МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

СІТЬКО ОЛЕКСАНДР ТАРАСОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д.т.н, доцент
_____Т. В. Нескородєва
«____» _____ 20__ р.

Розробка панелі управління розумним будинком

Спеціальність 122 «Комп'ютерні науки»
Кваліфікаційна (бакалаврська) робота

Керівник:
Мартянова Т.А., старший викладач
кафедри інформаційних технологій,
старший викладач

Оцінка:
___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК:
_____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Людство постійно прогресує та знаходить методи спрощення собі життя, одним із таких методів стає автоматизація та можливість керувати будь-чим дистанційно. На даному етапі розробки, щоб користувач міг увійти у систему, йому потрібни лише мати пристрій, що надає змоги увійти до мережі інтернет. Користувачем даного веб-додатку може бути як власник будинку, так і наприклад адміністратор хостелу. Метою курсової роботи є розробка зручного веб-додатку із графічним інтерфейсом для можливості керування певних електроприборів, перегляду показників різних кімнат тощо. Уся інформація зберігається у JSON-файлах. Веб-додаток, що був спроектнований, розброблений та протестований у рамках практичної частини даної курсової роботи, являє собою інтерфейс панелі керування розумним будинком.

Ключові слова: веб-додаток, інформаційна система, HTML5, CSS3, JavaScript, Vue.js, GUI, JSON.

ABSTRACT

Mankind is constantly progressing and finding methods to simplify their lives, one of such methods is automation and the ability to control anything remotely. At this stage of development, in order for a user to log in, he only needs to have a device that allows him to log on to the Internet. The user of this web application can be both a homeowner and, for example, a hostel administrator. The purpose of the course work is to develop a user-friendly web application with a graphical interface for the ability to control certain appliances, view the performance of different rooms and more. All information is stored in JSON files. The web application, which was designed, developed and tested as part of the practical part of this course work, is the interface of the smart home control panel.

Keywords: web application, information system, HTML5, CSS3, JavaScript, Vue.js, GUI, JSON.

ЗМІСТ

ВСТУП

В бакалаврській роботі розглядається проектування інформаційних систем.Інформаційна система – це сукупність технологій та методів для збереження та обробки даних із подальшою презентацією користувачу. Із розвитком технологій людство по різному спрощує собі життя, це стосується всіх областей, починаючи від медицини закінчуючи автоматизованим машино-будівництвом. Не стає винятком і побутове життя. Все більше і більше стають популярними так звані «Розумні будинки». Ця технологія вважається проривом в плані комфортного життя. Власник такого задоволення може, тримаючи в руках лише телефон, перевірити температуру повітря у кімнатах, ввімкнути посудомийну машину та відкрити/закрити вікна, кількома натисканнями на дисплей.

Уся система «Розумного будинку» складається із:

- Прикладного обладнання (датчики, плати та інші прибори вмонтовані по всьому будинку);
- Сервера (достатньо потужний комп’ютер для обробки необхідного потоку даних);
- Пульту керування (програмне забезпечення наприклад на телефоні чи планшеті).

У цій роботі буде з емульована робота пульту керування «Розумним будинком».

Метою бакалаврської роботи є проектування та розробка зручного та інтуїтивно зрозумілого веб-додатку за допомогою фреймворку Vue.js.

Vue.js[1] – JavaScript-фреймворк, що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв’язування даних.

MVVM[2](Model-View-ViewModel) – шаблон проектування, що застосовується для проектування додатків, в тому числі і веб-додатків. Застосовується для відокремлення моделі(логіки) та її відображення(графіка).

Це дає можливість одночасно вести розробку Back-end та Front-end без шкоди один одному.

Веб-додаток буде представляти собою пульт керування «Розумним будинком». Дизайн був взятий із готових концепт-артів та доповнений мною із міркувань «дружелюбного» UI/UX дизайну та моїх власних побажань таких як показ прогнозу погоди та новин у режимі реального часу за допомогою API запитів.

UI-дизайн – це частина дизайну, що відповідає за гармонічні поєднання кольорів, якісну анімацію, шрифти та все інше, що являється графікою.

UX-дизайн – це частина дизайну, яка відповідає за зручне користування додатком, продумані зв'язки між елементами інтерфейсу тощо. Загалом UX-дизайнер складає карту додатку.

У веб-додатку передбачені такі функції як:

- Можливість переходу на різні вкладки;
- Можливість редагувати свій профіль;
- Можливість змінювати показники в кімнатах.

У користувача буде тісна взаємодія із базами даних через графічну частину веб-додатку. Бази дані будуть представлені у вигляді JSON файлів, користувач матиме можливість додавати, видаляти та змінювати дані цих файлів.

Веб-додаток буде реалізований за допомогою HTML5, CSS3, JavaScript та фреймворк Vue.js 2 версії. Вся візуальна частина реалізована за допомогою HTML5 [3] та CSS3, всі взаємодії між візуальними елементами реалізовані за допомогою JavaScript, а вся робота із базами даних та передачами даних в самому додатку реалізована за допомогою Vue.js.

Взагалі зараз існує три версії Vue.js [1], проте для розробки додатку обрана саме друга версія, тому що третя вийшла відносно не давно та має багато корисних методів, але вона не стабільна.

Теоретичне та практичне значення одержаних результатів полягає в подальшому практичному використані при проектуванні власних інтелектуальних рішень в області керування.

Окрім вище зазначених статей, на основі аналізу предметної області, отримані матеріали були оформленні у вигляді тез та опубліковані.

Структура кваліфікаційної (бакалаврської) роботи передбачає такі структурні підрозділи: аналіз предметної області, проектування ПЗ і його реалізація.

Обсяг і структура роботи: ст. 55, рис. 15, посилань 19.

РОЗДІЛ 1. Огляд сучасних підходів та опис предметної області

Метою даної бакалаврської роботи є створення веб-додатку панелі керування розумним будинком. Із вимог до цього веб-додатку можна виділити зручність у користуванні, тобто «доброзичливість» інтерфейсу, це включає в себе інтуїтивно зрозумілий інтерфейс, мінімалістичне оформлення або виділення ключових елементів, а також легка для сприйняття кольорова палітра. Користувач матиме можливість переглядати показники різних кімнати (температуру повітря, вологість повітря, які прибори ввімкненні, а які вимкненні), переглядати останні новини із порталу BBC News, записувати план справ у ToDoList тощо [8].

1.1 Опис функціональних можливостей системи, що розробляється

Функціональна система передбачає собою веб-додаток типу «Панель керування “Розумним будинком”», який буде складатись із двох великих частин:

1. Візуальна частина, з якою буде взаємодіяти користувач безпосередньо, тобто User Interface Design (UI) та User Experience Design (UX).
2. Функціональна частина, з якою користувач буде взаємодіяти за допомогою UI/UX.

Функціонал:

- Можилвість переходу на різні вкладки:

У плануванні передбачаються такі вкладки, як Login (сторінка авторизації), Home (перелік тих сусідів, що авторизувались вдома; перелік кімнат в будинку; ToDo List – перелік справ/нотаток/нагадувань; час/дата; температура надворі та останні новини, що будуть братись за допомогою API-запитів у реальному часі), Room (перелік показників кімнати, таких як температура, рівень освітлення та перелік електронних приборів, що

ввімкнені), Members (перелік авторизованих користувачів та коротка інформація про них), Profile [9].

- Можливість редагувати свій профіль:

Можна вказати інформацію, що буде відображатись у вкладці Members. Наприклад: коротко про себе; номер телефону; емейл; дата народження; посилання на соціальні мережі.

- Можливість змінювати показники в кімнатах:

Змінювати рівень освітлення, змінювати температуру повітря, закривати/відкривати вікна, вмикати/вимикати техніку.

- У адміністратора передбачена можливість додавати нових користувачів:

Користувач із правами алміністратора матиме можливість додавати нових користувачів, даючи їм логіни та паролі.

На даннний момент у веб-додатку є декілька шарів таких як:

- Візуальна частина, яку бачить користувач.
- Функціональна частина, яка повністю відповідає за логіку додатку, бази даних тощо.

1.2 Опис функціональності кожної з частин

Візуальна частина – Візуальна частина додатку відображає найбільшу частину контенту та реалізує можливість взаємодії користувача з програмним продуктом [10].

Функціональна частина – Функціональна частина відповідає за всю логіку веб-додатку, тобто за взаємодію з базами даних, їх обмеження.

Опис функціонально-візуальної частини рисунок 1.1.

Один з головних принципів веб-додатку є взаємодія візуальной частини та функціональної.

1. Візуальна частина, яка відображає усі її компоненти. Візуальна частина надає можливість користувачу взаємодіяти з програмним додатком.

2. Функціональна частина відповідає за поведінку усіх об'єктів, їх обмеження та впливом на бази даних. У цій частині прописана логіка для кожного обьекту, яка потім використовується для маніпуляції з цими об'єктами.

Рисунок 1.1 – Функціонально-візуальної архітектура ІС.

Визначимо інформаційні потоки даних з урахуванням необхідних функцій:

- Візуальна частина, яка містить потокову інформацію.
- Функціональна частина, яка містить логіку.

У веб додатку будуть використовуватись REST API запити із відкритих баз даних за протоколом HTTP (рис. 1.2) [7].

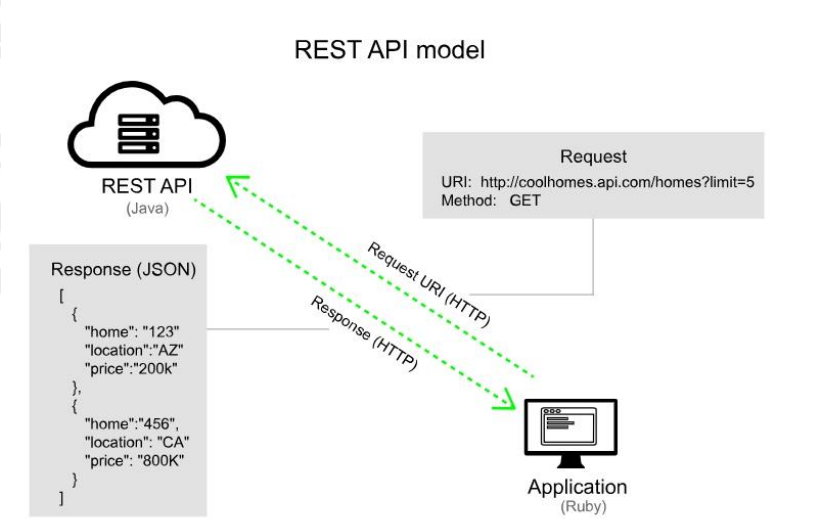


Рисунок 1.2 – Схема REST API запиту

1.3 Опис характеристик необхідних додатків та середовищ

Веб-додаток буде реалізований за допомогою HTML5, CSS3, JavaScript та фреймворк Vue.js 2 версії. Вся візуальна частина реалізована за допомогою HTML5 та CSS3, всі взаємодії між візуальними елементами реалізовані за допомогою JavaScript, а вся робота із базами даних та передачами даних в самому додатку реалізована за допомогою Vue.js [1].

Взагалі зараз існує три версії Vue.js, проте для розробки додатку обрана саме друга версія, тому що третя вийшла відносно не давно та має багато корисних методів, але вона не стабільна.

Середовищем для розробки було обрано Visual Studio Code останньої версії через зручність, компактність та велику кількість налаштувань персоналізації, а також через можливість підключення до GitHub.

Також у даному веб-додатку використовуються API-запити, це обумовленно тим, що планується показувати прогноз погоди та актуальні новини у реальному часі.

API-запити надсилаються до відкритих джерел у мережі Інтернет, отримують відповідь у вигляді масиву за правилами синтаксису JSON файлу. Після отримання масиву, він записується у JSON файл та передається до візуальної частини зрозумілому людині вигляді [7].

База даних у веб-додатку буде реалізована у вигляді різних JSON-файлів у яких буде міститись інформація про різні показники (логін/пароль, температура у кімнаті тощо).

Користувач матиме можливість вносити до них зміни, через взаємодію із графічним інтерфейсом користувача.

РОЗДІЛ 2. Логічна модель веб-додатку

2.1 Обґрунтування вибору технології, мови програмування та інструментальних засобів для розробки шарів системи

Візуальна частина. Для візуальної частини були використані технології та методи UI/UX дизайну та Front-end розробки.

Для створення та відображення UI/UX дизайну була використана платформа Figma. Figma — векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач. Є безкоштовним для індивідуальних користувачів і платним для фахових команд. Це зручна та безкоштовна платформа для створення та поширення дизайну макетів, що згодом можуть бути втіленні у життя за допомогою коду.

У частині Front-end розробки були використані такі технології як HTML5, CSS3 та JavaScript.

HTML5[3] – це мова розмітки гіпертексту веб-сторінки. Із 2008 року HTML версії 5 стала підтримуватись всіма браузерами та є актуальної і до сьогодні. У Front-end розробці HTML – це так званий кістяк веб-сторінки, за допомогою нього записуються блоки, текст, посилання та все інше.

CSS3[4] – це каскадні таблиці стилів веб-сторінки. На даний момент CSS версії 3 є найактуальнішою, проте наразі ведеться розробка CSS версії 4, хоча це лише чорновики, виглядає багатообіцяюче. У Front-end розробці CSS – це всі стилі та більшість випадків із розміщенням блоків на веб-сторінці.

JavaScript[5]–це динамічна, імперативна та об'єктно-орієнтована мова програмування. На сьогоднішній день ця мова досить популярна, в основному використовується для реалізації сценаріїв веб-сторінок. У цій курсовій роботі також використовується бібліотека jQuery. Синтаксис jQuery робить зручним

вибір елементів структури DOM, створення унікальної анімації, обробки подій, розробки AJAX-додатків тощо.

Функціональна частина. До логічної частини відноситься фреймворк мови JavaScript – Vue.js.Перший реліз Vue.js стався у 2014 році, а у 2016 році реліз версії 2, що і буде використовуватись у цій роботі. Vue використовують такі фірми-гіганти як Alibaba, Baidu, Xiaomi, GitLab тощо.

Синтаксис Vue дозволяє використовувати шаблони на основі HTML, це у свою чергу дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue.

Одним із переваг Vue є модульність. Vue.js надає можливість прописувати окремі файли-модулі, в яких описаний шаблон HTML, стилі CSS та скрипти JavaScript. Це дозволяє швидко знайти потрібний елемент та редагувати його без небезпеки зламати інший модуль.

Для даної бакалаврської роботи був обраний саме Vue.js, тому що на даний момент цей продукт є доволі перспективним та амбіційним, про це свідчить хоча б те, що Vue підтримується лише спільнотою, проте достойно стоїть напроти продуктів Google(Angular2) та Facebook(React 15).

Дана система поділяється на три модулі [14]:

- Модуль взаємодії з користувачем, тобто засоби введення/виведення інформації;
- Модуль формування даних, включає в себе роботу з файловою системою, а саме за методи та функції зчитування та запису даних JSON-файлів, а також включає в себе API-запити;
- Модуль перевірки даних, відповідає за перевірку введених даних, опираючись на завчасно задані умови та обмеження.

Ці модулі взаємодіють між собою та передають один одному дані впродовж усієї роботи веб-додатку. Наприклад: користувач вводить дані (модуль взаємодії з користувачем), дані проходять перевірку на коректність (модуль перевірки даних), при коректних даних відбувається запис у JSON-файл (модуль формування даних) та відображається користувачу модальне

вікно із повідомленням про успіх операції (модуль взаємодії з користувачем), при не коректних даних відображається користувачу модальне вікно із повідомленням про помилку (модуль взаємодії з користувачем).

Для розробки даного веб-додатку використано такі інструментальні засоби:

1. VS Code (Visual Studio Code) – редактор початкового коду та зневаджувач. Популярний редактор коду від компанії Microsoft. Підтримується ОС Windows, macOS і Linux. Редактор містить вбудований зневаджувач(дебагер), інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціюється як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

2. Figma — векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач. Є безкоштовним для індивідуальних користувачів і платним для фахових команд.

3. Framework Vue.js 2 — JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних. Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість

компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться. В Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто замініть шаблон на рендерингову функцію. Рендерингова функція відкриває можливості для потужних патернів базованих на компонентах - для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині.

4. Система контролю версій Git — розподілена система керування версіями файлів та спільної роботи. Проєкт створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

2.2 Опис окремих складових веб-додатку

У веб-додатку передбачені такі сторінки:

- Login;
- Home;
- ToDo List;
- Rooms;
- Members;
- Profile.

Кожна із цих сторінок має своє призначення та певний функціонал. На всіх сторінках, окрім сторінки Login, зліва присутня панель – меню із можливістю переходу на різні сторінки.

Усі дані, що вводяться та/або змінюються на усіх сторінках проходять перевірку на валідність та записуються у JSON-файли. При наступних змінах дані беруться також із них.

Use Case Diagram показана на рис. 2.1.

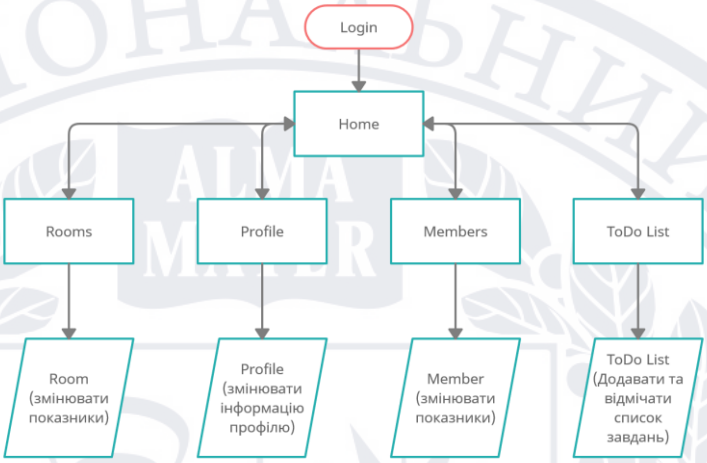


Рисунок 2.1 – Схема UX

При вході у веб-додаток спочатку потрібно пройти авторизацію, тобто ввести так званий логін та пароль від вашого облікового запису, що зареєстрований у системі. При введені логіну та паролю відбувається перевірка даних, чи існують взагалі такі дані у базі даних та чи відповідає пароль логіну. Додавати нових користувачів матиме можливість лише користувач із правами адміністратора (рис.2.2).

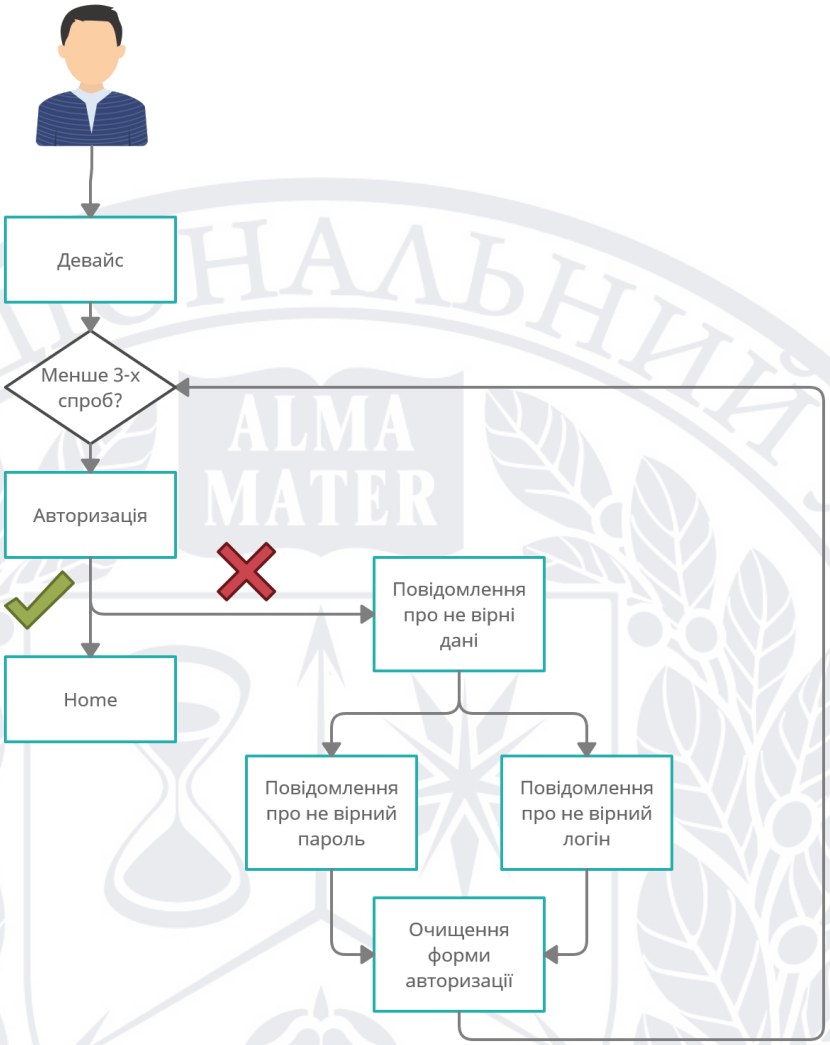


Рисунок 2.2 – Логічна схема авторизації у додатку

Після проходження авторизації, користувач потрапляє на головну сторінку (Home). Ця сторінка існує лише для виводу інформації та переходу на інші сторінки, внесення будь-яких змін на головній сторінці не передбачено, окрім відмічання виконаних завдань із ToDo List.

На головній сторінці виводиться така інформація [15]:

- Дата та час;
- Користувачі, які авторизувались вдома (можна перейти на їхні профілі);
- Перелік кімнат (можна перейти на конкретну кімнату);
- ToDo List (на сторінці Home, можна лише відмітити виконане завдання);

- Прогноз погоди у реальному часі;
- Останні новини у реальному часі.

ToDo List. Після переходу на цю сторінку відображається блок із вже написаними завданнями. Ці завдання можна помічати як вже зроблені або видаляти їх, також присутній функціонал додавання нових завдань [16].

Rooms. Ця сторінка являє собою слайдер із всіх кімнат. На кожному слайді є фотографія кімнати, її назва, записані показники кімнати (температура, вологість повітря, освітленість) та перелік техніки із позначкою ввімкнено чи вимкнено. При натисканні кімнату можна змінювати її показники та вмикати/вимикати техніку.

Members. На цій сторінці показані всі користувачі у вигляді карток на яких є ім'я, логін та статус (вдома чи ні). При натисканні на картку здійснюється перехід на обраного користувача. Там є така інформація як ім'я, логін, статус, номер телефону, емейл, дата народження та коротка інформація про користувача (що він пише).

Profile. На сторінці профілю є такі дані як ім'я, логін, посада, коротка інформація, номер телефону, емейл, дата народження та статус. Усе вище перечислене власник профілю може змінювати.

РОЗДІЛ 3. Опис реалізації веб-додатку

3.1 Опис засобів реалізації веб-додатку

Для реалізації веб-додатку для даної роботи було обране середовище розробки Visual Studio Code. Такий вибір був зроблений через те, що це середовище розробки спрямоване саме у напрямку веб-розробки, містить доволі багато офіційних плагінів, що спрощують та прискорюють розробку, також має вбудовані так звангі «емет» функції, це функції, що завершують написання задекларованих слів (якщо ввести в html-документі букву a, то «емет» видасть таку конструкцію: ; ця конструкція являє собою тег посилання), також перевагою VS Code є вбудований модуль GIT, що дозволяє додавати коміти прямо під час розробки не виходячи із редактора. Також VS Code підтримує всі формати файлів, що було використано (.html, .css, .scss, .js, .vue, .json), також має офіційні плагіни для полегшення та прискорення роботи із ними.

Так званий «кістяк» веб-додатку був написаний задопомогою мови гіпертекстової розмітки HTML версії 5.

Всі стилі були написані за допомогою каскадних таблиць стилівCSS3, а саме було використано препроцесор SCSS. SCSS – це така мова, код якої переводиться у формат звичайного CSS за допомогою програми на мові RUBY. Після збереження файлу у форматі SCSS автоматично створюєть ще два файли: .css та .min.css; різниця між ними полягає у тому, що .min.css це мініфікована версія того самого .css, тобто весь код записаний в оди рядок, без коментарів та зайвих пробілів, це зменшує об'єм файлу та як наслідок підвищує швидкодію сайту або веб-додатку [17].

Більшість функцій були написані на мові програмування JavaScript. JavaScript – це динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта взаємодіяти з користувачем, керувати браузером, асинхронно

обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки. Під час користування JavaScript використовувалась його бібліотека jQuery.

jQuery – це велика бібліотека, що спрощує та значно прискорює написання коду, полегшує звернення до елементів структури DOM, дозволяє зручно створювати анімацію, обробляти події тощо. Серед прикладів використання у даній курсовій роботі можна зазначити годинник, що працює у реальному часі, функціонування модальних вікон тощо.

Відповідно до об'єктної моделі документа («Document Object Model», коротко DOM), кожен HTML-тег є об'єктом. Вкладені теги є дітьми батьківського елемента. Текст, що знаходиться всередині тега, також є об'єктом.Всі ці об'єкти доступні за допомогою JavaScript, ми можемо використовувати їх, щоб змінити сторінку.Наприклад:

document.body

Нижче описаний код являє собою об'єкт для тега <body>.

```
<!DOCTYPEHTML>
<html>
<head>
<title>Олосях</title>
</head>
<body>
Правда о лосях.
</body>
</html>
```

Ця частина коду дасть змогу легко розібратись у понятті структури DOM, через рисунок 3.1.

Згодом статичний додаток був перенесений на фреймворк Vue.js 2. Vue.js – це фреймворк мови програмування JavaScript призначений для розробки саме веб-додатків. Для таких цілей існують ще такі фреймворки як Angular та React, проте вибір впав саме на Vue.js через його простоту та

лаконічність. У самому Vue було використано пакувальник Webpack, це програма з відкритим кодом написана на JavaScript, яка відповідає за генерацію статичних файлів із окремих модулів, що мають певні залежності.

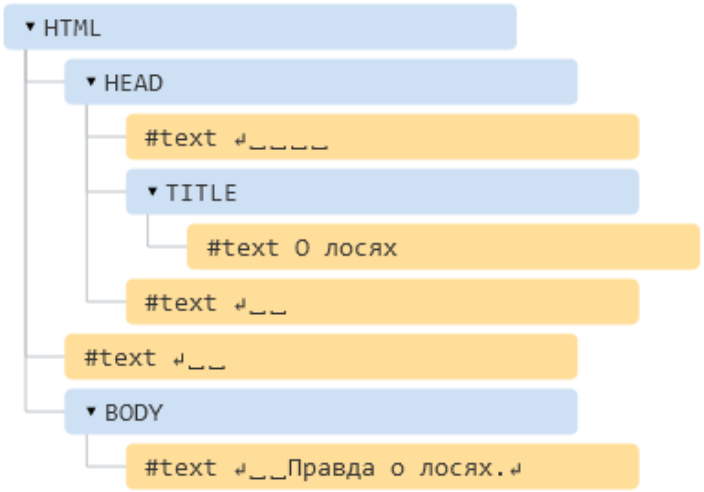


Рисунок 3.1 – Структура DOM для частини коду представленої вище

Також у Vue були використані такі плагіни як:

- Vue-router – плагін для зв'язку різних умовно кажучи сторінок веб-додатку;
- Vue-eslint – це плагін для аналізу статичного коду та виявлення проблемних зразків із можливістю автоматичного виправлення;
- Vue-babel – це плагін, що відповідає за транскомпіляцію файлів JavaScript, тобто переписує функції так, щоб вони відпрацьовували на більш старому ПЗ;
- Vue-cli-service – це обов'язковий плагін, що керує усіма зв'язками веб-додатку, вбудовується у кожен проект.

У додатку відсутня база даних у класичному її розумінні. Всі дані зберігаються у JSON-файлах, з них дані зчитуються і записуються туди ж. Такий спосіб було вибрано із двох причин:

1. Дані, що зберігаються не мають ніяких зв’язків між собою, тому не має необхідності використовувати СУБД;

2. Такий підхід пришвидшує розробку, оскільки JavaScript має всі інструменти спеціально для маніпулюванням даними у JSON-файлах.

Працює це так (рис. 3.2):

1. Користувач вводить данні у відповідне поле;

2. Ці дані проходять перевірку на коректність, тобто перевіряється чи у тих полях, де потрібно вводити цілочисельне значення, не ввели число із плаваючою комою, або чи у полі емейлу ввели саме емейл;

3. У разі валідності даних у результат записується повідмолення про успіх та відбувається запис нових дних у відповідний JSON-файл;

4. У разі не валідності даних у результат записується повідомлення про не коректність даних та вказується сама помилка, потім відповідне поле очищається;

5. Відображення результату.

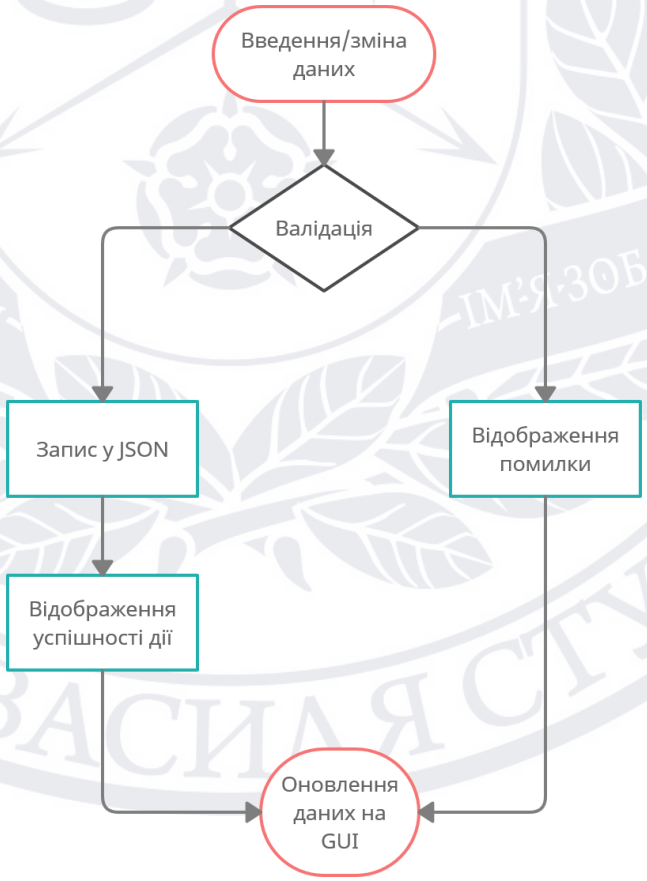


Рисунок 3.2 – Логічний опис роботи функцій валідації у додатку

3.2 Етапи реалізації веб-додатку

Всі етапи реалізації додатку можна поділити на [18]:

1. Макет. Першим етапом у реалізації веб-додатку даної роботи став вибір макету для верстання у відкритих джерелах. На цьому етапі виникла проблема у повноті макетів, тобто на одному макеті є лише домашня сторінка, а на іншому лише сторінка кімнат. Вирішенням цієї проблеми стало об'єднанням декількох макетів за одним стилем.
2. Статичні сторінки. Після визначення остаточного макету було почато верстання цього макету, тобто створення статичних сторінок за допомогою HTML5 та CSS3, після чого було додано JavaScript код.
3. HTML. При написанні коду HTML було дотримано всіх правил семантики та загальноприйнятих методів. Серед правил семантики хочу виділити базову розмітку сторінки. Схему семантики педставлено на рис. 3.3.

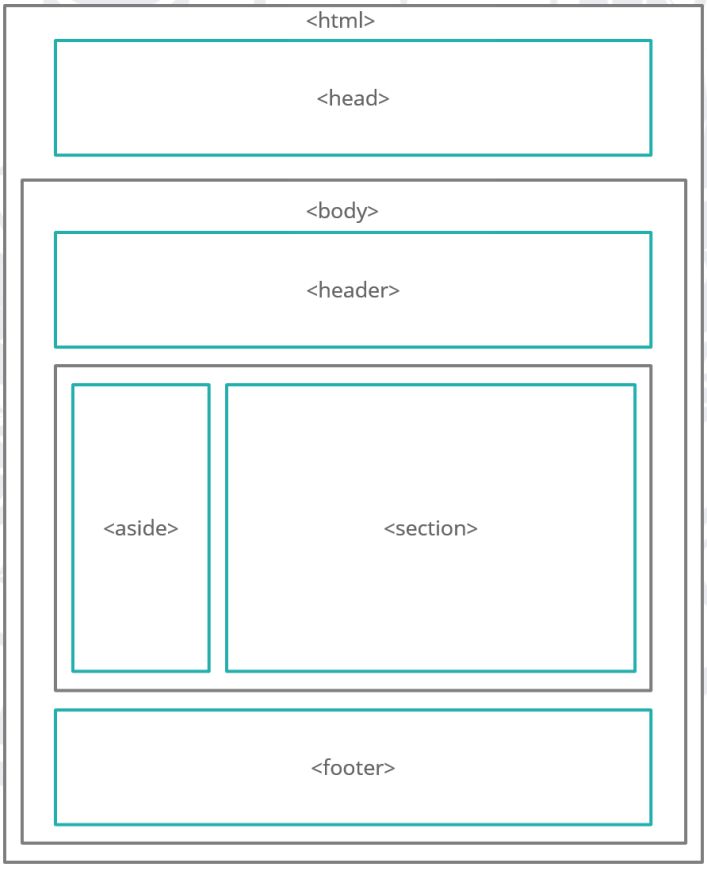


Рисунок 3.3 – Схема семантики HTML-документу

Перед початком написання коду потрібно написати тег, який показує на якій мові написаний сам код, у випадку даної роботи це такий тег:

<!DOCTYPE html>

Цей тег означає, що код у даному файлі написаний на HTML5.

Увесь код гіпертекстової розмітки записується у тег <html>, цей тег має атрибут “lang”, який приймає в себе значення коду мови.

<html lang="en">

Даний запис означає, що основна частина контенту написана англійською мовою.

Першим тегом в середині <html> йде тег <head>. У ньому вписуються всі мета-теги, назва html-документу та підключення css тегом <link>.

<meta charset="UTF-8">

<title>Document</title>

<link rel="stylesheet" href="assets/css/style.min.css">

Після тегу <head>, йде тег <body>. У ньому записується весь контент, який спочатку ділиться на три основних частини:

1. <header>
2. <main>
3. <footer>

Частина <header>. Записується зазвичай лого сайту, можливо маленьке оголошення(під час початку пандемії інтернет-магазини таким чином писали, що їхні фізичні магазини не працюють) та меню.

Логотип сайту зазвчиай роблять посиланням, що містить в собі картинку, посилання веде на головну сторінку.

<img src="assets/img/logo.png" alt="logo">

Оголошення, якщо є, то записується звичайним тегом .

Меню записується спеціальним тегом для навігації <nav>, він використовується для не лише для семантики, а й для якісного SEO, щоб

Google-робот зміг виділити меню сайту, як навігацію. У випадку даної курсової роботи меню має таку структуру:

```
<nav>
<ul class="menu_list">
<li class="menu_list_item">
<a href="index.html">
<svg>
<use xlink:href="assets/img/sprite.svg#home"></use>
</svg>
</a>
</li>
</ul>
</nav>
```

Дану структуру можна описати так – тег <nav> містить в собі тег списку <ul>, - це тег не нумерованого списку, список містить в собі пункти, тобто тег <li>, всередині цього тега вкладений тег <a>, тобто посилання, всередині якого міститься тег <svg>, цей тег оголошує елемент із форматом .svg, так оголошуються іконки, що зроблені за допомогою векторної графіки, <use> цей тег означає, то код svg буде записаний не явно, а має бути svg-файл, що і буде підключений.

Частина <main>.Записується весь головний контент, він включає в себе статті, різні секції, оголошення тощо. Серед найпопулярніших тегів можна виділити такі, як <section>, <article>, <aside>. Далі в них вже використовуються менші теги такі, як , , <p>, <img>, <a> тощо. Це все теги саме для розміщення контенту, наприклад тег <p>, це тег абзацу, а - це блок у якому може міститись інформація(ті самі . <p>. <img>). Потрібно виділити теги <img src=” ” alt=” “>та . У тегу <img src=” ” alt=” “>, те що записується значенням атрибута src являє собою шлях до зображення, якщо воно знаходиться локально, або посилання на зображення в інтернеті, значенням атрибуту alt є текст, що з'явиться у разі не

завантаження зображення(якщо шлях не вірний або зображення не завантажується з інших причин). У тегу значенням атрибуту href є посилання.

Частина <footer>.Записується інформація про автора сайту або про власника. Також можливий запис головного або другорядних меню (в тезі <nav>). Зазвичай ще в футері присутній логотип.

Після тега <footer> можливий запис тега <script src=” “></script>, цей тег відповідає за під'єднання скриптів JavaScript, значення атрибуту src є шлях до файлу зі скриптами, або тег <script></script>, цей тег дає можливість записувати скрипти JavaScript прямо у файлі HTML. У самому кінці відбувається закриття тегів <body> та <html>.

```
</body>
</html>
```

Правила семантики існують для того, щоб розробники дотримувались певного стандарту. Контролюється це пошуковими роботами браузерів, що індексують сайти.

Також варто відмітити, що даний веб-додаток був написаний за методологією БЕМ.

БЕМ(Блок-Елемент-Модифікатор) – це мотодологія веб-розробки, що дозволяє писати зрозумілу архітектуру шляхом давання логічних та зв'язаних назв. Наприклад є список елементів, перший елемент виділений червоним кольором:

```
<ul class="list">
<li class="list_item list_item--red">1</li>
<li class="list_item">2</li>
<li class="list_item">3</li>
<li class="list_item">4</li>
<li class="list_item">5</li>
</ul>
```

Паралельно із написанням коду html відбувалося і написання стилів у каскадних таблицях, тобто у css. Для цього створюється окремий файл формату .css, що підключається в html-документ у тег <head>, задопомогою тега <link>.

<link rel="stylesheet" href="assets/css/style.min.css">

Атрибут rel приймає у значення тип файлу, в даному випадку це таблиці стилів.

Атрибут href приймає у значення шлях до файлу.

CSS має простий синтаксис, спочатку вписується селектор, тобто ім'я класу/ідентифікатор/тег, а потім у фігурних дужках записується назва стилю та його значення.

```
p, .className, #id{
  color:#fff;
  background-color: #000;
  position: absolute;
  top: 0;
  left: 0;
}
```

У частині коду, представленій зверху перший рядок представляє собою селектори, де p – тег; .className – елемент із класом className; #id – елемент із ідентифікатором id. Записані ці селектори через кому, що означає – записані нижче стилі приміняться для усіх селекторів. Далі у фігурних дужках записані самі стилі, де color – це колір тексту, приймає зарезервовані назви кольорів, кольори у форматі rgb/rgba, а також кольори у шістнадцятковому коді; background-color – це колір фону, приймає такі ж формати кольору, що й color; position – це стиль позиціонування, в даному випадку означає, що елемент позиціонується або від всієї сторінки або від батьківського елементу із стилем position: relation, якщо такий існує; top та left – це стилі, що дозволяють пересувати позиціонуємий елемент.

Хоч синтаксис css доволі простий у даній курсовій роботі був використаний препроцесор SCSS, що дозволяє писати стилі швидше та більш універсальні. Він дозволяє використовувати змінні, а також просту логіку (if/else).

```
.rooms {
   display: grid;
&_item {
      width: 192px;
      height: 140px;
&:hover {
        cursor: pointer;
      }
  }
}
```

У даній частині коду стилів є важливий символ «&», він приймає вигляд батьківського селектору, тобто у першому рядку селектор - .rooms, у третьому рядку - &_item, тобто .rooms_item, а у шостому відповідно - .rooms_item:hover. Такі прості прийому дозволяють швидше писати код стилів. Особливо зручно використовувати цей прийом, якщо html-код був написаний за методологією БЕМ.

Після написання більшої частини html та css, було почато додавання коду JavaScript. У веб-додатку він використовується для створення API-запитів, роботи модальних вікон тощо. Код був написаний не на чистому JavaScript, а задопомогою бібліотеки jQuery, що значно пришвидшило час розробки.

```
$('.button').click(function (e) {
   e.preventDefault()
   $('.modal').toggleClass('show')
})
```

Показаний вище код виконує такий алгоритм: при натисканні на елемент із класом “button”, відбувається пошук елементу із класом “modal”, після того, як такий елемент знайдено відпрацьовує функція “toggleClass”, тобто якщо елемент має клас (у даному випадку “show”), то він видаляється, а якщо такого класу немає, то він додається; по суті ця функція є аналогом такого запису:

```
$('.button').click(function (e) {
    e.preventDefault()
    if($('.modal').hasClass('show')){
        $('.modal').removeClass('show')
    }else{
        $('.modal').addClass('show')
    }
})
```

Зазвичай елементи, які можуть натискати і, які не переводять на іншу сторінку записуються тегом <a>, за замовчуванням при натисканні на цей тег відбувається оновлення сторінки, щоб це відмінити використовується такий запис “e. preventDefault()”.

При написанні веб-додатку для даної курсової роботи була використана бібліотека JavaScript – axios. Axios – це JavaScript-бібліотека для виконання HTTP-запитів в Node.js. У веб-додатку ця бібліотека була використана для API-запитів до відкритих баз даних мережі інтернет, а саме до «openweathermap» та «newsapi», із першого беруться показники погоди, а із другого новини із порталу BBC News. Відповіддю на запит приходить об'єкт, що згодом парситься та показується у GUI.

```
axios
.get('https://api.openweathermap.org/data/2.5/weather?lat=49.232419&lon=28.474768&units=metric&appid=#####my_api_key####')
        .then((resp)=>{
            this.curWeather = resp.data;
            this.tempr = Math.round(this.curWeather.main.temp);
```

```
            this.temprFeel = Math.round(this.curWeather.main.feels_like);
            this.icon = 'http://openweathermap.org/img/wn/' +
this.curWeather.weather[0].icon + '@2x.png'
            console.log(this.curWeather)
            console.log(this.tempr)
        })
```

Після завершення написання статичних сторінок відбувався етап перетягування їх на фреймворк Vue. При створені проекту Vue автоматично творюються папки, що утворюють структуру проекту [19]:

- Assets;
- Components;
- Router;
- Views.

Assets. Містить у собі усі файли стилів, усі файли коду JavaScript, усі масиви даних та усі зображення.

Components. Містить у собі частини коду. При створенні сторінок у Vue, вони зазвичай розбиваються на кілька компонентів і додаються окремо, у саме у цій папці зберігаються компоненти. Також однією із причин розбиття на компонент може бути дублювання коду.

Router. Містить у собі один файл за замовчуванням index.js. У цьому файлі записані динамічні шляхи між документами, тобто у Vue переключення між різними сторінками здійснюється задопомогою цього файлу.

Views. Містить у собі самі сторінки, на які і посилається файл index.js у папці router.

Спочатку були створені файли у папці views тобто самі шаблони сторінок, потім був етап розбору повних сторінок на компоненти, після чого був відредагований код файлу index.js у папці router.

Код html також зазнав змін, всі теги <a>, що посилаються на інші сторінки, тобто тепер файли папки view, були замінені на такі конструкції:

```
<router-link to="/todo"></router-link>
```

Цей запис означає, що при натиснені буде здійснений перехід на файл папки views, що у файлі index.js у папці router, записаний як “/todo”.

Всі елементи, що повторювались в межах одного компоненту, переписані за допомогою вбудованих циклів Vue.

```
<ul class="todo_list">
<li class="todo_list_item" v-for="(d, t) in todos" :key="t">
<span class="todo_list_item_link">{{d.description}}</span>
</li>
</ul>
```

Раніше динамічні дані підставляв на свої місця код JavaScript, це було перероблено, оскільки синтаксис Vue дозволяє ініціювати виклики даних прямо з того місця, де вони потрібні.

```
<div class="date_day">{{this.dateDay}} {{this.dateMonth}}
{{this.dateYear}}</div>
```

А також частини стилів та скриптів були записані локально до кожної сторінки у файл, а не глобально, як раніше.

РОЗДІЛ 4. Тестування веб-додатку

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

В рамках даної курсової роботи було проведено альфа та бета тестування. Це методології перевірки споживачів, які допомагають зміцнити впевненість у запуску продукту і, таким чином, призводять до успіху товару на ринку.

4.1 Альфа-тестування веб-додатку

Альфа-тестування ІС розділяється на функціональне тестування, тестування сумісності та тестування безпеки.

Функціональне тестування – тестування функціональностей і операційної поведінки продукту з метою переконатися, що вони відповідають специфікаціям. Таке тестування опирається лише на вихідні дані. Це тестування потрібно для переконання, що продукт відповідає потрібній функціональній специфікації, заданій у технічному завданні.

Тестування сумісності – це тестування перевіряє сумісість додатку із іншими елементами системи, в якій воно працює, тобто браузери, операційні системи або апаратною частиною девайсу. Це тестування оцінка того, на скільки добре ПЗ працює в певному браузері, ОС та АЗ.

Тестування безпеки – тестування для пошуку недоліків з точки зору безпеки додатку.

При альфа-тестуванні веб-додатку «Панель керування розумним будинком», перевірено такі сценарії поведінки системи:

- Протестовано коректність авторизації;

- Протестовано автоматичний перехід на домашню сторінку після авторизації;
- Протестовано коректне відображення результатів API-запитів(погода, новини);
- Протестована сумісність веб-додатку на різних браузерах;
- Протестована можливість вводу даних, їх видалення та зміни.

Веб-додаток на різних браузерах працює однаково, проте варто зауважити, що у браузері Firefox Mozilla не спрацьовує стиль filter: blur(3px) (рис. 4.1). Це відбувається не через проблеми у додатку, а тому що цей браузер просто не підтримує цей стиль, на відміну від інших браузерів (рис. 4.2).

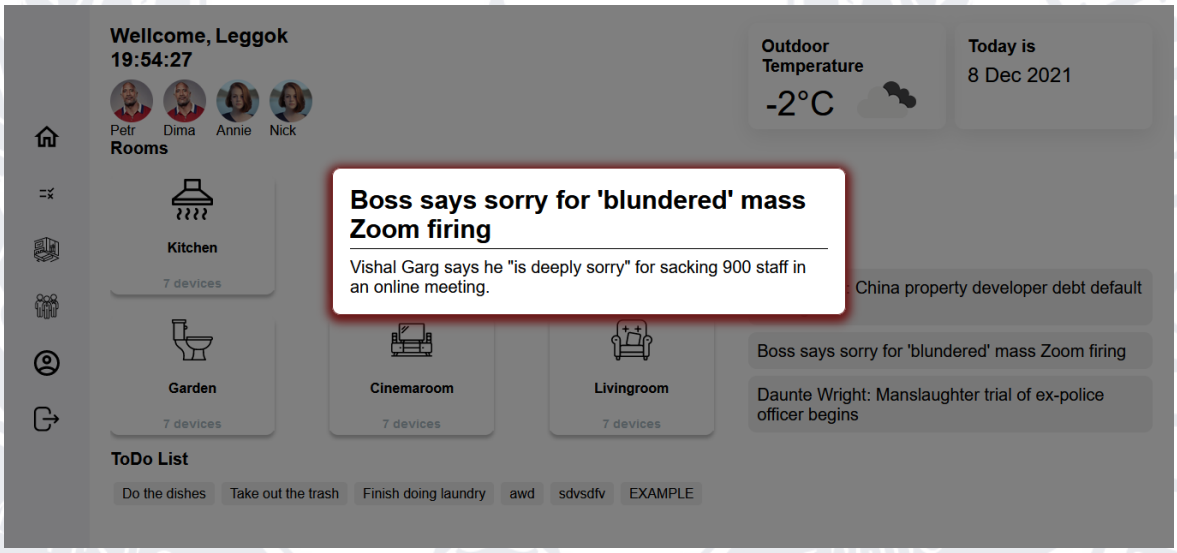


Рисунок 4.1 – Веб-додаток у Firefox Mozilla

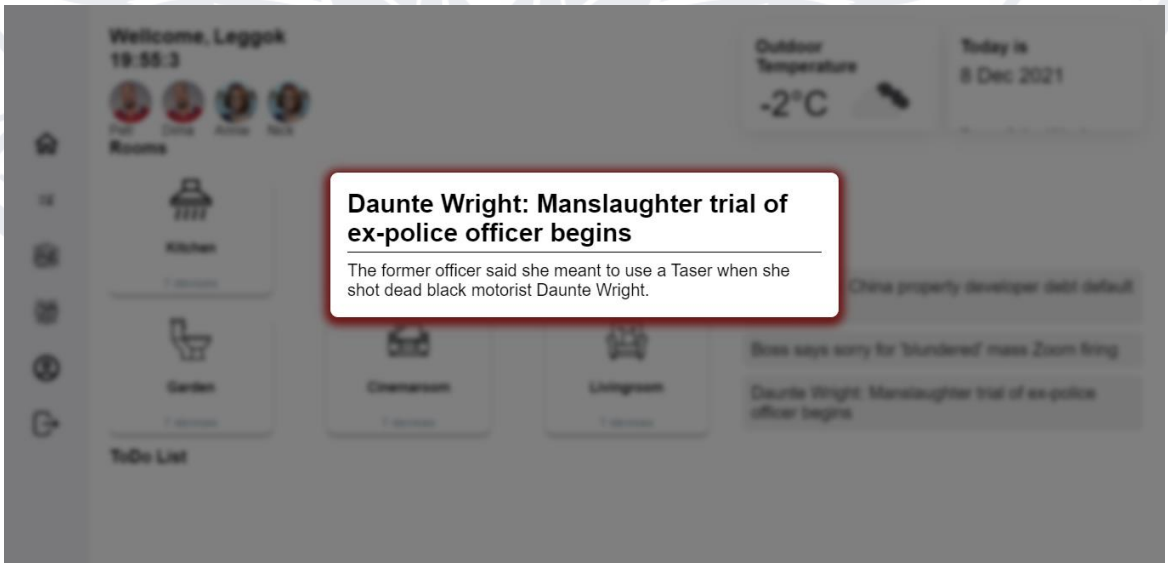


Рисунок 4.2 – Веб-додаток у Google Chrome

Також було окремо протестовано сторінку ToDoList. Перевірено коректність відображення та можливість додавання нових пунктів. Рисунки 4.3 – 4.6 показують порядок дій:

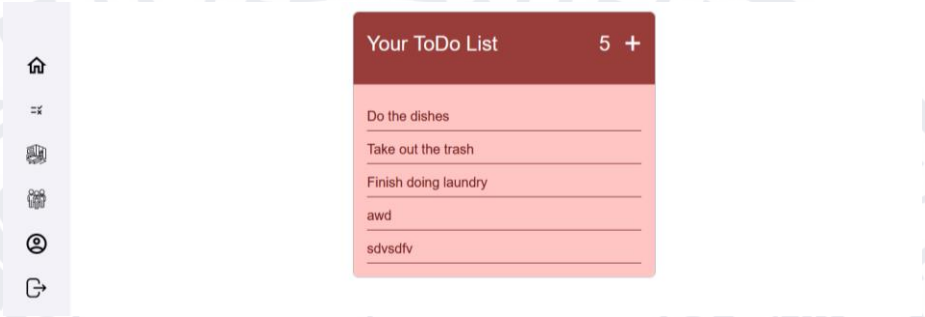


Рисунок 4.3 – Початковий вигляд сторінки «ToDoList»

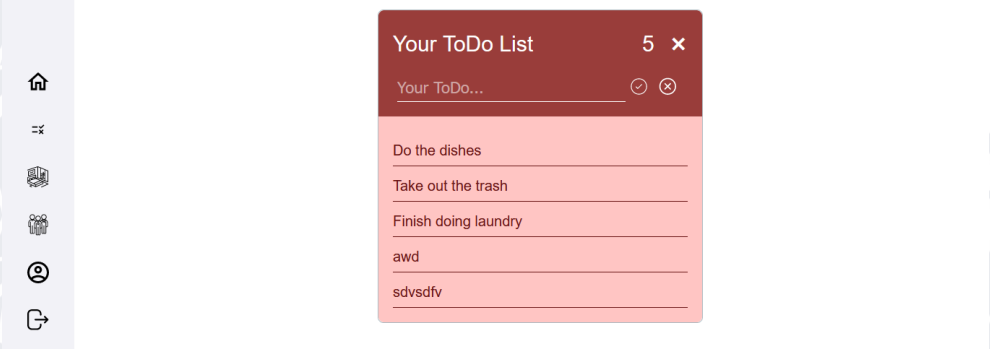


Рисунок 4.4 – Вигляд сторінки «ToDoList», після натискання на кнопку додавання елементу «+»

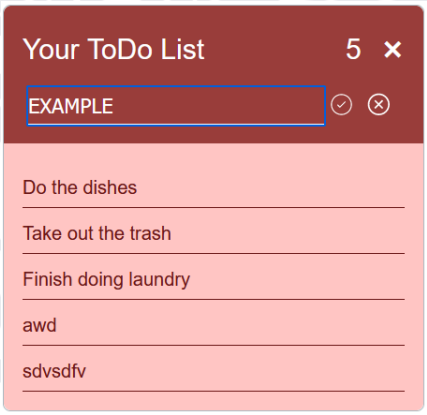


Рисунок 4.5 – Вигляд сторінки «ToDoList», під час введення назви елементу

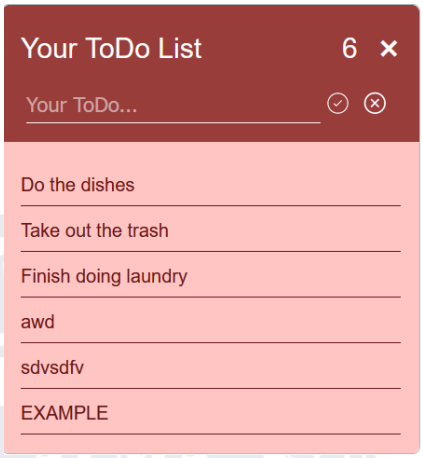


Рисунок 4.6 – Вигляд сторінки «ToDoList», після додавання нового елементу

Після виконання цих пунктів відбувається запис нового елементу у LocalStorage, а потім і запис у потрібний JSON-файл.

LocalStorage[6]– це вастивість тільки для читання на window інтерфейс, дозволяє отримати доступ до Storage об'єкту. Збережені дані зберігаються через сеанси браузера. У LocalStorage зберігаються такі дані як наприклад налаштування сайту (вибір темної теми або налаштування інтерфейсу).

LocalStorage аналогічний SessionStorage (рис.4.7), за винятком того, що, хоча LocalStorageдані не мають терміну придатності, SessionStorage дані очищаються при завершенні сеансу сторінки, тобто при закритті сторінки. В LocalStorage дані для документа, завантаженого в сеансі «приватного перегляду» або «інкогніто», видаляються при закритті останньої «приватної» вкладки (рис. 4.8).

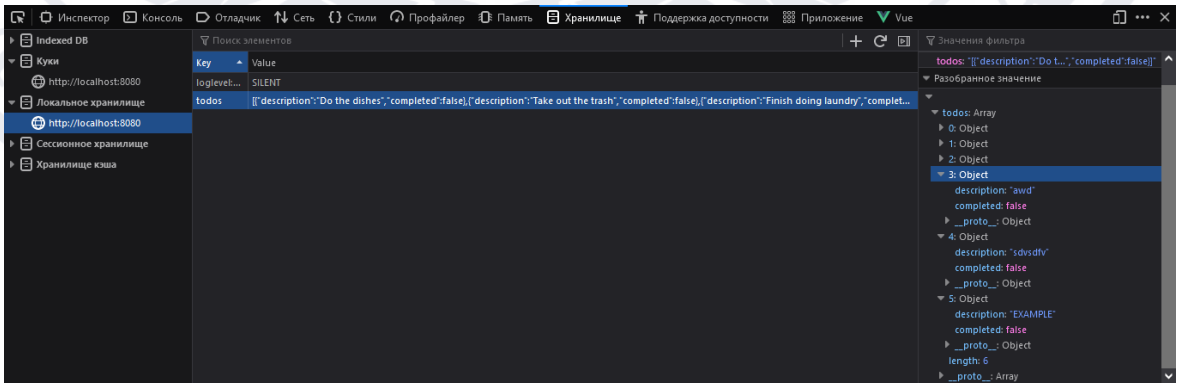


Рисунок 4.7 – Скріншот панелі розробника у вкладці Storage

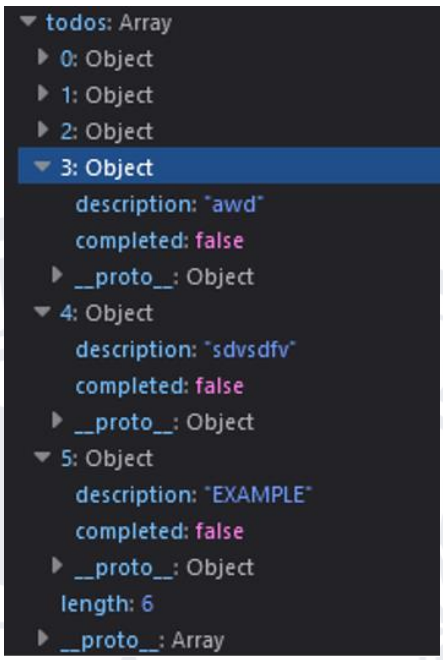


Рисунок 4.8 – Скріншот конкретного об'єкту LocalStorage, на якому видно, що до нього було додано нові елементи

todos = JSON.parse(localStorage.getItem('todos'))

УрядкупредставленомувищевідбуваєтьсязчитуваннядanихізlocalStorage, їхнійпарсингтазаписузмінну «todos».

Об'єкт JSON [7], доступнийувсіхсучаснихбраузерах, включаєдвакорисніметодидляроботизконтентомуформаті JSON: parse та stringify.

JSON.parse() – цеметод, щоберерядок JSON ітрансформуєйоговоб'єкт JavaScript.

JSON.stringify() – цеметод, щобереоб'єкт JavaScript ітрансформуєйоговрядок JSON.

4.2 Бета-тестування додатку

Тестування зручності використання – це тестування «доброзичливості» додатку для користувача. При цьому виді тестування потрібно переконатись у простоті та ефективності використання продукту. Ідеальне проходження цього тестування вважається мінімалізм та інтуїтивно зрозумілий інтерфейс.

При бета-тестуванні веб-додатку було перевірено такі сценарії поведінки:

- Інтуїтивно зрозумілий інтерфейс (двом тестувальникам);
- Шрифти відпвідають загальним вимогам (розмір та жирність) та відповідають початковому дизайну;
- Орфографічних та граматичних помилок не виявлено;
- Адаптивний дизайн(передбачено будь-який розмір екрану, тому об'єкти розміщені нормально)
- Протестована можливість переходу на іншу сторінку веб-додатку у будь-який момент часу;

У додатку А представлені скріншоти веб-додатку

ВИСНОВОК

У результаті виконання даної бакалаврської роботи було розроблено веб-додаток «Панель керування розумним будинком» з інтуїтивно зрозумілим графічним інтерфейсом користувача, API-запитами до відкритих баз даних, а також з коректною роботою усіх модулів та функцій, що є у веб-додатку. Також веб-додаток був розроблений адаптивним, задля комфортного та ефективного користування для усіх користувачів на різних екранах пристроїв. Була наведена основна інформація про концептуальну модель інформаційної системи «Панель керування «Розумним будинком»», її можливості та функціонал, а також про технології та програмні засоби для написання веб-додатку. Також було наведено логічну модель веб-додатку цієї курсової роботи, її можливості та обмеження. Була надана інформація про всі компоненти системи та їхні можливості.

В майбутньому планується виконати такі завдання для модернізації веб-додатку:

- Доповнення інтерфейсу;
- Покращення анімації;
- Розробка системи безпеки веб-додатку;
- Збільшеня функціоналу.

СПИСОК ПОСИЛАНЬ

1. Vue.js [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Vue.js
2. MVVM [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Model-View-ViewModel
3. HTML5[Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/HTML5
4. CSS3[Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/CSS
5. JavaScript[Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/JavaScript
6. LocalStorage [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
7. JSON.parse/JSON.stringify [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/community/tutorials/js-json-parse-stringify
8. Tesla Nikola Method of and apparatus for controlling mechanism of moving vessels and vehicles//Patent 613809. — United States Patent and Trademark Office, 8 November 1898
9. Spicer Dag If You Can't Stand the Coding, Stay Out of the Kitchen. — Dr. Dobb's Journal, Retrieved 2010-09-02
10. Anogianakis George (1997) Advancement of assistive technology. — IOS Press.
11. 1.5 Million Home Automation Systems Installed in the US This Year. [Електронний ресурс] – Режим доступу:www.abiresearch.com.
12. Gerhart James Home Automation and Wiring//McGraw-Hill Professional, 31 March 1999. — ISBN 0-07-024674-2.
13. Бондарев О. Хто в дом і господар. Розумні будинки через кілька років набудутьширокої популярності / Олексій Бондарев. // Корреспондент. – 2012. – №30. – С. 42–46.

14. Дужак І. О. Розумний будинок / І. О. Дужак. // Автоматизація технологічних ібізнес-процесів. Одеська національна академія харчових технологій. – 2013. – №13. – С. 31.

15. Климатичні системи і автоматика розумногобудинку [Електронний ресурс] – Режим доступу до ресурсу: http://ingsvd.ru/main/ventilation/56-klimaticheskiesistemy-i-avtomatika.html

16. Мультирум [Електронний ресурс]– Режим доступу доресурсу: http://hifidom.com.ua/statti/multirum.

17. Сафронова О.О. Альтернативні методи освітлення в контекст і вирішенняпитання підвищення енергоефективності інтер'єрного простору ВНЗ / Сафронова О.О. // Н.-техн. зб.: Вісник КНУТД – К.:КНУТД , 2013, – Вип. №6 (74) – С. 166-174 (ф)

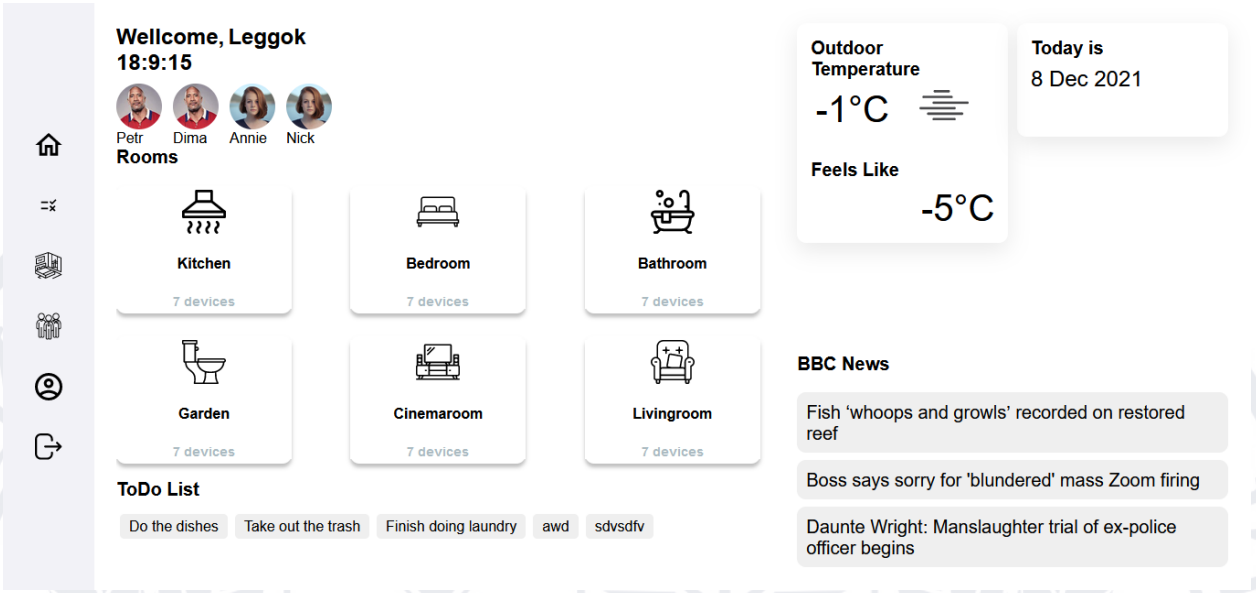
18. Harper, Richard. Inside the Smart House. – London: Springer. – 2003.

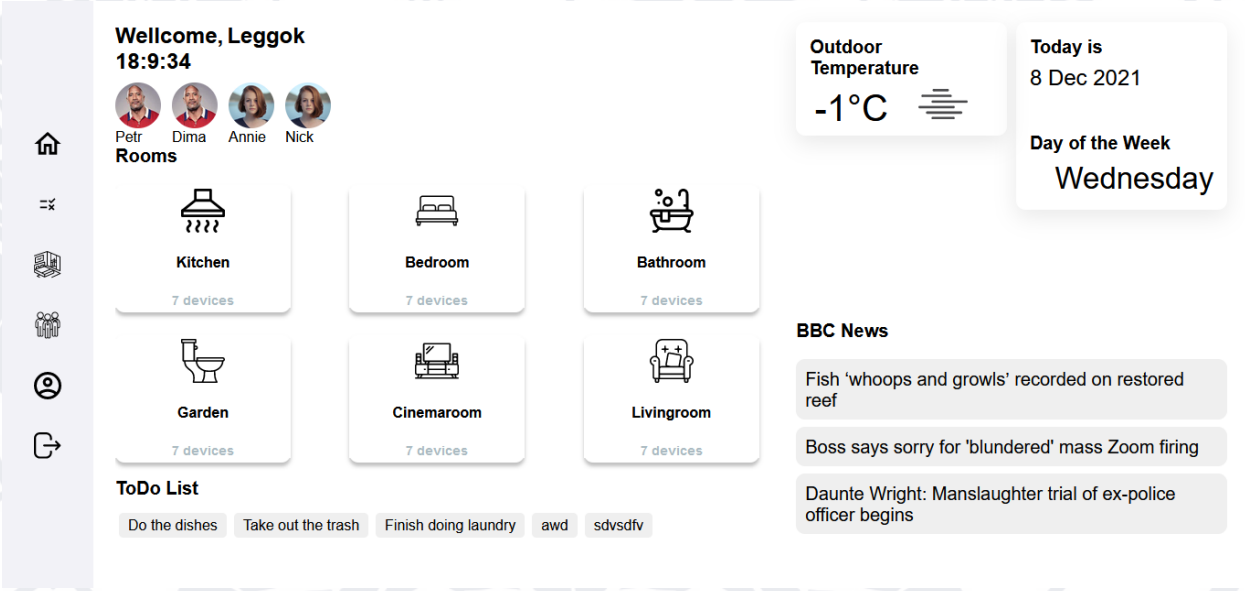
19. Mann William C. The state of the science // Smart technology for aging, disability andindependence. – John Wiley and Sons, 7 July 2005. – ISBN 0-471-69694-3.

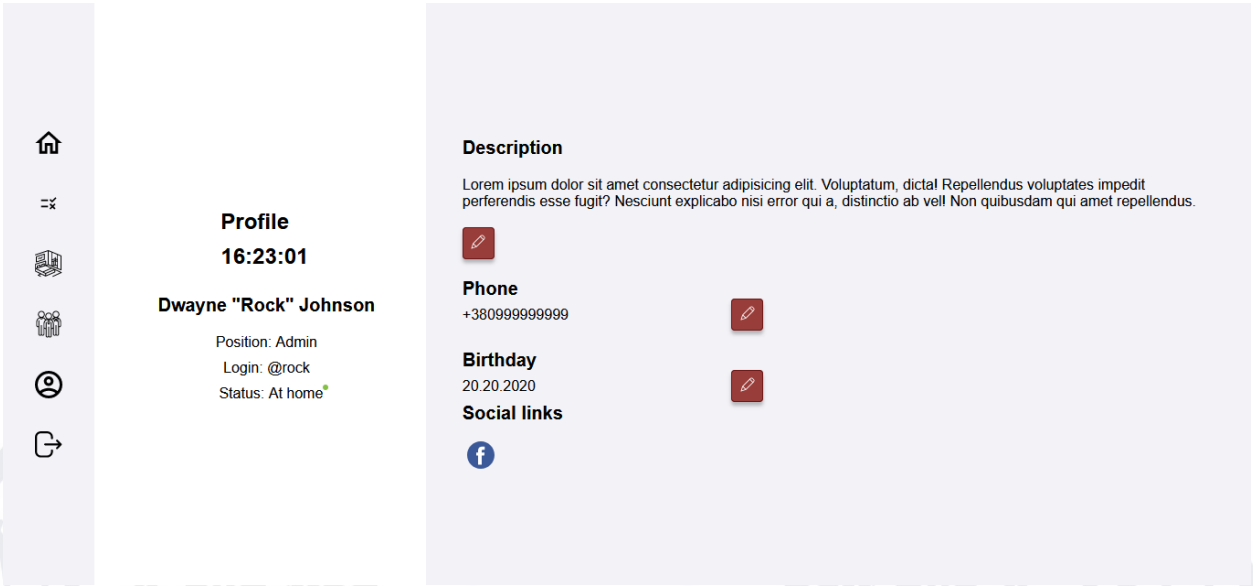
ДОДАТКИ

ДОДАТОКА

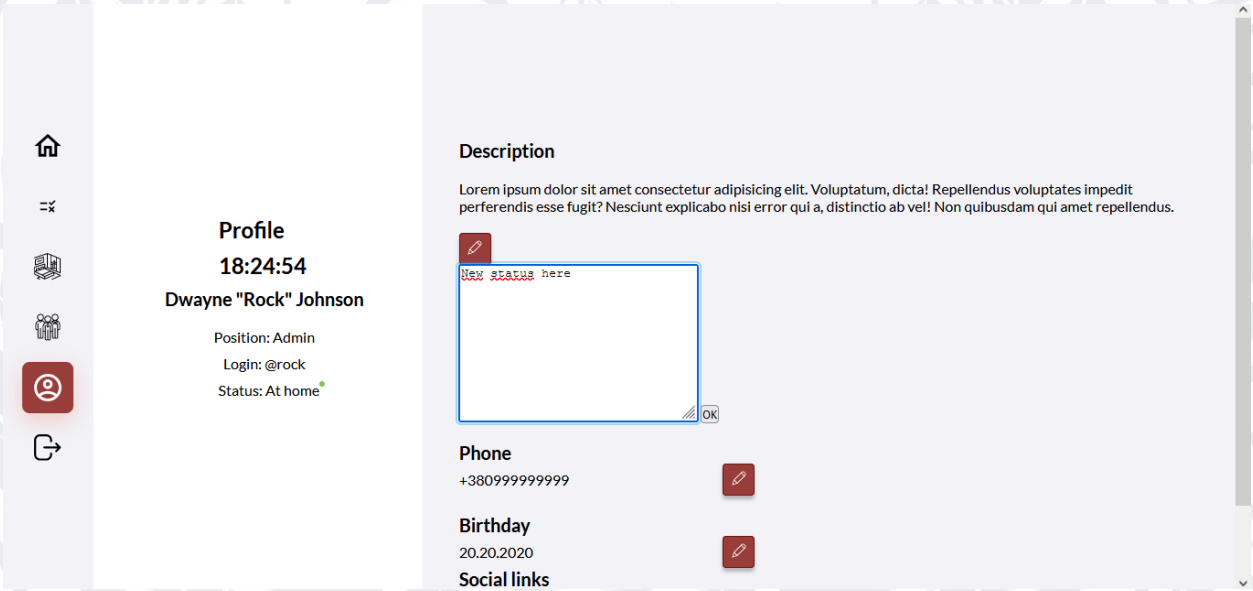
Скріншоти веб-додатку







ДОДАТОК А(продовження)



Profile

18:26:18

Dwayne "Rock" Johnson

Position: Admin

Login: @rock

Status: At home

Description

New status here

Phone

+380999999999

Birthday

20.20.2020

Social links

ДОДАТОК А(продовження)

ДОДАТОК Б
Лістинг коду Home.vue

```
<template>
<main class="home_page">
<div class="left">
<div class="wellcome">
<div class="hw">Wellcome,
</div>
<div class="user_name">Leggok</div>
</div>
<div class="time">16:23:01</div>
<div class="members">
<ul class="members_list">
<li class="members_list_item">
<a href="#">
<div class="ava">
<img src="../assets/img/members/f.jpg" alt="">
</div>
                    Petr
</a>
</li>
</ul>
</div>
<div class="rooms_wrap">
<div class="title">Rooms</div>
<stackrooms/>
</div>
<div class="todo">
<div class="title">
                ToDo List
</div>
<ul class="todo_list">
<li class="todo_list_item" v-for="(d, t) in todos" :key="t">
<span class="todo_list_item_link">{{d.description}}</span>
</li>
</ul>
</div>
</div>
<div class="right">
<div class="right_top">
<div class="weather">
<div class="weather_title title">Outdoor Temperature</div>
<div class="weather_show">
<div class="weather_show_temperature">{{ this.tempr }}&#176;C</div>
<div class="weather_show_icon">
<img :src=this.icon alt="">
</div>
</div>
<div class="weather_hover">
<div class="weather_title title feel">Feels Like</div>
```

ДОДАТОКБ (продовження)

```
<div class="weather_show_temperature_feel">{{ this.temprFeel }}&#176;C</div>
</div>
</div>
<div class="date">
<div class="date_title title">Today is</div>
<div class="date_day">{{this.dateDay}} {{this.dateMonth}} {{this.dateYear}}</div>
<div class="date_title title dow">Day of the Week</div>
<div class="date_day_week">{{this.weekDay}}</div>
</div>
</div>
<div class="news">
<div class="news_title title">BBC News</div>
<ul class="news_list">
<li class="news_list_item">
<div class="show_news">
<div class="news_head">{{this.newsHead[0]}}</div>
</div>
<div class="news_modal">
<div class="modal_body">
<h2 class="newsHead">{{this.newsHead[0]}}</h2>
<div class="news_desc">{{this.newsDesc[0]}}</div>
</div>
</div>
</li>
<li class="news_list_item">
<div class="show_news">
<div class="news_head">{{this.newsHead[1]}}</div>
</div>
<div class="news_modal">
<div class="modal_body">
<h2 class="newsHead">{{this.newsHead[1]}}</h2>
<div class="news_desc">{{this.newsDesc[1]}}</div>
</div>
</div>
</li>
<li class="news_list_item">
<div class="show_news">
<div class="news_head">{{this.newsHead[2]}}</div>
</div>
<div class="news_modal">
<div class="modal_body">
<h2 class="newsHead">{{this.newsHead[2]}}</h2>
<div class="news_desc">{{this.newsDesc[2]}}</div>
</div>
</div>
</li>
</ul>
</div>
```

```
</div>
</main>
</template>

<style scoped>
.newsHead{
    border-bottom: 1px solid #000;
    padding-bottom: 5px;
    margin-bottom: 10px;
}
.news_list_item{
    margin: 8px 0;
    transition: all .2s ease;
    font-size: 20px;
    background: #efefef;
    padding: 10px;
    border-radius: 10px;
}
.show_news:hover{
    transition: all .2s ease;
    cursor: pointer;
    text-decoration: underline;
}
.show_news{
    transition: all .2s ease;
    display: flex;
    align-items: center;
}
.news{
    width: 100%;
}
.news_author{
    position: relative;
    width: 25%;
}
.news_head{
    position: relative;
}
.news_modal{
    display: none;
    width: 100%;
    height: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(0,0,0,.5);
    backdrop-filter: blur(3px);
}
.show{
    display: block;
}</style>
```

ДОДАТОКБ(продовження)

```
<script>
import stackrooms from '@/components/stack_rooms.vue'
import JQuery from 'jquery'
import axios from 'axios'
const $ = JQuery

export default {
  components: {
    stackrooms
  },
    data(){
       return{
          tempr: 0,
          temprFeel: 0,
          dateDay: 0,
          dateMonth: 0,
          dateYear: 0,
          weekDay: 0,
          week:['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
          month:['Jan', 'Feb', 'Mar', 'Thursday', 'April', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
'Nov', 'Dec'],
          icon: 0,
          newsAuthor: [],
          newsHead: [],
          newsDesc: [],
          todos: []
       }
    },
   mounted () {
    $('.todo_list_item_link').click(function (e) {
       e.preventDefault()
       $(this).toggleClass('done')
    })
    $('.news_list_item').click(function (e) {
       e.preventDefault()
       if($(this).children('.news_modal').hasClass('show')){
          $(this).children('.news_modal').removeClass('show')
       }else{
          $('.news_modal').removeClass('show')
          $(this).children('.news_modal').addClass('show')
       }
    })
    window.setInterval(() => {
     var dt = new Date()
     $('.time').text(dt.getHours() + ':' + dt.getMinutes() + ':' + dt.getSeconds())
    }, 1000);
    var now = new Date();
    this.dateDay = now.getDate()
    this.dateMonth = this.month[now.getMonth() + 1]
    this.dateYear = now.getFullYear()
    this.weekDay = this.week[now.getDay() - 1]
```

ДОДАТОК Б (продовження)

```
        axios
.get('https://api.openweathermap.org/data/2.5/weather?lat=49.232419&lon=28.474768&units=met
ric&appid=8ca11de0bcfacc5ccdef6c5290c7aa37')
        .then((resp)=>{
                this.curWeather = resp.data;
                this.tempr = Math.round(this.curWeather.main.temp);
                this.temprFeel = Math.round(this.curWeather.main.feels_like);
                this.icon = 'http://openweathermap.org/img/wn/' + this.curWeather.weather[0].icon
+ '@2x.png'
                console.log(this.curWeather)
                console.log(this.tempr)
            })
          axios
            .get('https://newsapi.org/v2/top-headlines?sources=bbc-
news&apiKey=f0f1c9a0ea70445da339e54f1afac1d7')
            .then((resp)=>{
                this.news = resp.data;
                this.newsAuthor = [this.news.articles[0].author, this.news.articles[1].author,
this.news.articles[2].author]
                this.newsHead = [this.news.articles[0].title, this.news.articles[1].title,
this.news.articles[2].title]
                this.newsDesc = [this.news.articles[0].description, this.news.articles[1].description,
this.news.articles[2].description]
                console.log(this.news)
            })
        },
        created(){
          if(JSON.parse(localStorage.getItem('todos')).length >= 1){
            this.todos =  JSON.parse(localStorage.getItem('todos'))
          }else{
            this.todos = [
              { description: "Do the dishes", completed: false },
              { description: "Take out the trash", completed: false },
              { description: "Finish doing laundry", completed: false }
            ]
          }
        }
    }

    </script>
```

ДОДАТОКВ
Лістингкодуrouter/index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'
Vue.use(VueRouter)
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    component: () => import('../views/About.vue')
  },
  {
    path: '/todo',
    name: 'todo',
    component: () => import('../views/todo_list.vue')
  },
  {
    path: '/rooms',
    name: 'rooms',
    component: () => import('../views/rooms.vue')
  },
  {
    path: '/one_room',
    name: 'one_room',
    component: () => import('../views/one_room.vue')
  },
  {
    path: '/members',
    name: 'members',
    component: () => import('../views/members.vue')
  },
  {
    path: '/profile',
    name: 'profile',
    component: () => import('../views/profile.vue')
  },
  {
    path: '/login',
    name: 'login',
    component: () => import('../views/login.vue')
  }]
const router = new VueRouter({
  routes
})
export default router
```

ДОДАТОКГ
Лістингкодуtodo_list.vue

```
<template>
<main class="todo">
<todo-list list-name="Your ToDo List" />
</main>
</template>

<script>
import TodoList from "@/components/NewTask.vue";
export default {
  name: "App",
  components: {
TodoList
  }
}
</script>
```

ДОДАТОКД
Лістингкодуrooms.vue

```
<template>
<section class="main_rooms">
<div class="top">
<div class="main_titlehw">
        Rooms
</div>
<div class="time">16:23:01</div>
</div>
<div class="room_carouselswipermySwiper">
<div class="swiper-wrapper">
<div class="swiper-slide">
<div class="room_carousel_content">
<div class="room_carousel_name title">Kitchen</div>
<div class="boxes">
<div class="inside">
<div class="where">This room</div>
<div class="vars">
<div class="temp">
<div class="room_carousel_temperature-word">Temp</div>
<div class="room_carousel_temperature-number">20C</div>
</div>
<div class="hum">
<div class="room_carousel_hum-word">Hum</div>
<div class="room_carousel_hum-number">50%</div>
</div>
</div>
</div>
<div class="outside">
<div class="where">Outside</div>
<div class="vars">
<div class="temp">
<div class="room_carousel_temperature-word">Temp</div>
<div class="room_carousel_temperature-number">20C</div>
</div>
<div class="hum">
<div class="room_carousel_hum-word">Hum</div>
<div class="room_carousel_hum-number">50%</div>
</div>
</div>
</div>
</div>
<div class="light">
<div class="room_carousel_light-word where">Room Lighting</div>
<div class="room_carousel_light-number">80%</div>
</div>
</div>
<ul class="room_carousel_list">
<li class="room_carousel_list_item fridge">
<svg>
```

ДОДАТОКД(продовження)

```
<use xlink:href="../assets/img/tech/tech.svg#fridge"></use>
</svg>
                Fridge</li>
<li class="room_carousel_list_item microwave">
<svg>
<use xlink:href="../assets/img/tech/tech.svg#microwave"></use>
</svg>
                Microwave</li>
</ul>
</div>
</div>
<div class="dropdown">
<a href="#" class="drop">
          Rooms
</a>
<div class="down">
<stackrooms/>
</div>
<div class="shadow"></div>
</div>
</div>
</section>
</template>
```

ДОДАТОКЕ

Лістингкодуprofile.vue

```
<template>
<main class="profile">
<div class="bot">
<section class="profile_left">
<div class="top">
<div class="main_titlehw">
                Profile
</div>
<div class="time">16:23:01</div>
</div>
<div class="profile_img">
<imgsrc="assets/img/members/f.jpg" alt="">
</div>
<div class="profile_name title">
              Dwayne "Rock" Johnson
</div>
<div class="profile_position">Position: Admin</div>
<div class="profile_login">Login: @rock</div>
<div class="profile_statusmember_status">Status: At home</div>
</section>
<section class="profile_right">
<div class="cont">
<div class="profile_title title">Description</div>
<div class="profile_sub_title">Lorem ipsum dolor sit ametconsecteturadipisicingelit.
Voluptatum, dicta! Repellendusvoluptatesimpeditperferendisesse fugit? Nesciuntexplicabo nisi
error qui a, distinctio ab vel! Non quibusdam qui ametrepellendus.</div>
<a id='change_descr' class="change" href="#">
<svg>
<use xlink:href="../assets/img/icons/pen_small.svg#surface1"></use>
</svg>
</a>
<div class="profile_descr">
<form action="#" onsubmit="change_descr()" class="form_descr">
<textarea name="description" id="profile_change_descr" placeholder="Change status"
cols="30" rows="10"></textarea>
<button type="submit">OK</button>
</form>
</div>
<div class="profile_phone">
<div class="phone_title title">Phone</div>
<div class="pho">
<div class="phone_number">+380999999999</div>
<a id='change_phone' class="change" href="#">
<svg>
<use xlink:href="../assets/img/icons/pen_small.svg#surface1"></use>
</svg>
</a>
</div>
```

ДОДАТОК Е (продовження)

```
<form action="#" onsubmit="change_phone()" class="form_phone">
<input type="text" name="phone" placeholder="Phone">
<button type="submit">OK</button>
</form>
</div>
</div>
<div class="birthday_title title">Birthday</div>
<div class="birth">
<div class="birth_date">20.20.2020</div>
<a id='change_birth' class="change" href="#">
<svg>
<use xlink:href="../assets/img/icons/pen_small.svg#surface1"></use>
</svg>
</a>
</div>
<div class="profile_birthday">
<form class="profile_birthday_form">
<label>Birthday:
</label>
<select id="year" name="yyyy" onchange="change_year(this)"></select>
<select id="month" name="mm" onchange="change_month(this)"></select>
<select id="day" name="dd"></select>
</form>
</div>
<div class="social">
<div class="social_title title">Social links</div>
<ul class="social_list">
<li class="social_list_item">
<a class="social_list_item_link" href="#">
<svg>
<use xlink:href="../assets/img/social/social.svg#facebook"></use>
</svg>
</a>
</li>
<li class="social_list_item">
<a class="social_list_item_link" href="#">
<svg>
<use xlink:href="../assets/img/social/social.svg#instagram"></use>
</svg>
</a>
</li>
<li class="social_list_item">
<a class="social_list_item_link" href="#">G</a>
</li>
</ul>
</div>
</section>
</div>
</main>
</template>
```

ДОДАТОК Ж

Лістингкодуlogin.vue

```
<template>
<main class="login_page">
<div class="container">
<div class="title">Login</div>
<div class="sign">
<form action="#">
<input type="text" name="login" id="login" placeholder="Login">
<input type="password" name="password" id="password"
placeholder="Password">
<button type="submit" id="submit">Login</button>
</form>
</div>
</div>
</main>
</template>
```

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Сітько Олександр Тарасович
Прізвище, ім’я, по батькові
Факультет інформаційних і прикладних технологій
Факультет
122 Комп’ютерні науки
Шифр і назва спеціальності
Сучасні інформаційні технології та програмування
Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка панелі управління розумним будинком» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю,. що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

дата підпис