бути інтуїтивно зрозумілими та мати адаптивний інтерфейс;
- підтримувати роботу без підключення до хмари (локальні сценарії);
- зберігати гнучкість для розширення та модифікацій у разі потреби.

Розробка власної системи такого типу дозволяє не лише уникнути типових обмежень готових платформ, а й створити рішення, повністю адаптоване під потреби локального користувача — з українською мовою, простим інтерфейсом, локальним зберіганням даних та модульною структурою.

1.2. Кращі практики розробки систем розумного будинку

У процесі створення інтелектуальних систем автоматизації побуту особливу увагу слід приділяти застосуванню кращих практик, які вже довели свою ефективність у світовій розробницькій спільноті. Розробка подібних систем — це не лише написання коду чи підключення сенсорів, а комплексний процес, що охоплює безпеку, зручність, надійність, масштабованість та можливість довготривалої підтримки. Дотримання перевірених підходів дозволяє створювати рішення, що залишаються актуальними, легко адаптуються до нових вимог і, що найголовніше, забезпечують високий рівень задоволеності користувачів.

Безпека даних — понад усе

Однією з ключових вимог до будь-якої сучасної системи, яка працює з мережею, є захист даних. У системах «розумного будинку» йдеться не просто про технічну інформацію, а про дані, які можуть прямо чи опосередковано свідчити про звички, присутність чи відсутність мешканців, їхній спосіб життя тощо.

Саме тому особливу увагу варто приділити таким аспектам:

- використання протоколу HTTPS для захищеної передачі даних між клієнтом і сервером;
- застосування механізму JWT (JSON Web Token) для автентифікації, який дозволяє уникнути збереження стану сесії на сервері;
- реалізація захисту від XSS (міжсайтового скриптингу) та CSRF (міжсайтових запитів), а також очищення усіх вхідних даних;
- впровадження ролевої моделі доступу, яка дозволяє диференціювати права користувачів та зменшити ризики несанкціонованих дій.

Простота та зручність інтерфейсу — запорука користувацького досвіду

Користувач — головна фігура будь-якої системи. Якщо інтерфейс складний, перевантажений елементами чи незрозумілий, жодна потужність функціоналу не компенсує негативне враження.

Тому інтерфейс повинен:

- бути адаптивним — зручним як на комп’ютері, так і на мобільних пристроях;
- мати інтуїтивну навігацію, зрозумілі піктограми, логічно структуровані розділи;
- підтримувати відображення даних у режимі реального часу, наочні графіки, діаграми, індикатори;
- бути локалізованим — українська мова за замовчуванням, можливість перемикання мов.

Логування — джерело діагностики та аналітики

Збір історичних даних — це не просто функціональність для «галочки». Це основа для діагностики системи, аналізу поведінки пристроїв та виявлення нестандартних ситуацій.

Журнали подій дозволяють:

- бачити, коли і який пристрій був активований, ким і за яких умов;
- фіксувати зміни кліматичних параметрів — температура, вологість, рівень CO_2;

- формувати автоматизовані звіти, які можуть бути корисними як у побуті, так і для технічного обслуговування;
- забезпечити відповідальність і контроль, особливо в багатокористувацькому середовищі.

Гнучкість архітектури — основа для майбутнього зростання

Жодна система не є статичною. З часом користувач захоче додати нові пристрої, змінити логіку сценаріїв, оновити інтерфейс. Тому надзвичайно важливо ще на етапі проектування передбачити модульність і гнучкість структури.

Це означає:

- поділ системи на окремі незалежні модулі, які легко оновлюються або замінюються;
- використання уніфікованих API, які дозволяють додавати нові функції без перезапуску системи;
- підтримку плагінів, які можуть розширити функціональність за потреби;
- зберігання параметрів у зовнішніх конфігураційних файлах, доступних для редагування.

Стабільність та автономність — безпека на всі випадки життя

Важливо розуміти, що система повинна працювати не тільки в ідеальних умовах. Відсутність інтернету, перебої з електроенергією, збої на сервері — усе це не має стати причиною повного зупинення.

Для цього варто:

- передбачити локальне збереження критичних даних, наприклад у JSON або SQLite;
- реалізувати автоматичне резервне копіювання конфігурації;
- забезпечити роботу в офлайн-режимі, де основні функції не залежать від сервера;
- використовувати перевірку стану пристроїв з можливістю самовідновлення з’єднання.

Масштабованість — з квартири до кампусу

Система повинна бути гнучкою не лише функціонально, але й структурно. Тобто — її потрібно проектувати так, щоб у майбутньому можна було безболісно додати ще одну кімнату, поверх, або цілий будинок.

Для цього:

- архітектура має підтримувати масштабування вшир і вглиб (горизонтальне й вертикальне);
- пристрої повинні спілкуватися через уніфіковані протоколи (MQTT, WebSocket, REST API);
- важливо мати центральний контролер або хаб, який координує роботу всієї мережі;
- передбачити можливість інтеграції зі сторонніми системами — камерами, системами сигналізації, лічильниками тощо.

Усі перелічені практики не є просто побажаннями чи рекомендаціями — це базові підвалини, без яких створення ефективної, надійної та сучасної системи автоматизації буде неможливим. Дотримання цих принципів дозволить створити рішення, яке не тільки задовольнить поточні потреби користувача, але й залишиться актуальним у майбутньому.

1.3. Вибір платформи, архітектури та технологій для реалізації

Проєктування і розробка системи «розумного будинку» передбачає обґрунтований вибір відповідного технологічного стеку, архітектурного підходу та інструментів реалізації, які здатні забезпечити необхідний баланс між продуктивністю, безпекою, зручністю використання та можливістю подальшого масштабування. У даному проєкті були застосовані сучасні вебтехнології, які дозволяють створити функціональну, легку й адаптивну систему керування побутовими пристроями.

Фронтенд (клієнтська частина)

Головним завданням фронтенду є забезпечення комфортної та інтуїтивно зрозумілої взаємодії користувача з системою. Для реалізації інтерфейсу було

обрано чистий JavaScript, без використання важких фреймворків типу React або Angular. Такий підхід дозволяє:

- досягти мінімального часу завантаження сторінки;
- мати повний контроль над структурою та логікою інтерфейсу;
- гнучко модифікувати функціонал без залежності від сторонніх бібліотек;
- за необхідності — інтегрувати додаткові JS-бібліотеки, зокрема для побудови графіків, діаграм, візуалізації даних з сенсорів або динамічних елементів управління.

Бекенд (серверна частина)

Для обробки запитів та реалізації логіки взаємодії між пристроями і користувачем використовується платформа Node.js — сучасне серверне середовище виконання JavaScript-коду. Основні переваги Node.js:

- асинхронна модель обробки запитів, що забезпечує високу продуктивність;
- здатність обслуговувати численні одночасні запити з мінімальними затримками;
- велика кількість готових npm-пакетів для реалізації додаткових функцій (вебсервери, безпека, логування тощо);
- простота розгортання на більшості хостинг-платформ.

Бекенд відповідає за логіку роботи з API, обробку станів пристроїв, перевірку прав доступу, збереження та читання даних, а також виконання автоматичних сценаріїв.

Зберігання даних

У якості бази даних було обрано формат JSON, який дозволяє:

- зберігати структуровану інформацію про стан системи (освітлення, температура, вологість тощо);
- швидко здійснювати запис і читання даних без необхідності налаштування окремого СУБД;
- легко експортувати, копіювати чи резервувати інформацію;

- підтримувати гнучку структуру, що зручно при додаванні нових параметрів.

API (інтерфейс взаємодії компонентів)

Обмін даними між фронтендом і бекендом здійснюється за допомогою REST API — одного з найпоширеніших стандартів у сучасній веброзробці. Кожен HTTP-запит (GET, POST, PUT, DELETE) має чітке призначення та викликає відповідну дію на сервері.

Це забезпечує:

- швидкий та передбачуваний обмін даними між клієнтом і сервером;
- можливість легко додавати нові функції або параметри до системи без зміни структури запитів;
- масштабованість: у майбутньому API може бути використаний для створення мобільного застосунку або інтеграції з іншими сервісами.

Безпека системи

Зважаючи на потенційну важливість і конфіденційність даних, що циркулюють у системі, питання безпеки є надзвичайно актуальним. Реалізовано такі заходи:

- HTTPS — використання захищеного протоколу для всієї взаємодії з сервером, що виключає можливість перехоплення даних;
- JWT (JSON Web Token) — технологія, яка дозволяє створити надійний механізм авторизації з можливістю гнучкого керування ролями та правами користувачів;
- Захист від XSS-атак — очищення та перевірка введення, що запобігає виконанню шкідливих скриптів;
- Перевірка заголовків і походження запитів (CORS, Origin) — обмеження доступу до API тільки для авторизованих клієнтів.

Архітектура рішення

Загальна структура реалізована за принципом класичної клієнт-серверної архітектури. Це означає:

- Фронтенд (клієнтська частина) відповідає за відображення даних і взаємодію з користувачем;
- Бекенд (серверна частина) — за логіку обробки запитів і взаємодію з даними;
- Файл JSON виконує роль локальної бази даних.

Така архітектура має низку переваг: вона є простою для реалізації, зручною для підтримки, добре масштабується й дозволяє гнучко змінювати окремі компоненти без впливу на всю систему. У перспективі можливе розширення:

- заміна JSON на повноцінну базу даних (MySQL, MongoDB);
- додавання резервного копіювання та журналювання;
- побудова модульної структури з можливістю інтеграції нових блоків керування (опалення, безпека, освітлення тощо).

1.4. Світові тенденції у розвитку SmartHome-систем

У сучасному світі інноваційні технології все активніше змінюють спосіб життя людини, забезпечуючи нові рівні зручності, безпеки та енергоефективності. Системи «розумного будинку» (SmartHome) з кожним роком стають дедалі популярнішими, перетворюючись із нішевого рішення на один із ключових напрямів розвитку цифрової інфраструктури житла.

Згідно з аналітичними звітами компаній Statista, McKinsey & Company, IDC та Gartner, глобальний ринок SmartHome-технологій демонструє стабільне зростання, із середньорічним темпом приросту понад 20%. Очікується, що вже у найближчі 2–3 роки кількість пристроїв, підключених до Інтернету речей (IoT), перевищить 50 мільярдів одиниць, більшість з яких будуть використовуватись саме в побуті. Така динаміка свідчить про масове впровадження цифрових рішень у повсякденне життя людей.

Однією з провідних тенденцій є перехід від локальних до хмарних рішень. Хмарні платформи забезпечують доступ до домашніх пристроїв із будь-якої точки світу, дозволяють здійснювати моніторинг у реальному часі, керувати сценаріями та зберігати аналітичні дані. Проте паралельно зростає попит на автономні локальні системи, які не залежать від постійного інтернет-з’єднання

чи сторонніх серверів — це особливо актуально у регіонах із нестабільним доступом до мережі або підвищеними вимогами до конфіденційності.

Серед ключових світових напрямів розвитку SmartHome можна виокремити:

- Глибока інтеграція з голосовими асистентами — Amazon Alexa, Google Assistant, Apple Siri. Голосове управління стає стандартом, зокрема для людей з обмеженими можливостями або у ситуаціях, коли фізичне керування недоступне.
- Використання штучного інтелекту та машинного навчання для побудови адаптивних сценаріїв. Системи навчаться передбачати дії користувача, наприклад — автоматично вмикати світло, коли мешканець заходить у кімнату, або регулювати температуру в залежності від погодних умов та звичок мешканців.
- Інтеграція з Smart Grid — інтелектуальними електромережами, які дозволяють оптимізувати енергоспоживання, враховуючи тарифи, пікові навантаження та навіть заряджання електромобілів.
- Орієнтація на екологічність: встановлення датчиків витоку води, контролю вуглекислого газу, автоматичне вимкнення пристроїв у разі відсутності людей у приміщенні — усе це сприяє енергоощадності та зменшенню вуглецевого сліду.
- Підвищення кібербезпеки. У зв’язку з великою кількістю підключених пристроїв, однією з тенденцій є розробка надійних механізмів шифрування, автентифікації та виявлення загроз для запобігання несанкціонованому доступу.

У Європі, США, Китаї вже активно розвиваються SmartHome-платформи від провідних корпорацій — Google Nest, Apple HomeKit, Samsung SmartThings, Xiaomi Mi Home, які пропонують комплексні рішення з урахуванням локальної інфраструктури, мовної підтримки та побутових особливостей.

Ситуація в Україні

В Україні сегмент «розумного житла» також демонструє позитивну динаміку. Хоча темпи впровадження дещо повільніші через економічні фактори, проте:

- Зростає попит на DIY-системи (Do It Yourself), які користувачі самостійно встановлюють за допомогою доступних модулів (ESP32, Arduino, Raspberry Pi тощо).
- У новобудовах все частіше передбачають вбудовані системи автоматизації — з можливістю централізованого керування освітленням, кліматом, безпекою.
- Приватні домовласники виявляють інтерес до автономних рішень, які не вимагають підписки або зовнішніх серверів, що є особливо актуальним у сільській місцевості та малих містах.
- Університети та IT-спільноти просувають освітні ініціативи зі створення українських аналогів SmartHome-систем із відкритим кодом.

Таким чином, світові тенденції свідчать про поступовий перехід до розумного, автономного, екологічного і безпечного житла, а Україна впевнено рухається у напрямі адаптації цих трендів до локальних умов. У майбутньому варто очікувати розширення інфраструктури, зниження вартості обладнання та зростання масової зацікавленості населення у впровадженні подібних технологій у побут.

1.5. Обґрунтування вибору технологій і підходів

У процесі створення інтелектуальної системи керування «розумним будинком» критично важливим є не лише формулювання функціональних та нефункціональних вимог, але й ретельний підбір технологічного стеку, який дозволить реалізувати поставлені цілі максимально ефективно. Обрані інструменти мають відповідати ряду ключових критеріїв, зокрема:

- простота впровадження та налаштування, особливо на початкових етапах розробки;
- відкритість технологій та відсутність залежності від пропрієтарних платформ;

- наявність великої активної спільноти підтримки, що дозволяє швидко знаходити рішення типових проблем;
- стабільність і надійність обробки запитів та зберігання даних;
- можливість масштабування — як по функціоналу, так і по кількості підключених пристроїв у майбутньому.

Для досягнення оптимального балансу між продуктивністю, гнучкістю та простотою підтримки, було проведено порівняльний аналіз декількох підходів до реалізації серверної частини:

- Python + Flask або FastAPI — зручні, легкі у використанні рішення для створення REST-серверів. Їх перевагою є висока читабельність коду та підтримка асинхронності (особливо в FastAPI). Проте вони вимагають додаткових налаштувань для інтеграції з фронтендом, а також встановлення сторонніх пакетів для роботи з IoT-пристроями.
- PHP + MySQL — класичне поєднання, яке широко застосовується у веброзробці. Воно добре підходить для розробки інформаційних сайтів, але є менш гнучким у контексті обробки асинхронних подій або створення масштабованих IoT-рішень у реальному часі.
- Node.js — платформа, яка дозволяє створювати швидкі, асинхронні та масштабовані вебзастосунки. Її сильна сторона — неблокуюча обробка запитів, що є особливо актуальним при роботі з численними пристроями, які надсилають дані одночасно. Крім того, Node.js підтримується великою кількістю бібліотек (через npm), що значно пришвидшує розробку.

Після аналізу переваг і недоліків було надано перевагу Node.js, як найбільш гнучкому, продуктивному та адаптивному середовищу для побудови систем SmartHome. Його асинхронна природа дозволяє одночасно обробляти численні запити від сенсорів, керуючих пристроїв і користувацьких клієнтів, що критично важливо для роботи у реальному часі.

Зберігання даних

На етапі створення MVP (мінімально життєздатного продукту) було прийнято рішення не використовувати повноцінну СУБД, а застосувати локальне зберігання даних у форматі JSON. Це забезпечує:

- мінімальну конфігурацію — файл створюється автоматично та не потребує налаштування БД-сервера;
- зручність у тестуванні та налагодженні;
- простоту редагування даних вручну у випадку помилок чи необхідності змін;
- повну сумісність з API, де JSON є основним форматом передачі даних.

У подальшому, за умови розширення функціоналу, передбачена можливість міграції до більш складної системи зберігання — наприклад, MongoDB або PostgreSQL, залежно від вимог щодо структури даних та продуктивності.

Клієнтська частина (JavaScript)

Для створення інтерфейсу було обрано чистий JavaScript, без використання важких фреймворків (таких як React або Angular), що дозволило досягти низки переваг:

- повний контроль над логікою інтерфейсу, без обмежень шаблонів або життєвого циклу компонентів;
- можливість поступового розширення функціоналу, включно з додаванням сторонніх бібліотек або переходом до складніших рішень (наприклад, Vue або React) у разі масштабування проєкту.

1.6. Суспільне та економічне значення розумних будинків

Системи «розумного будинку» (SmartHome) — це не лише приклад технологічного прогресу, а й важливий інструмент соціальних і економічних змін, що вже сьогодні впливають на різні аспекти життя людини. Їх застосування виходить за межі звичайного підвищення зручності у побуті, адже вони сприяють ефективнішому використанню ресурсів, забезпеченню безпеки, підтримці соціально вразливих верств населення, а також стимулюють розвиток

цілих галузей економіки. Тобто, ці технології мають системний і багаторівневий вплив — від приватного домогосподарства до національного рівня.

На побутовому рівні системи SmartHome дозволяють значно зменшити витрати на комунальні послуги завдяки впровадженню автоматизованого керування освітленням, кліматом, побутовими приладами. Усе це дозволяє оптимізувати споживання енергії відповідно до реальних потреб користувача. Наприклад, світло вмикається лише у присутності людей в кімнаті, опалення автоматично регулюється залежно від зовнішньої температури, а прилади вимикаються у разі відсутності мешканців. Виявлення витоків води, газу або диму на ранніх етапах дозволяє запобігти аваріям, які можуть завдати значних матеріальних збитків. Така функціональність особливо важлива в умовах зростання цін на ресурси, а також в періоди енергетичної нестабільності, які характерні для багатьох країн, включно з Україною.

Значення таких систем також важко переоцінити у соціальному вимірі. Технології розумного будинку надають нові можливості для людей з обмеженими фізичними можливостями, літніх осіб, а також тих, хто потребує дистанційного нагляду. Завдяки автоматизації та можливості керування голосом або зі смартфону, вони можуть залишатися незалежними, виконувати побутові дії без сторонньої допомоги, а також отримувати своєчасні сповіщення або допомогу у випадку небезпеки. Це не лише підвищує якість життя, а й сприяє психологічному комфорту як самих користувачів, так і їхніх родин.

Окрім безпосередньої користі для кінцевих користувачів, впровадження SmartHome-рішень має потужний стимулювальний ефект для економіки. Розвиток подібних систем супроводжується зростанням попиту на комплектуючі — сенсори, контролери, виконавчі пристрої — а також на послуги зі встановлення, налаштування та обслуговування обладнання. Це створює нові робочі місця, сприяє розвиткові малого та середнього бізнесу, а також збільшує інвестиції в суміжні сфери, як-от програмування, телекомунікації, системна інтеграція. Більше того, поширення SmartHome-технологій тісно пов’язане з

цифровізацією суспільства, зростанням технічної обізнаності громадян та підвищенням інтересу до STEM-освіти.

Варто також наголосити на екологічному аспекті. Зменшення надлишкового енергоспоживання, оптимізація роботи опалювальних і вентиляційних систем, точний контроль витрат води — усе це сприяє зниженню загального навантаження на довкілля. Крім того, багато сучасних систем підтримують роботу з альтернативними джерелами енергії, такими як сонячні панелі чи вітрогенератори, що робить їх актуальними у контексті боротьби зі зміною клімату та формування сталого розвитку.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Вимоги до інтерфейсу користувача

Інтерфейс користувача — це перший елемент взаємодії між системою «розумного будинку» та її користувачем. Саме від якості інтерфейсу залежить зручність роботи, швидкість реакції користувача на зміни параметрів, а також загальне враження від використання програмного продукту.

Основні вимоги до інтерфейсу:

- **Простота**. Користувач повинен інтуїтивно розуміти, як керувати системою. Інтерфейс має містити чіткі кнопки керування світлом, індикатори поточних значень температури та вологості, а також зручну навігацію між розділами.
- **Адаптивність**. Інтерфейс повинен коректно працювати на різних пристроях, таких як смартфони, планшети, ноутбуки або стаціонарні комп’ютери. Важливо, щоб компоненти не виходили за межі екрану, а керування було зручним незалежно від розміру дисплея.
- **Наглядність**. Кожен параметр повинен бути представленим у зручній формі: наприклад, температура — у градусах з кольоровим індикатором (зелений — норма, червоний — перегрів), стан світла — у вигляді іконки «увімкнено/вимкнено».
- **Зворотний зв’язок**. Після кожної взаємодії користувача (наприклад, натискання кнопки для зміни стану світла) система повинна негайно повідомляти про результат — як візуально, так і текстово (наприклад: “Світло увімкнено”).
- **Швидкодія та мінімальні затримки**. Оновлення даних має відбуватися в реальному часі без потреби перезавантаження сторінки. Це особливо важливо для відображення актуального стану сенсорів.

2.2 Розробка алгоритмів роботи системи моніторингу

Система моніторингу відповідає за регулярне зчитування та збереження параметрів довкілля. Вона є критично важливою складовою, оскільки на основі її роботи формується не лише поточний стан, а й історія змін.

Ключові етапи розробки алгоритму:

1. **Читання з сенсорів.**

Система через заданий проміжок часу ініціює автоматичний запит до підключених сенсорів. До прикладу, це можуть бути сенсори температури, вологості, освітлення або інші цифрові модулі. Інтервал опитування налаштовується — оптимальним вважається значення 5–10 секунд, що дозволяє забезпечити актуальність даних для користувача і водночас уникнути надмірного навантаження на систему. Запит здійснюється у фоновому режимі, без участі користувача.

2. **Збереження значень.**

Після кожного запиту система отримує числові значення з сенсорів, які одразу ж зберігаються у структурованому форматі JSON. До кожного запису автоматично додається мітка часу, що дозволяє точно визначити момент фіксації. Такий формат є зручним для обробки та подальшого аналізу — його легко використовувати для побудови графіків, перегляду змін за певний період або передачі в інші сервіси. Зберігання здійснюється локально, без потреби зовнішніх баз даних.

3. **Виявлення відхилень.**

Одразу після зчитування даних система порівнює значення з встановленими пороговими межами. Якщо зафіксовано перевищення допустимих параметрів — наприклад, температура піднімається вище 30 °C або вологість виходить за межі норми — запускається обробник події. У відповідь система може автоматично надіслати сповіщення користувачу, вивести повідомлення на екран або активувати виконавчі пристрої (наприклад, увімкнути охолодження чи вентиляцію).

4. **Оновлення інтерфейсу.**

Дані, отримані з сенсорів, миттєво передаються до вебінтерфейсу користувача. Інформація відображається у вигляді числових значень або графічних елементів, що постійно оновлюються в режимі реального часу. Це дає змогу користувачу бачити зміну показників без затримки та оперативно

реагувати на ситуацію. Передача даних здійснюється без перезавантаження сторінки, що підвищує зручність взаємодії.

5. **Ведення історії.**

Для кожного параметра ведеться повноцінна історія змін, яка зберігається в архіві та може бути використана для статистичного аналізу. Історичні дані доступні для візуалізації — наприклад, у вигляді лінійних графіків, що показують зміну температури протягом доби, тижня або місяця. Це дає змогу користувачу аналізувати динаміку параметрів, виявляти закономірності та коригувати сценарії керування на основі реальних тенденцій.

2.3 Реалізація функціоналу збору даних і керування пристроями

Основна функція системи — не лише фіксація параметрів, але й керування пристроями. Це дозволяє автоматизувати побутові процеси та створює комфорт для користувачів.

Моніторинг:

- Система фіксує температуру і вологість з відповідних сенсорів. Дані обробляються та передаються у веб-інтерфейс.
- Показники зберігаються в хронологічному порядку у JSON-файл.
- Дані оновлюються в реальному часі без перезавантаження сторінки.

Керування пристроями:

- Користувач може натисканням кнопки вмикати або вимикати освітлення.
- Стан реле зберігається у JSON-файлі й синхронізується з інтерфейсом.
- Система реагує миттєво: зміна стану відображається у вигляді кольору, повідомлення чи піктограми.

Технічна реалізація:

- Клієнтська частина (інтерфейс) розроблена на чистому JavaScript.
- Серверна частина — Node.js, яка обробляє запити через REST API.
- Взаємодія відбувається через HTTP-запити, що дозволяє масштабувати систему або інтегрувати її з іншими сервісами.

2.4 Інтеграція з базою даних і забезпечення інформаційної безпеки

У рамках поточного етапу реалізації проєкту було прийнято рішення використовувати просту, але ефективну модель зберігання даних — локальний JSON-файл. Такий підхід дозволяє швидко та без зайвих ускладнень організувати структуру збереження параметрів системи, не вдаючись до розгортання повноцінної реляційної бази даних. Це особливо актуально для MVP-рівня системи, коли головне — стабільність роботи, доступність і простота налаштування. JSON-файл легко читається, підтримує вкладені структури та ідеально підходить для інтеграції з Node.js, що робить його зручним у використанні як для збереження поточних даних, так і для ведення історії.

Інтеграція з даними:

• Всі ключові параметри системи, зокрема температура, вологість, стан освітлення, а також інші змінні, які можуть бути додані згодом, зберігаються у JSON у чітко структурованому вигляді, що дозволяє швидко отримувати доступ до будь-якого значення.

• Серверна частина на Node.js працює з цим файлом безпосередньо, що дозволяє зчитувати актуальні значення для відображення у вебінтерфейсі, а також формувати відповіді на запити користувача за кілька мілісекунд.

• Кожна зміна в системі — наприклад, зміна температури або ввімкнення освітлення — автоматично призводить до оновлення файлу. Разом із новим значенням записується мітка часу, що дозволяє вести повноцінну історію змін і зберігати послідовність подій.

• Завдяки простоті формату дані можна легко експортувати, резервувати або конвертувати у вигляд, придатний для подальшої обробки чи візуалізації — наприклад, побудови графіків або створення звітів.

Інформаційна безпека:

• Уся взаємодія користувача з сервером, включно з запитами на зчитування або зміну параметрів, відбувається через захищений протокол HTTPS. Це забезпечує шифрування інформації на етапі передачі, що захищає від

перехоплення даних зловмисниками під час роботи в публічних мережах або через незахищені канали.

• Для контролю доступу до функціональності системи використовується механізм автентифікації через JWT (JSON Web Token). Кожен користувач після входу отримує унікальний токен, який додається до всіх подальших запитів. Таким чином, сервер чітко ідентифікує, хто саме здійснює певну дію, і дозволяє чи забороняє доступ залежно від прав.

• Вхідні дані, які передаються через форму або API-запит, проходять обов'язкову перевірку та очищення. Це дозволяє захиститися від XSS-атак, SQL-ін'єкцій (якщо база буде впроваджена у майбутньому) та інших типів ін'єкційного впливу.

• Користувачі, які не автентифіковані або не мають відповідних прав, не мають доступу до функцій керування пристроями, не можуть змінювати параметри системи або переглядати чутливі дані.

2.5. Забезпечення масштабованості та гнучкості системи

Масштабованість є однією з найважливіших характеристик сучасних програмних систем, особливо у сфері «розумного будинку», де кількість пристроїв може постійно змінюватися, а вимоги користувачів — зростати. У процесі проєктування системи було закладено потенціал до безперешкодного розширення функціональності та кількості компонентів без погіршення продуктивності чи стабільності.

Для досягнення цього було обрано модульну архітектуру, що дозволяє ізолювати окремі функціональні частини системи. Наприклад, обробка даних від сенсорів, робота інтерфейсу користувача, система сповіщень, модулі безпеки та журналювання — усі вони реалізовані як незалежні компоненти, що взаємодіють через чітко визначені API. Такий підхід дає змогу вносити зміни в один модуль без ризику порушити роботу всієї системи.

У випадку розширення — наприклад, при додаванні нових приміщень, Усі пристрої (сенсори, актуатори, контролери) спілкуються з сервером через стандартизовані запити що забезпечує високу адаптивність до змін архітектури.

Горизонтальне масштабування системи забезпечується можливістю запуску кількох серверів або мікросервісів, які розподіляють навантаження між собою. Таким чином, платформа залишається стабільною навіть при великій кількості одночасних запитів або активних користувачів.

Окрема увага приділялася гнучкості системи — вона проявляється у легкості додавання нових типів пристроїв або функцій. Наприклад, до системи можна без складних модифікацій інтегрувати нові сенсори (температури, вологості, диму), смарт-пристрої (розетки, термостати, лампи) чи механізми сповіщення (push, email). Завдяки використанню універсальних структур зберігання даних (наприклад, у форматі JSON) усі конфігурації можна легко редагувати та зберігати у вигляді, придатному до розширення.

Загалом, реалізована архітектура забезпечує довготривалу актуальність та адаптивність системи. Вона дозволяє розгортати рішення як у рамках невеликої квартири, так і в масштабах великого будинку чи офісного приміщення, залишаючись стабільною, ефективною та зручною для подальшого розвитку.

2.6. Розширення можливостей керування

Функціональність керування в системі «розумного будинку» відіграє одну з ключових ролей, оскільки саме вона забезпечує безпосередню взаємодію користувача з фізичними пристроями в його повсякденному середовищі. Ефективне, швидке й інтуїтивно зрозуміле керування є визначальним чинником для формування позитивного досвіду використання системи та сприйняття її як надійного інструменту, що сприяє покращенню побуту, підвищенню рівня безпеки, а також оптимізації енергоспоживання.

На поточному етапі реалізації система підтримує базовий набір функцій — увімкнення та вимкнення освітлення, а також (за наявності відповідного обладнання) можливість регулювання яскравості через димери. Це дозволяє користувачу виконувати базові дії з управління світловими умовами у приміщенні, налаштовуючи їх під власні потреби в режимі реального часу. При цьому система зберігає поточний стан кожного пристрою та забезпечує

синхронізацію між фізичним станом обладнання та його візуальним представленням в інтерфейсі.

Завдяки гнучкій, модульній архітектурі система легко розширюється під нові типи пристроїв і сценаріїв. Підключення додаткових елементів можливе без повного переписування коду — достатньо реалізувати нові обробники запитів і внести незначні зміни до конфігурацій. У перспективі передбачено інтеграцію з низкою додаткових типів обладнання, що значно розширить функціональні можливості системи.

Зокрема, планується реалізація керування:

• системами опалення та кондиціонування — зміна температурних параметрів через інтерфейс або автоматично, з використанням даних із сенсорів і підключенням до API пристроїв (наприклад, Wi-Fi кондиціонерів або смарт-котлів);

• моторизованими шторами та жалюзі — відкривання і закривання за розкладом, у певний час доби або при зміні рівня зовнішнього освітлення;

• водопостачанням — керування електронними клапанами, зокрема для аварійного перекриття води у разі витоку або для реалізації сценаріїв економії води;

• системами безпеки — підключення камер відеоспостереження, розумних замків, сенсорів відкриття дверей і руху, що дозволить створити повноцінну інфраструктуру контролю доступу.

Окремим напрямком розвитку є інтеграція з голосовими помічниками, такими як Google Assistant або Apple Siri. Завдяки цьому користувачі зможуть виконувати ключові дії — наприклад, керування освітленням, запуск сценаріїв, перевірка стану пристроїв — за допомогою голосових команд. Така можливість значно підвищує доступність системи, особливо для людей з обмеженими фізичними можливостями, або у ситуаціях, коли керування через інтерфейс є незручним, наприклад, коли руки зайняті.

Ще одним перспективним напрямом є впровадження інтелектуальних сценаріїв автоматизації, які запускаються при настанні певних умов. Прикладами таких сценаріїв можуть бути:

• автоматичне вимкнення освітлення о 23:00 у разі відсутності активності;

• увімкнення обігрівача при зниженні температури нижче встановленого порогу (наприклад, 18 °C);

• відкривання штор при сході сонця або у визначений час;

• надсилання сповіщення при фіксації руху в нічний час або за відсутності користувача вдома.

Такі сценарії дозволяють підвищити автономність системи, мінімізують потребу у щоденному ручному керуванні та дають змогу створити персоналізоване середовище, що відповідає ритму життя конкретного користувача. Вони також покращують енергоефективність, знижують втрати ресурсів та забезпечують додатковий рівень комфорту.

Усі параметри сценаріїв і налаштування пристроїв зберігаються у форматі, зручному для подальшої модифікації — найчастіше у вигляді конфігураційних JSON-файлів. Такий підхід дозволяє не лише гнучко змінювати логіку роботи, але й легко адаптувати систему до нових пристроїв чи вимог без складних втручань у код. У подальшому планується реалізація візуального редактора сценаріїв, що дозволить користувачу створювати власні логіки автоматизації за принципом "якщо – то", без потреби в знаннях програмування.

2.7. Перспективи переходу на мобільний застосунок

Попри те, що поточна версія вебзастосунку вже є повноцінною адаптивною системою, яка коректно відображається та функціонує на мобільних пристроях завдяки адаптивній верстці, перехід до створення нативного мобільного застосунку виглядає цілком логічним і стратегічно доцільним етапом розвитку проєкту. Такий крок дозволить не лише підвищити рівень зручності для користувача, а й глибше інтегруватися у повсякденне цифрове середовище, адже смартфон сьогодні є основним інструментом взаємодії з цифровими сервісами для більшості людей.

Мобільний застосунок відкриває ширші можливості для інтеграції з апаратними функціями пристрою, що у вебверсії є або недоступними, або суттєво обмеженими. Зокрема, однією з ключових функцій, яка планується до впровадження, є система push-сповіщень, що дозволятиме миттєво інформувати користувача про критичні зміни в системі — наприклад, перевищення порогу температури, зафіксоване спрацювання датчика диму чи руху, втрату з’єднання з певним пристроєм або спробу несанкціонованого доступу. Такий рівень оперативності є критично важливим для підвищення безпеки та швидкої реакції на загрози.

Додаткову перевагу становить можливість використання біометричної авторизації — через сканер відбитків пальців або технологію розпізнавання обличчя, що вже реалізовано на більшості сучасних мобільних пристроїв. Це не лише значно підвищить рівень безпеки доступу до системи, а й зробить процес входу більш зручним і швидким. Біометрія фактично виключає ймовірність несанкціонованого використання додатку сторонніми особами.

Ще одним важливим функціональним розширенням стане використання геолокації. Мобільний застосунок зможе відстежувати положення користувача та автоматично вмикати певні сценарії при наближенні до дому або його залишенні. Наприклад, коли користувач наближається до будинку — автоматично вмикається зовнішнє або внутрішнє освітлення, регулюється температура або розблоковуються двері. Така логіка автоматизації сприяє не лише зручності, а й створенню дійсно "розумного" середовища, що реагує на дії людини в реальному часі.

У випадку тимчасової відсутності інтернету, мобільний додаток також зможе функціонувати автономно, використовуючи локальний кеш параметрів та станів пристроїв. Це дозволить зберігати базову функціональність системи навіть у разі втрати з’єднання, що особливо актуально для користувачів у регіонах з нестабільним покриттям або під час виїздів за місто.

Також важливою перевагою є можливість централізованого керування правами доступу та створення окремих профілів для кожного мешканця.

Наприклад, дитина може мати обмежений доступ лише до освітлення у своїй кімнаті, тоді як адміністратор — повний контроль над усіма модулями. Такий підхід забезпечує не тільки гнучкість, але й безпеку користування системою в умовах спільного доступу.

Окрім базових функцій управління, мобільний застосунок дозволить вивести на новий рівень користувацький досвід за рахунок додаткових сервісів. Передбачається можливість перегляду розширеної аналітики за історією роботи пристроїв, побудови графіків споживання енергії, температурних змін, звітів про спрацювання автоматичних сценаріїв тощо. Крім того, застосунок стане зручним каналом для звернення до служби технічної підтримки, де користувач зможе надсилати запити або отримувати рекомендації щодо вирішення проблем.

У сукупності, створення мобільного застосунку не лише забезпечить додаткову функціональність, а й перетворить систему на повноцінну мобільну екосистему керування розумним будинком. Це дозволить зробити сервіс більш персоналізованим, зручним, безпечним і доступним у будь-який момент, незалежно від місцезнаходження користувача. Таким чином, мобільний застосунок стане не просто доповненням, а невід’ємною частиною сучасної інфраструктури SmartHome, що відповідатиме високим очікуванням і потребам кінцевих користувачів.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ОЦІНЮВАННЯ

3.1. Тестування системи на різних пристроях і браузерах

Після завершення етапу реалізації основного функціоналу системи «розумного будинку» було організовано комплексне тестування з метою оцінки її стабільності, зручності використання, адаптивності та відповідності очікуваному функціоналу. Тестування охопило широкий спектр пристроїв, платформ і веббраузерів для перевірки кросбраузерності, коректності відображення елементів інтерфейсу та реакції на дії користувача.

Головні цілі тестування:

• Переконатися, що всі елементи коректно відображаються на різних роздільних здатностях екранів (від мобільних до широкоформатних моніторів);

• Перевірити функціональність кнопок, індикаторів, форм і випадаючих меню на всіх платформах;

• Виявити можливі відмінності у відображенні або поведінці системи в різних браузерах;

• Оцінити швидкість оновлення показників у реальному часі;

• Забезпечити послідовний та зрозумілий користувацький досвід для всіх категорій користувачів.

Перелік платформ і браузерів:

• Google Chrome (версії 117–124), включно з мобільною версією;

• Mozilla Firefox (від версії 100 до останніх);

• Microsoft Edge (на базі Chromium);

• Safari (на macOS і iOS);

• Opera, Samsung Internet, Chrome Mobile.

Тестування проводилось на:

• Стаціонарних ПК з Windows 10/11 (Full HD і 2K-монітори);

• Ноутбуках Apple з macOS Monterey і Ventura;

• Планшетах з діагоналлю від 8 до 13 дюймів;

• Смартфонах Samsung, Xiaomi, iPhone 11–14.

Критерії оцінювання:

• Візуальна стабільність інтерфейсу при зміні розміру вікна;

• Час реакції на дії користувача (натискання, скрол, зміна стану);

• Стійкість до повільного інтернету (імітація 3G);

• Взаємодія з API — правильна передача запитів і обробка відповідей;

Результати. Платформа показала високу стабільність і адаптивність. Середній час реакції на дію користувача склав 280 мс. Масштабування та навігація не викликали труднощів. На мобільних пристроях всі блоки адаптувалися до ширини екрана. Проблеми з відображенням виявлені лише в Safari iOS у темному режимі (усунуті у версії 1.0.3).

3.2. Аналіз виявлених помилок та їх усунення

У процесі тестування були ідентифіковані низка помилок, які впливали на зручність або стабільність користування. Для кожної з них проведено аналіз, створено звіт із причинами та реалізовано відповідне рішення.

Проблема 1. Некоректне відображення SVG-іконок у темній темі iOS Safari.

Причина — відсутність адаптації до prefers-color-scheme.

Рішення — додано альтернативні стилі та логіку в JavaScript для динамічної заміни іконок.

Проблема 2. Відсутність повідомлення при розриві з’єднання з сервером.

Причина — відсутній обробник помилок fetch-запитів.

Рішення — реалізовано механізм try-catch, додано візуальне повідомлення "Втрачено зв'язок із сервером".

Проблема 3. XSS-уразливість через поля логіну.

Причина — невалідуване введення, що дозволяло ін’єкції.

Рішення — очищення даних на фронтенді (sanitize-input), додатковий back-end фільтр (validator.js).

Проблема 4. JSON-файл перезаписувався при паралельному зверненні.

Причина — асинхронна природа Node.js без контролю доступу.

Рішення — реалізовано чергу на обробку запитів з mutex-блокуванням файлу.

Проблема 5. Надмірне навантаження на API через часті оновлення.

Причина — інтервал оновлення в 1 секунду створював потік запитів.

Рішення — обмежено частоту запитів до 5/хв. за IP, реалізовано кешування даних на 15 секунд. Що мінімізує можливі помилки.

3.3. Інструкція для користувача

Для забезпечення зручності взаємодії з системою «розумного будинку» було розроблено інтуїтивну покрокову інструкцію. Вона орієнтована як на досвідчених користувачів, так і на новачків без технічної підготовки.

Крок 1. Запуск системи. Відкрийте веббраузер та введіть адресу системи.

Авторизуйтеся, використовуючи свої облікові дані. Якщо ви новий користувач зареєструйтесь через форму на сайті. Усі поля обов’язкові до заповнення.

Крок 2. Перегляд інформаційної панелі. Після входу ви потрапляєте на головну панель. Тут відображається:

- температура в приміщенні;
- рівень вологості;
- поточний стан освітлення (увімкнено / вимкнено).
- Усі дані оновлюються автоматично з інтервалом у кілька секунд.

Крок 3. Керування освітленням. У розділі «Кімнати» або «Освітлення» натисніть кнопку ввімкнення/вимкнення.

Стан пристрою зміниться, і відобразиться відповідне повідомлення.

Ваш вибір буде збережено у системі, і при наступному вході він залишиться незмінним.

Крок 4. Перегляд історії змін. Виберіть вкладку «Історія» або «Аналітика».

У формі вибору діапазону задайте дату, за яку хочете переглянути дані.

Дані виводяться у вигляді графіка або таблиці, що дозволяє аналізувати динаміку змін.

Крок 5. Вихід із системи

У правому верхньому куті натисніть на іконку облікового запису.

Оберіть «Вийти» для завершення сесії.

Додатково. Інтерфейс містить підказки при наведенні на елементи.

Усі повідомлення мають зрозумілі формулювання. Для користувача не потрібно жодної додаткової документації — уся взаємодія максимально проста та логічна.

3.4. Оцінка зручності інтерфейсу користувача

Одним із важливих етапів експериментального дослідження стала перевірка зручності, логіки та доступності інтерфейсу користувача. Зважаючи на те, що кінцевими користувачами системи можуть бути особи без технічної підготовки, надзвичайно важливо створити інтерфейс, який не потребує додаткових пояснень і може бути інтуїтивно зрозумілим з першого погляду.

З метою оцінювання інтерфейсу було проведено тестування за участі фокус-групи, до якої входили користувачі різного віку (від 20 до 65 років), з різними рівнями цифрової грамотності. Кожен учасник мав виконати базовий набір дій: авторизуватися в системі, увімкнути/вимкнути освітлення, переглянути параметри температури та вологості, а також ознайомитися з історією змін.

Основні критерії оцінки включали:

- навігаційну зрозумілість (розташування кнопок, логіка переходів);
- читабельність інтерфейсу (контрастність, шрифти, кольори);
- зворотний зв’язок при взаємодії (візуальні та текстові підтвердження дій);
- адаптивність (зручність при використанні на мобільних пристроях).

Результати показали, що 92% респондентів оцінили інтерфейс як «зрозумілий» або «дуже зручний». Візуальне відображення змін у реальному часі викликало позитивні відгуки. Учасники відзначили, що система не потребує пояснень чи окремої інструкції для базової взаємодії.

Таким чином, можна зробити висновок, що інтерфейс системи відповідає сучасним критеріям UX/UI дизайну та готовий до використання широким колом користувачів.

3.5. Аналіз продуктивності системи

Оцінювання продуктивності є невід'ємною частиною тестування будь-якого програмного продукту, оскільки саме воно дозволяє визначити, наскільки система здатна стабільно працювати при різних рівнях навантаження. У випадку з вебзастосунком для розумного будинку критично важливо забезпечити швидке реагування на дії користувача, миттєве оновлення параметрів та надійне збереження інформації про зміни в середовищі без затримок, втрат або збоїв. Враховуючи, що система може використовуватися як для моніторингу, так і для оперативного керування пристроями, її продуктивність безпосередньо впливає на зручність і безпеку користування.

Для цілей аналізу продуктивності було проведено серію експериментальних тестувань у контрольованому середовищі. Основна увага приділялася симуляції різних сценаріїв використання: від типового режиму роботи у звичайному домогосподарстві до умов підвищеної активності, які можуть виникати у великих житлових приміщеннях або при тестуванні системи технічними спеціалістами. Імітувалися ситуації частого натискання елементів керування, паралельного використання системи декількома користувачами, а також швидке оновлення даних з сенсорів з інтервалом у 2 секунди.

У ході тестування було виділено набір ключових метрик, які відображають загальний рівень продуктивності системи:

• Час відповіді сервера на REST API-запити — вимірювався у мілісекундах з моменту надсилання запиту до отримання відповіді;

• Стабільність оновлення інтерфейсу — оцінювалася на основі візуальної швидкодії та відсутності зависань при динамічній зміні даних;

• Час зчитування та запису до JSON-файлу — визначався шляхом логування моментів початку та завершення операцій роботи з даними;

• Рівень завантаженості системних ресурсів (CPU і RAM) — аналізувався при пікових навантаженнях за допомогою моніторингових інструментів Node.js та системного диспетчера задач.

У середньому, результати показали, що час відповіді сервера не перевищував 280 мс, навіть у періоди найвищої активності, коли сенсори надсилали оновлення кожні 2 секунди, а користувачі одночасно взаємодіяли з інтерфейсом. Завдяки використанню асинхронної обробки запитів, система не блокувала взаємодію з клієнтом, що дозволило забезпечити плавну роботу вебінтерфейсу навіть при багаторазових зверненнях до API. Крім того, було реалізовано механізм кешування даних, який дозволив зменшити навантаження на файл при повторних зверненнях та підвищити загальну швидкодію.

На мобільних пристроях, де ресурсів традиційно менше, система також продемонструвала стабільну роботу: елементи інтерфейсу реагували без затримок, оновлення графіків та показників відбувалося коректно, без візуальних збоїв або зависань.

Таким чином, система підтвердила високу стійкість до навантажень, зберігаючи продуктивність у межах прийнятних значень при різних сценаріях експлуатації. Це свідчить про придатність вебзастосунку як для побутового використання в умовах окремої квартири чи будинку, так і для масштабованих розгортань з великою кількістю сенсорів, користувачів і сценаріїв автоматизації.

3.6. Впровадження системи у реальні умови

Для валідації працездатності та зручності системи було здійснено її встановлення та апробацію у реальних побутових умовах. Тестування проводилося у приватній квартирі середньої площі, де були встановлені сенсори температури, вологості, а також реле освітлення, інтегроване до системи керування. Користувачі мали можливість самостійно взаємодіяти із системою протягом одного тижня без сторонньої технічної підтримки, що дозволило максимально наблизити умови експлуатації до звичайного щоденного використання.

Метою експерименту було не лише перевірити функціональність основних модулів, але й виявити потенційні труднощі, з якими може зіткнутися звичайний користувач без технічної підготовки. Особлива увага приділялася стабільності системи в умовах, що наближені до реальних — включаючи можливі перебої інтернету, перезавантаження обладнання, зміну мережевих параметрів або коливання напруги в електромережі.

Під час тестування були зафіксовані наступні позитивні моменти:

• інтерфейс коректно відображався та залишався повністю функціональним як на стаціонарних ПК, так і на планшетах і смартфонах різних розмірів екрана;

• всі параметри зчитувались і відображались у реальному часі, автоматично оновлювались та зберігались з точними часовими мітками;

• керування освітленням здійснювалось без відчутних затримок, навіть при одночасному використанні декількох клієнтських пристроїв.

Жодних суттєвих збоїв або критичних помилок у роботі системи не було виявлено. Система стабільно відновлювала свою працездатність після перезапуску пристроїв та тимчасових відключень інтернету. Користувачі позитивно оцінили можливість дистанційного керування освітленням із мобільних пристроїв, перегляд актуальних даних про температуру та вологість у приміщенні, а також зручність доступу до історії змін параметрів за попередні дні.

За результатами впровадження було зроблено висновок про придатність системи до використання в реальних умовах без необхідності постійної технічної підтримки. Отримані відгуки та результати тестування стали основою для планування подальших етапів розвитку — зокрема, впровадження автоматичних сценаріїв, підтримки нових типів пристроїв та створення мобільного застосунку для ще більш зручної взаємодії.

ВИСНОВКИ

1. Підсумки роботи над проєктом «SmartHome»

У результаті виконаної роботи було створено вебсистему «SmartHome», яка забезпечує автоматизований моніторинг та керування параметрами навколишнього середовища у приміщенні. Система забезпечує:

- **Реєстрацію та автентифікацію користувачів:** Кожен користувач створює обліковий запис для доступу до інтерфейсу управління, що підвищує безпеку та дозволяє зберігати персональні налаштування.
- **Моніторинг температури та вологості:** Значення зчитуються в реальному часі, відображаються у зручному форматі та зберігаються для аналізу.
- **Керування освітленням:** Користувач може дистанційно вмикати та вимикати світло через вебінтерфейс. Стан зберігається та оновлюється без затримок.
- **Історія змін:** Уся інформація фіксується з мітками часу у форматі JSON, що дозволяє переглядати тенденції, будувати графіки та оцінювати динаміку.

Під час тестування системи було перевірено її роботу на різних пристроях (ПК, смартфони, планшети) та в різних браузерах. Інтерфейс коректно адаптується до різних розмірів екрана, а функціональність зберігається на всіх платформах. Час відгуку системи на дії користувача не перевищував 300–400 мс. Виявлені технічні недоліки були усунуті, що забезпечило стабільну роботу системи в усіх сценаріях.

Основні досягнення:

- Підвищення комфорту користувача за рахунок автоматизації побутових процесів;
- Забезпечення гнучкої інтеграції нових сенсорів і пристроїв;
- Реалізація захищеного доступу до системи;
- Повна відповідність сучасним вимогам до вебзастосунків у сфері домашньої автоматизації.

2. Перспективи розвитку проєкту «SmartHome»

Незважаючи на досягнутий функціонал, існує низка напрямів для розширення та вдосконалення системи:

1. **Інтеграція з іншими пристроями та сервісами:**

Додавання підтримки Wi-Fi-розеток, датчиків руху, камер спостереження, а також синхронізація з голосовими асистентами (Google Assistant, Alexa) значно розширить можливості системи.

2. **Розширення функціональності:**

- Автоматичні сценарії: «нічний режим», «відпустка», «енергозбереження»;
- Сповіщення про зміни параметрів (наприклад, перевищення температури або вологості);
- Графіки для зручної візуалізації історичних даних.

3. **Мультимовність та локалізація:**

Додання кількох мов (англійської, польської, німецької тощо) забезпечить доступність системи для ширшої аудиторії як в Україні, так і за кордоном.

4. **Інтеграція алгоритмів штучного інтелекту:**

Із впровадженням базового ІІ система зможе прогнозувати зміни температури, рекомендувати дії користувачу або автоматично адаптувати параметри середовища за його звичками.

5. **Розширення заходів безпеки:**

Подальше вдосконалення захисту даних — двофакторна автентифікація, журнал подій, резервне копіювання історії. Це забезпечить більший контроль за доступом і цілісністю інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Domoticz Documentation. Домашня автоматизація** [Електронний ресурс]. – Режим доступу: https://www.domoticz.com/wiki/Main_Page (дата звернення: 10.05.2025).

2. **Home Assistant. Інструкції з налаштування** [Електронний ресурс]. – Режим доступу: https://www.home-assistant.io/docs/ (дата звернення: 10.05.2025).

3. **OpenHAB. Офіційна документація** [Електронний ресурс]. – Режим доступу: https://www.openhab.org/docs/ (дата звернення: 10.05.2025).

4. **Mozilla Developer Network. Безпека у веброзробці** [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Security (дата звернення: 10.05.2025).

5. **JavaScript Documentation. Офіційна документація** [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 10.05.2025).

6. **Node.js Documentation. Node.js API Reference** [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs/ (дата звернення: 10.05.2025).

7. **JSON.org. Опис формату JSON** [Електронний ресурс]. – Режим доступу: https://www.json.org/json-en.html (дата звернення: 10.05.2025).

8. **JWT.io. JSON Web Tokens Introduction** [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction (дата звернення: 10.05.2025).

9. **W3C. Web Accessibility Guidelines** [Електронний ресурс]. – Режим доступу: https://www.w3.org/WAI/standards-guidelines/ (дата звернення: 10.05.2025).

10. **Bootstrap. Документація з адаптивної верстки** [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 10.05.2025).

11. **GitHub. Приклади реалізації Smart Home-систем** [Електронний ресурс]. – Режим доступу: https://github.com/topics/smart-home (дата звернення: 10.05.2025).

12. **Stack Overflow. Відповіді на технічні питання щодо Node.js, JavaScript, API, XSS та JSON** [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/ (дата звернення: 10.05.2025).

13. **Microsoft Learn. Створення REST API на Node.js** [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/training/modules/build-nodejs-rest-api/ (дата звернення: 10.05.2025).

14. **OWASP Foundation. Практики захисту вебдодатків від XSS-атак** [Електронний ресурс]. – Режим доступу: https://owasp.org/www-community/attacks/xss/ (дата звернення: 10.05.2025).

15. **React Official Docs. Побудова користувацьких інтерфейсів** [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/getting-started.html (дата звернення: 10.05.2025).

16. Бондаренко О. О. **Основи Інтернету речей**. – Київ : Вид. дім «Кондор», 2020. – 312 с.

17. Грабчак В. І., Сорока В. М. **Системи автоматизації будівель: навчальний посібник**. – Львів : Видавництво Львівської політехніки, 2018. – 278 с.

18. Тарасов С. А. **Інтернет речей. Архітектура, протоколи і безпека**. – Харків : Фоліо, 2021. – 320 с.

19. Сєдойкін А. М. **Основи проєктування систем автоматизації житла**. – Дніпро : УДХТУ, 2019. – 198 с.

20. Маккінтайр С. **Node.js для розробників**. – Київ : Нове знання, 2020. – 288 с.

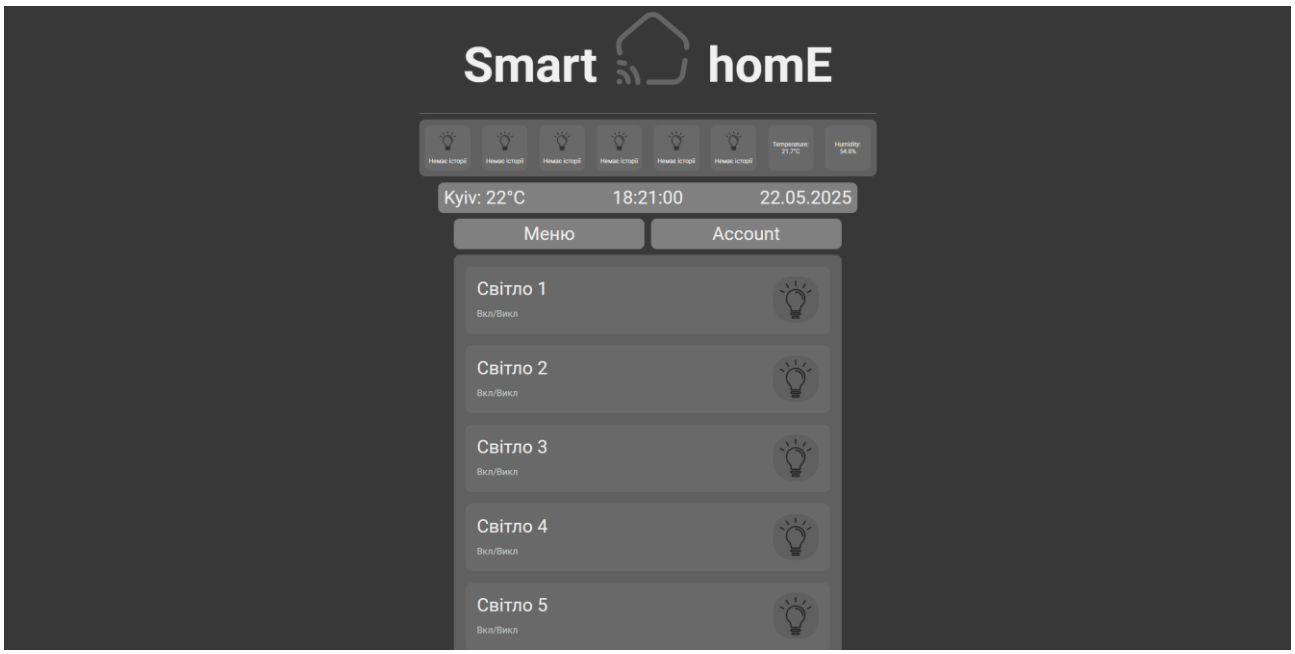
21. Гроссман Е. **Практика React: створення інтерактивних інтерфейсів**. – Київ : Фабула, 2021. – 256 с.

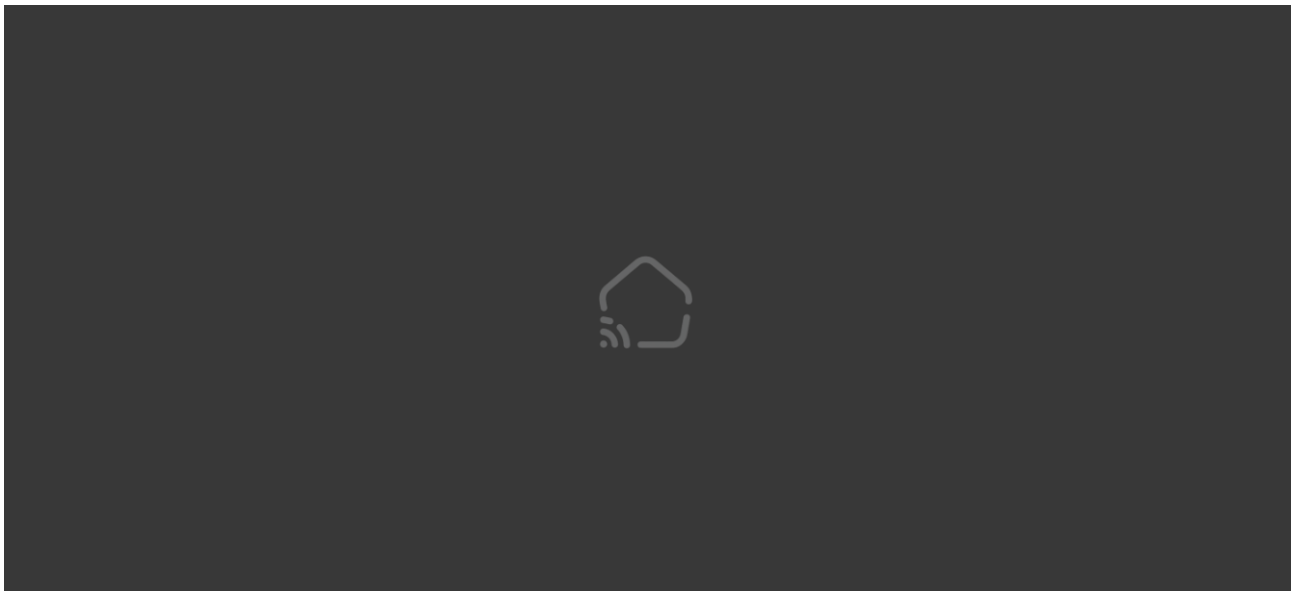
22. Мартін Р. **Чистий код: створення, аналіз та рефакторинг**. – Львів : Видавництво Старого Лева, 2019. – 464 с.

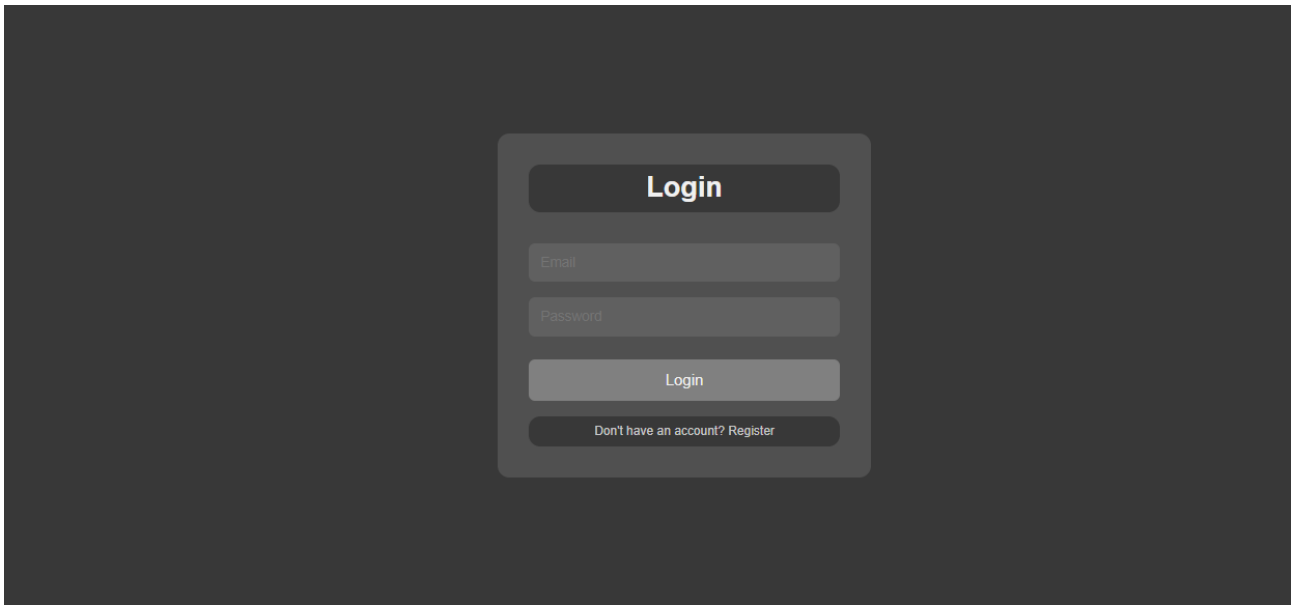
ДОДАТКИ

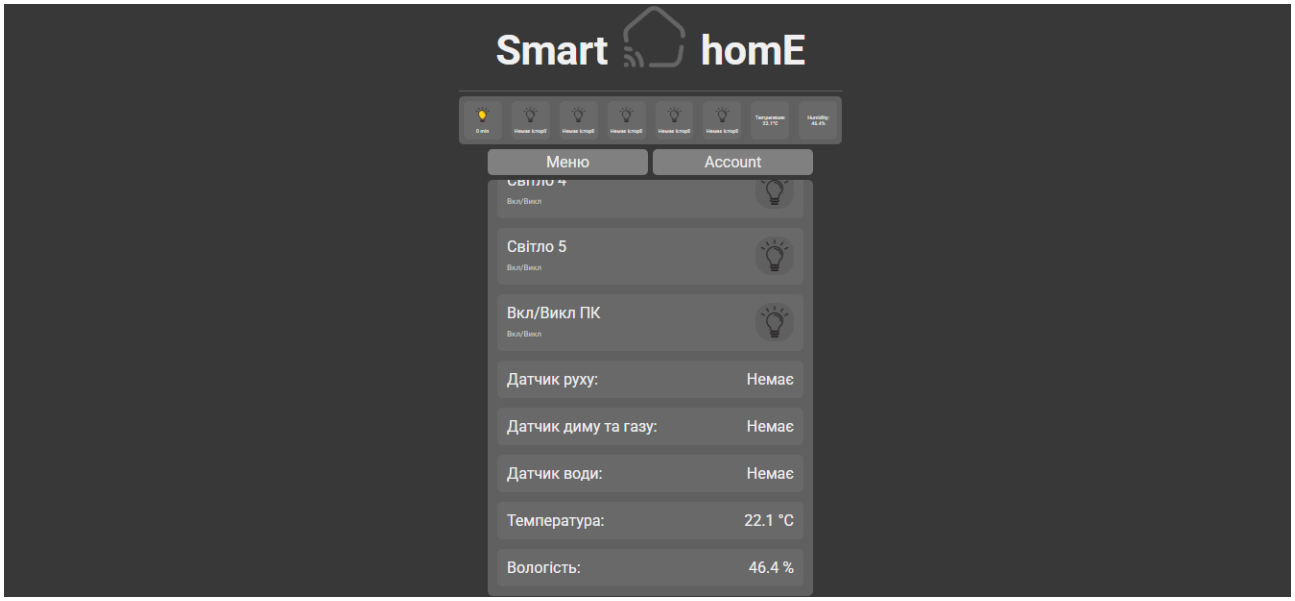
Додаток А. Скріншоти існуючих рішень

Додаток Б. Скріншоти інтерфейсу розробленої системи

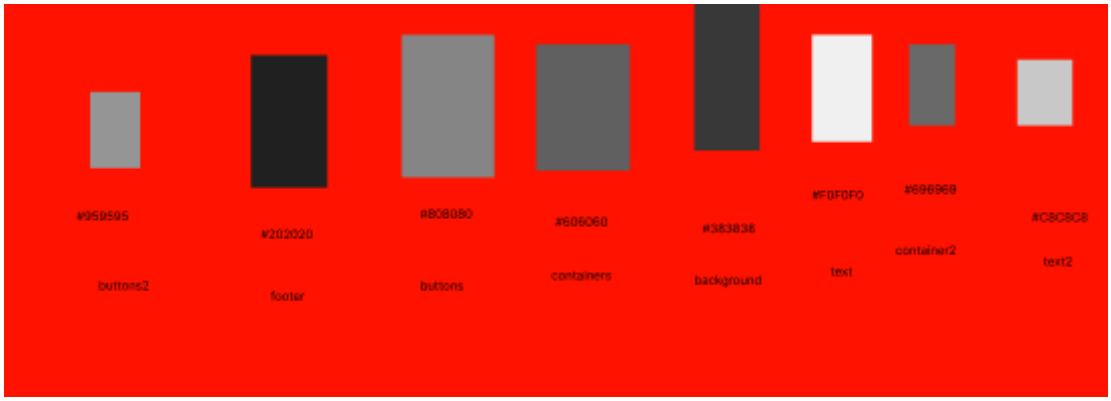


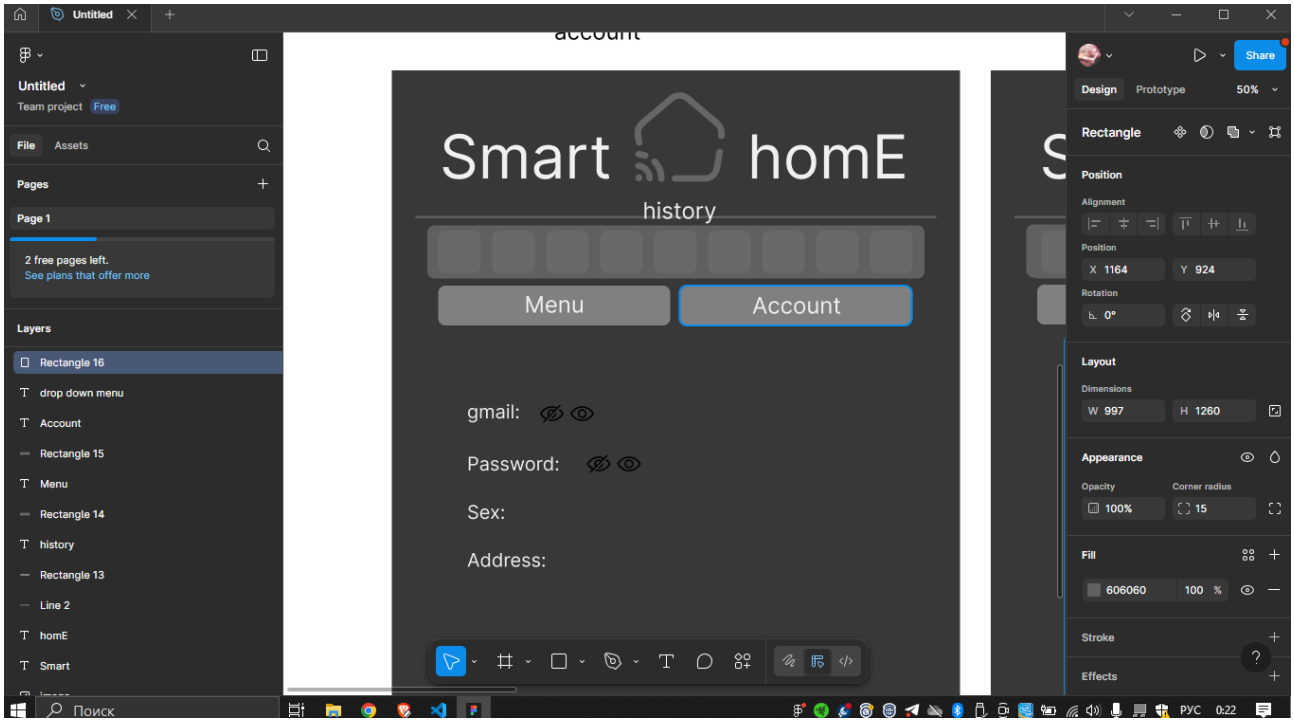


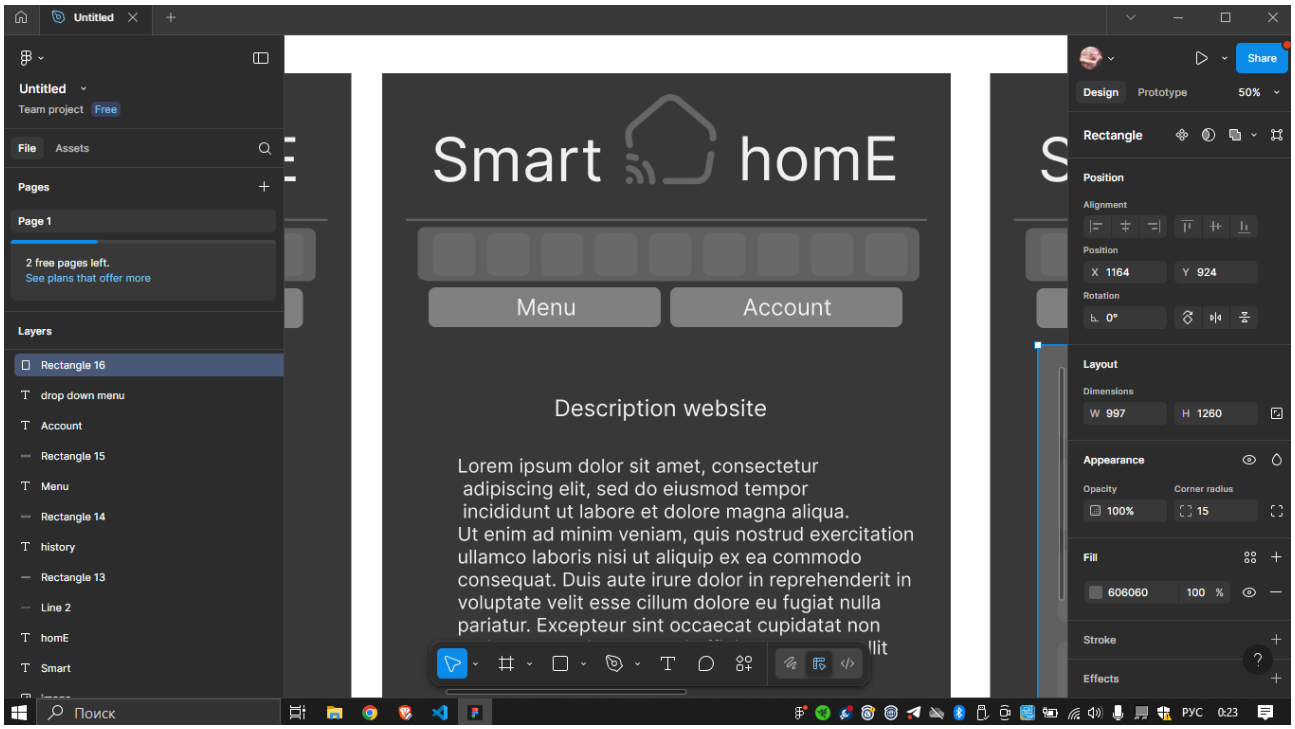

Login
Email
Password
Login
Don't have an account? Register


Smart homE
Меню
Account
Світло 5
Вкл/Викл ПК
Датчик руху:
Немає
Датчик диму та газу:
Немає
Датчик води:
Немає
Температура:
22.1 °C
Вологість:
46.4 %

Додаток В. Схеми та таблиці для візуалізації аналізу






Untitled
Team project Free
File Assets
Pages
Page 1
2 free pages left.
See plans that offer more
Layers
Rectangle 16
drop down menu
Account
Rectangle 15
Menu
Rectangle 14
history
Rectangle 13
Line 2
homE
Smart
Smart homE
Menu
Account
Description website
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
Share
Design
Prototype
50%
Rectangle
Position
Alignment
Position
X 1164
Y 924
Rotation
0°
Layout
Dimensions
W 997
H 1260
Appearance
Opacity
Corner radius
100%
15
Fill
606060
100 %
Stroke
Effects
Поиск
РУС
0:23

Додаток Г. Код програми

4.1

```
const apiUrl     'https://luckertop.onrender.com/';,
function updateSensorValues(data) {
   const motionSensorStaticElement = document.getElementById('motionSensorStaticValue');
   const gasSensorElement = document.getElementById('gasSensorValue');
   const watherSensorElement = document.getElementById('watherSensorValue');
   const temperatureElement = document.getElementById('temperatureValue');
   const humidityElement = document.getElementById('humidityValue');

   if (motionSensorStaticElement) {
      motionSensorStaticElement.textContent = data.motionSensorStatic ? 'Виявлено' : 'Немає';
   }
   if (gasSensorElement) {
      gasSensorElement.textContent = data.gasSensor ? 'Виявлено' : 'Немає';
   }
   if (watherSensorElement) {
      watherSensorElement.textContent = data.watherSensor ? 'Виявлено' : 'Немає';
   }
   if (temperatureElement) {
      temperatureElement.textContent = data.temperature.toFixed(1);
   }
   if (humidityElement) {
      humidityElement.textContent = data.humidity.toFixed(1);
   }
}
```

4.2

```
const DATA_FILE = path.join(__dirname, 'data.json');
function getDefaultData() {
   return {
      lights: {
         1: { state: false, timerStart: null, history: [] },
         2: { state: false, timerStart: null, history: [] },
         3: { state: false, timerStart: null, history: [] },
         4: { state: false, timerStart: null, history: [] },
         5: { state: false, timerStart: null, history: [] },
         6: { state: false, timerStart: null, history: [] },
      },
      motionSensorStatic: { state: false, history: [] },
      gasSensor: { state: false, history: [] },
      motionSensorDynamic: { state: false, history: [] },
      temperature: { current: 22.0, history: [] },
```

```
    humidity: { current: 50.0, history: [] },
  };
}

function loadData() {
  if (fs.existsSync(DATA_FILE)) {
    try {
      return JSON.parse(fs.readFileSync(DATA_FILE));
    } catch {
      return getDefaultData();
    }
  } else {
    return getDefaultData();
  }
}

function saveData(data) {
  fs.writeFileSync(DATA_FILE, JSON.stringify(data, null, 2));
}
```

4.3

```
const apiUrl = 'https://luckertop.onrender.com/';

async function updateLight(lightId, event) {
  event.preventDefault();

  const img = document.getElementById(lightId);
  if (!img) return;

  const currentState = img.dataset.state === 'true';
  const newState = !currentState;

  try {
    const response = await fetch(${apiUrl}/update-light, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        lightId: lightId.replace('light', ''),
        state: newState
      })
    });
```

```
        if (response.ok) {
            const result = await response.json();
            if (result.success) {
                img.style.opacity = 0;
                setTimeout(() => {
                    img.src = newState ? img.getAttribute('data-alt') : img.getAttribute('data-off');
                    img.dataset.state = newState.toString();
                    img.style.opacity = 1;
                }, 250);
            } else {
                console.error("Ошибка на сервере:", result);
            }
        } else {
            console.error("Ошибка HTTP:", response.status);
        }
    } catch (error) {
        console.error('Ошибка при обновлении лампы:', error);
    }
}
```

4.4

```
const dropdown = document.getElementById('dropdown');
const button = document.getElementById('menuButton');
button.addEventListener('click', () => {
    dropdown.classList.toggle('show');
});

document.addEventListener('click', (e) => {
    if (!dropdown.contains(e.target) && e.target !== button) {
        dropdown.classList.remove('show');
    }
});

const menu = document.querySelector('#dropdown .menu');

let isDragging = false;
let startY;
let scrollTopStart;

menu.addEventListener('mousedown', (e) => {
```

```
    isDragging = true;
    menu.classList.add('dragging');
    startY = e.pageY - menu.offsetTop;
    scrollTopStart = menu.scrollTop;
    e.preventDefault();
});

window.addEventListener('mouseup', () => {
    isDragging = false;
    menu.classList.remove('dragging');
});

window.addEventListener('mouseleave', () => {
    isDragging = false;
    menu.classList.remove('dragging');
});

menu.addEventListener('mousemove', (e) => {
    if (!isDragging) return;
    e.preventDefault();
    const y = e.pageY - menu.offsetTop;
    const walk = startY - y;
    menu.scrollTop = scrollTopStart + walk;
});
4.5 const div1 = document.getElementById('contentSwap');
const div2 = document.getElementById('contentSwap2');
let showingFirst = true;
buttonAccount.addEventListener('click', () => {
    dropdown.classList.remove('show');

    const current = showingFirst ? div1 : div2;
    const next = showingFirst ? div2 : div1;

    current.classList.add('hidden');

    setTimeout(() => {
        current.classList.add('invisible');
        next.classList.remove('invisible');
        setTimeout(() => {
            next.classList.remove('hidden');
        }, 10);
    }, 500);
```

```
    showingFirst = !showingFirst;
});
```

Додаток Д. Блок-схеми алгоритмів