Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра інформаційних технологій електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему Розробка веб-додатку керування системою доступу до ліфтового обладнання для ОСББ

Виконав: студент(ка) 4 курсу, групи РТ-618сп

Спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Автоматизація, мехатроніка та робототехніка»

Кіщак О. А.
(прізвище та ініціали)

Керівник Фурманова Н. І.
(прізвище та ініціали)

Рецензент Неласа Г. В.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет Інститут інформатики та радіоелектроніки, Факультет радіоелектроніки та телекомунікацій
Кафедра інформаційних технологій електронних засобів
Ступінь вищої освіти бакалавр
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
(код і найменування)
Освітня програма (спеціалізація) Автоматизація, мехатроніка та робототехніка
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТЕЗ НУЗП
Сіренич Євген Вікторович
«26» квітня 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кіщак Олександр Анатолійович
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка веб-додатку керування системою доступу до ліфтового обладнання для ОСББ
керівник проєкту (роботи) Фурманова Наталія Іванівна, к.т.н, доцент каф.ІТЕЗ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «26» квітня 2021 року №159
2. Строк подання студентом проєкту (роботи) 7 червня 2021 року
3. Вихідні дані до проєкту (роботи) системи керування ліфтовим обладнанням

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області, розробка програмного забезпечення, тестування програмного забезпечення, висновки, перелік посилань.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
55 рисунків; 3 таблиці; 19 слайдів

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 - 3	Фурманова Н.І., доц.		
Нормоконтроль	Поспеєва І.Є., ст. викладач		

7. Дата видачі завдання «26» квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Огляд існуючих систем керування ліфтовим обладнанням	26.04.21	
2	Постановка технічного завдання	28.04.21	
3	Огляд принципів роботи сучасних веб-додатків	29.04.21	
4	Розробка веб-додатку	03.05.21	
5	Тестування веб-додатку	29.05.21	
6	Оформлення ПЗ та захист дипломного проекту	01.06.21	

Студент(ка) _____ Кіщак О. А.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Фурманова Н. І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 58 с., 3 табл., 55 рис., 11 джерел.

Об'єкт розробки: система керування доступом до ліфтового обладнання.

Предмет розробки: веб-додаток для керування доступом до ліфтового обладнання для ОСББ.

Для здійснення поставленої мети необхідно було вирішити наступні задачі:

– ознайомитись з архітектурою і особливостями організації веб-додатків і принципами взаємодії клієнта і сервера;

– провести аналіз технологій, присутніх на ринку, для розробки сучасних веб-додатків та виконати розробку веб-додатку для перегляду інформації про користування ліфтами для об'єднання співвласників багатоквартирних будинків;

– розробити заходи щодо техніки безпеки при роботі за комп’ютером.

HTML, JAVASCIPT, ВЕБ-ДОДАТОК, ДОСТУПНІСТЬ, ЛІФТ, ОСББ, ПРОГРАМУВАННЯ, СЕРВЕР, СИСТЕМА КОНТРОЛЮ ДОСТУПУ, ТЕХНОЛОГІЯ, ФРЕЙМВОРК, ФРОНТЕНД.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface

AJAX – asynchronous javascript and XML

CLI – command line interface

DOM – document object model

JSON – JavaScript object notation

OSI – the Open Systems Interconnection model

SEO – search engine optimization

SOAP – Simple Object Access Protocol

SPA – single page application

UI – user interface

UX – user experience

ОСББ – об'єднання співвласників багатоквартирного будинку

ОС – операційна система

ПЗ – програмне забезпечення

ВСТУП

Веб-додаток – клієнт-серверний додаток, основна частина якого міститься на вилученому сервері, а призначений для користувача інтерфейс відображається в браузері у вигляді веб-сторінок. Сьогодні у кожної компанії, установи, тощо є свій сайт.

Сайти, зазвичай, виконують маркетингову роль, але зараз з’являється все більше веб-додатків, які здатні робити корисніші речі, ніж просте розміщення контенту для залучення трафіку, наприклад, робота з документами, електронними підписами, стрімінгові сервіси, редагування зображень онлайн, тощо.

Веб-сервіс менш вимогливий до ресурсів комп'ютера, ніж «настільний» додаток, оскільки всі складні обчислення відбуваються на стороні сервера.

У онлайн-сервісів є багато переваг перед їх аналогами офлайн. Наприклад, людям не потрібно стояти в черзі з іншими людьми, що, в теперішній, коронавірусний час, є дуже важливою перевагою. Користувач має змогу отримати доступ до потрібної інформації в будь-який момент часу, коли йому цього захочеться і він не прив’язані до графіку роботи міських установ.

У важких економічних умовах з кожним місяцем стає актуальною установка електронної системи оплати проїзду в ліфті, оскільки ця система дозволяє скористатися ліфтом тільки при наявності активного електронного ключа.

На сьогоднішній така система доступу – це єдиний спосіб отримувати гроші з населення напряму, оскільки тут діє принцип: заплатив – скористався ліфтом, не заплатив – йди пішки.

Але є і певні складнощі. Для того, щоб продовжувати користуватися електронними ключами, їх, так чи інакше, потрібно оновлювати. А для того, щоб оновити ключ потрібно йти до керуючого ОСББ. Трапляються ситуації коли до керуючих, на протязі декількох днів, приходять мешканці усіх підконтрольних керуючому будинків щоб оновити свій ключ та стоять у черзі

що є нераціональним. Тому у даній кваліфікаційній роботі бакалавра пропонується розробка веб-додатку, за допомогою якого можна б було віддалено керувати доступом до ліфтів для електронних ключів усіх мешканців.

1 ОГЛЯД ОБЛАСТІ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Системи розумних ліфтів

Система доступу до ліфтів являє собою пристрій, який монтується в кабіні ліфта над пультом керування і підключає керівні кнопки тільки при наявності оплаченої і справної пластикової картки (таблетки).

Принцип дії безконтактної картки (таблетки) заснований на радіосигналі. Це означає, що для коректної роботи такої картки (таблетки) механічний контакт зі зчитувачем картоприймача не потрібен. Пристрій спрацьовує при піднесенні безконтактної картки (таблетки) до передньої панелі картоприймача. У багатьох випадках для спрацьовування картоприймача досить піднести гаманець, де зберігається картка, не виймаючи саму карту.

Пасажир заходить в ліфт, підносить картку або брелок до картоприймача, при цьому система робить запит до сервера через інтернет та отримує у відповідь статус картки (дозволено чи заборонено проїзд), після чого, в разі отримання дозволу, картоприймач замикає попередньо розімкнутий їм ланцюг дротів пульта керування, пасажир прибирає картку або брелок і натискає кнопку потрібного поверху і ліфт починає рухатися попередньо закривши двері.

Кожен з мешканців під'їзду отримує індивідуальну пластикову картку. Поки людина щомісяця платить за обслуговування ліфта, його картка працює. Якщо ж людина не платить, то керуючий ОСББ може натиснути кнопку і картка боржника перестане працювати. Немає необхідності у фізичному контакті з боржником з метою позбавлення його можливості користування ліфтом.

Якщо ж ліфт застрягне, на екрані в диспетчерській висвічується сигнал тривоги. Диспетчер бачить адресу ліфта, який застряг. Він знає навіть прізвище тієї людини, яка застрягла, і номер його телефону. Диспетчер може

зв'язатися з його родичами, які будуть людину розважати і заспокоювати, поки їде аварійна бригада і ліфт стоїть.

При установці на ліфт такої системи, ресурс ліфта збільшується приблизно в 2,5 – 3 рази, оскільки змушує людей економити поїздки, внаслідок чого зменшується пробіг ліфта і економиться електроенергія.

1.2 Огляд існуючих аналогів

На даний момент існує декілька схожих систем керування доступом до ліфтів. Вони знаходяться у Херсоні, Дніпрорудному та Житомирі.

Система у Херсоні встановлюється компанією «Діністор». Система оплати проїзду в ліфті являє собою картоприймач, вмонтований в кабіні ліфта над панеллю наказів. Для цієї мети вирізається електролобзиком прямокутний отвір висотою 85 мм і шириною 95 мм над панеллю наказів (або в будь-якому іншому зручному для монтажу і користування місці), після чого туди вставляється картоприймач і закріплюється двома гвинтами. Виходить, передня панель картоприймача щільно прилягає до купе кабіни (аналогічно пульту наказів). Габарити картоприймача: висота – 125 мм, ширина – 100 мм, глибина – 50 мм.

Мешканцям надаються пластикові картки, на яких записана певна кількість поїздок. Така картка може запам’ятати дату та час останніх 126 поїздок. Але незручність полягає в тому, що для того, щоб відновити або поповнити кількість поїздок, треба звертатися до керуючого ОСББ. Також доступні поїздки можуть закінчитися у самий невідповідний час.

Також у цій системі передбачений екран, на який виводиться кількість поїздок яка залишилася. Коли мешканець хоче подивитися кількість поїздок, з його карти списується одна поїздка, навіть якщо він не скористувався ліфтом, що також нераціонально.

У Дніпрорудному така система була встановлена ще у 2010 р. Фото картоприймача наведено на рис. 1.1.

Рисунок 1.1 – Картоприймач у ліфті

У цій системі є схожий картоприймач, але логіка роботи з картками дещо інакша. Спочатку мешканець повинен придбати пластикову картку яка коштує приблизно 4,5 грн, а потім поповнити її на будь-яку суму. Одна поїздка у ліфті з такою системою коштує 18 копійок, які списуються після прикладання картки. Громадяни пільгових категорій мають на картці певну кількість безкоштовних поїздок, а медичним працівникам, дільничним міліціонерам, співробітникам організацій, які обслуговують будинки, картки видаються безкоштовно.

Незручності у цієї системи такі ж самі, як і у попередньої – необхідність кожен місяць ходити до керуючого ОСББ та поповнювати баланс.

У Житомирі загальна логіка така ж сама, але встановлені картоприймачі дещо старіших моделей. На рис. 1.2 зображений картоприймач в одному із ліфтів Житомира. Приклад картки, яка працює з цим картоприймачем, наведено на рис. 1.3.

Рисунок 1.2 – Картоприймач у ліфті в Житомирі

Рисунок 1.3 – Карта для проїзду у Житомирському ліфті

Система оплати проїзду в ліфті наступна: якщо у мешканця немає заборгованості з комунальних платежів, то він зможе придбати іменну картку (на 100 поїздок) в ЖЕКу, сплативши лише одноразовий внесок – 2,7 грн. Після закінчення ліміту поїздок можна звернутися в ЖЕК для поповнення картки. Однак користуватися надалі ліфтом безкоштовно мешканець зможе тільки в разі відсутності заборгованості.

Підсумовуюче порівняння цих систем наведено у табл. 1.1.

Таблиця 1.1 – Порівняння параметрів систем доступу до ліфтів

Найменування параметру	Система у Житомирі	Система у Херсоні	Система у Дніпрорудному
Противандальний захист	+	+	+
Фіксування дати та часу останніх поїздок	–	+	–
Можливість подивитись кількість поїздок, які залишилися	–	+	+
Наявність безконтактної картки (брелку)	–	+	+
Можливість віддаленого керування статусом картки	–	–	–
Можливість віддаленого поповнення картки	–	–	–

1.3 Типи веб-додатків

Веб-додатки мають ряд переваг перед «настільними» програмами:

– доступ з будь-якого пристрою. Користувачу не потрібно встановлювати веб-додаток у систему. Також користувач не залежить від ОС

(Windows, MacOS, Linux, Android, тощо). Для доступу досить інтернет-з’єднання;

– економія. Так як веб-додатки працюють на всіх платформах, це виключає необхідність розробки програми окремо для Android і iOS;

– відсутність клієнтського ПЗ. Дешевше і простіше установка, обслуговування та модернізація клієнтського інтерфейсу. Оновлення до останньої версії відбувається при черговому завантаженні сторінки;

– масштабованість. З ростом навантаження на систему не треба нарощувати потужність клієнтських місць. Веб-додаток дозволяє обробляти більшу кількість даних, як правило, тільки силами апаратних ресурсів, без переписування коду і зміни архітектури;

– захист від втрати даних. База даних, створена в онлайн-системі, зберігається на сервері в спеціалізованому дата-центрі. Завдяки резервному копіюванню вона надійно захищена від втрат, а зашифрований канал передачі даних гарантує їх конфіденційність і захист від перехоплення.

Загальна архітектура, за якою працюють усі веб-додатки, наведена на рис. 1.4. Веб-додатки можна розділити на кілька типів, в залежності від різних поєднань їх основних складових.

Backend (бекенд або серверна частина програми) працює на віддаленому комп'ютері, який може знаходитися де завгодно. Вона може бути написана на різних мовах програмування: PHP, Python, Ruby, C # та інших. Якщо створювати додаток, використовуючи тільки серверну частину, то в результаті будь-яких переходів між розділами, відправок форм, оновлення даних, сервером буде генеруватися новий HTML-файл і сторінка в браузері перезавантажуватиметься [1].

Frontend (фронтенд або клієнтська частина програми) виконується в браузері користувача. Ця частина написана на мові програмування Javascript. Додаток може складатися тільки з клієнтської частини, якщо не потрібно

зберігати дані користувача довше однієї сесії. Це можуть бути, наприклад, фоторедактори або прості іграшки [1].

Single page application (SPA або односторінковий додаток). Більш цікавий варіант, коли використовуються і бекенд і фронтенд. За допомогою їх взаємодії можна створити додаток, який буде працювати зовсім без перезавантажень сторінки в браузері. Або в спрощеному варіанті, коли переходи між розділами викликають перезавантаження, але будь-які дії в розділі обходяться без них [1].

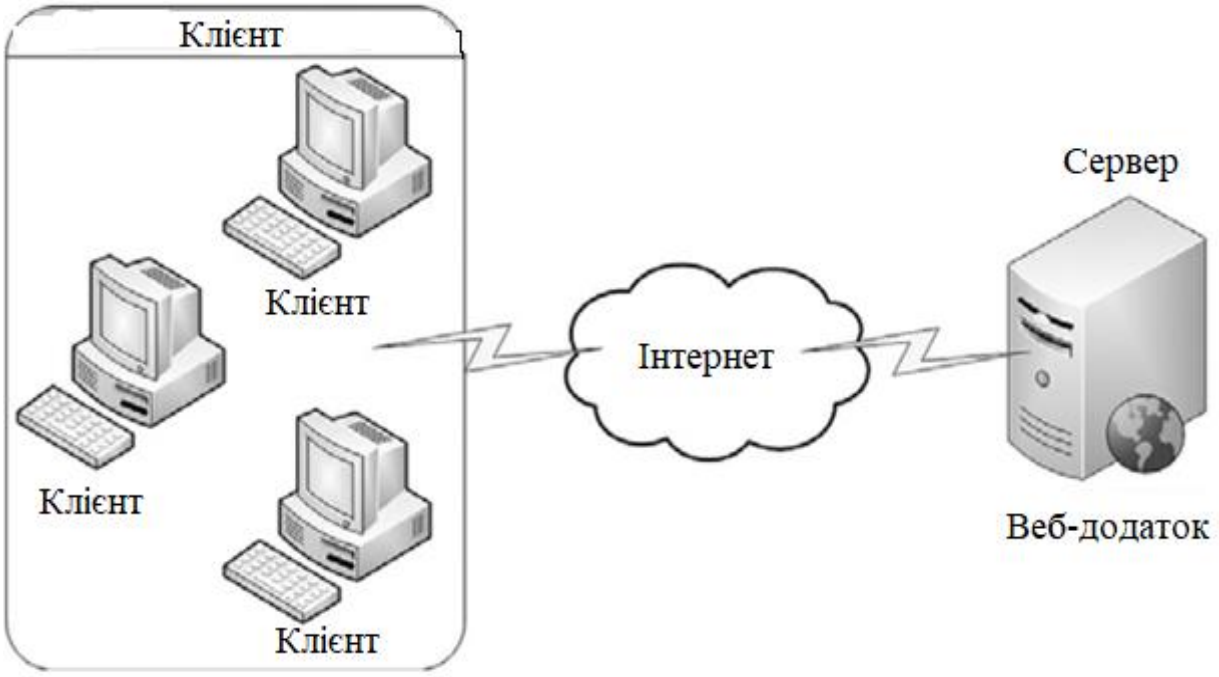

Рисунок 1.4 – Архітектура веб-додатку

Для розроблюваного додатку потрібне зберігання даних користувача у базі даних, тому другий варіант не підходить. Якщо обирати між першим та третім варіантом, то краще надати перевагу третьому, тому що для користувача це буде зручніше. Тому далі буде розглядатися третій варіант.

Клієнтська частина програми – це скрипти, написані на мові програмування JavaScript (JS) і які виконуються в браузері користувача. Раніше вся клієнтська логіка ґрунтувалася на використанні бібліотеки JQuery, яка дозволяє працювати з DOM, анімацією на сторінці і робити AJAX запити.

DOM – це структура HTML-сторінки. Робота з DOM – це пошук, додавання, зміна, переміщення і видалення HTML-тегів [1].

AJAX – це загальна назва для технологій, які дозволяють робити асинхронні (без перезавантаження сторінки) запити до сервера і обмінюватися даними. Так як клієнтська і серверна частини веб-додатку написані на різних мовах програмування, то для обміну інформацією необхідно перетворювати структури даних (наприклад, списки і словники), в яких вона зберігається, в JSON-формат.

JSON – це універсальний формат для обміну даними між клієнтом і сервером. Він являє собою простий рядок, який може бути використаний в будь-якій мові програмування [1].

За допомогою маніпуляцій з DOM можна повністю керувати вмістом сторінок. За допомогою AJAX можна обмінюватися даними між клієнтом і сервером. З цими технологіями вже можна створити SPA. Але при створенні складного додатка код фронтенда, заснованого на JQuery, швидко стає заплутаним і важко підтримуваним.

На зміну JQuery прийшли Javascript-фреймворки: Angular, React, Vue і інші. У них різна філософія і синтаксис, але всі вони дозволяють з набагато більшою зручністю управляти даними сторінки, мають шаблонізатор та інструменти для створення навігації між сторінками.

Фреймворк – це каркас програмної системи (або підсистеми). Може включати: допоміжні програми, бібліотеки коду, що полегшують розробку і об'єднання різних компонентів великого програмного проекту [2].

HTML-шаблон – це «розумна» HTML-сторінка, в якій замість конкретних значень використовуються змінні і доступні різні оператори: if, цикл for і інші. Процес отримання HTML-сторінки з шаблону, коли підставляються змінні і застосовуються оператори, називається рендерингом шаблону [1].

Отримана в результаті рендерингу сторінка показується користувачеві. Перехід в інший розділ в SPA це застосування іншого шаблону. Якщо

необхідно використовувати в шаблоні інші дані, то вони запитуються у сервера. Всі відправки форм з даними це AJAX запити на сервер.

Зараз існує три основних фреймворка для розробки веб-додатків, які закріпилися на ринку та є найбільш популярними. Це React, Angular та Vue. Рівень популярності цих фреймворків наведений на рис. 1.5.

Vue – це молодий фреймворк зі зростаючою аудиторією. Найпростіший для вивчення в трійці, можна почати роботу «з коробки», при цьому досить потужний для професійних розробників. Vue не має такої кількості вбудованих функцій, як Angular, але їх більше, ніж у React. Популярний вибір серед новачків, є докладна документація російською мовою.

React пропонує просте і функціональне створення компонентів, а також пропагує їх використання для підтримки елегантного коду. Фреймворк дуже популярний, особливо в різних стартапах. Маючи великий вибір легкодоступних плагінів і розширень, можна розробити практично будь-який тип веб-додатку.

Angular популярний в ентерпрайз-середовищі, де має широку публіку. Код трохи перенасичений і складний у порівнянні з іншими фреймворками. Також варто сказати, що розробка у Angular виконується на мові програмування TypeScript, що додає складності, але має свої плюси.

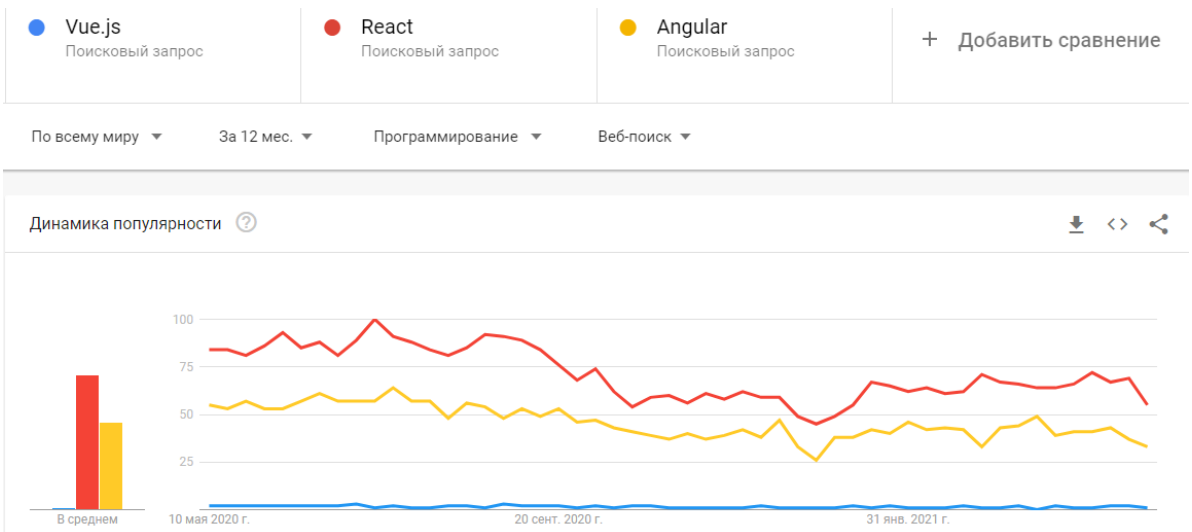

Рисунок 1.5 – Динаміка популярності фреймворків

Далі будуть розглядатися React та Vue, тому що у розроблюваному веб-додатку не планується багато функціональності, тому обирати фреймворк, який більш підходить для ентерпрайз рішень, немає сенсу. Також React та Vue не змушують використовувати TypeScript, можна використовувати звичайний JavaScript, який є більш популярним на ринку. Це означає, що більше людей зможуть здійснювати підтримку розроблюваного веб-додатку у майбутньому.

Як React, так і Vue – виключно швидкі, тому швидкість навряд чи буде вирішальним фактором при виборі між ними [3]. У табл. 1.2 наведено порівняння деяких параметрів фреймворків React і Vue.

Таблиця 1.2 – Порівняння фреймворків

Найменування параметра	React	Vue
Зірки на GitHub	168 тис.	183 тис.
Кількість коммітів	14,1 тис.	3.1 тис.
Повнота екосистеми	Має найбільшу кількість доступних бібліотек та модулів серед порівнюваних фреймворків	Також має велику кількість бібліотек, але меньше ніж у React тому, що фреймворк порівняно молодий
Кількість контриб'юторів	1,5 тис.	400
Кількість сайтів з використанням фреймворку	1067 тис.	851 тис.

Мобільна розробка. React має аналог для розробки додатків для iOS та Android – React Native. React Native – безперечний король міжплатформенної розробки. Він дозволяє повторно використовувати до 99% коду JavaScript між Android та iOS з React-подібними компонентами.

Незважаючи на те, що Vue відстає від React, він пропонує кілька рішень для мобільної розробки. По-перше, у вас є NativeScript, який дозволяє писати

програми Vue та компілювати їх у програми для iOS / Android. По-друге, є Vue Native. Фреймворк поєднує в собі переваги екосистем Vue та React Native, декларативний рендер, двосторонній зв'язок та набір базових компонентів для створення крос-платформного додатка.

Висновки**:** Обидві бібліотеки задовольняють вимоги сучасних веб-додатків. React – лідер за підтримки корпорацій та величезної спільноти з відкритим кодом. Бібліотечні масштаби краще дозволяють створювати складні корпоративні програми.

Величезна екосистема React означає, що розробник можете знайти рішення практично для будь-якої проблеми, яка може виникнути у проекті.

Нарешті, React Native пропонує найкраще крос-платформне рішення на ринку.

Vue – молодий проект, який швидко розвивається. Його легко вивчити та він має більш традиційний синтаксис, що дозволяє поступово мігрувати існуючі проекти до Vue.

Vue пропонує більше нестандартних інструментів за підтримки основної команди.

Зважаючи на перелічену інформацію вище, буде розумним обрати React тому, що він є більш популярним та має більш розвинуту екосистему. Це означає, що існує більше готових рішень для різного типу проблем.

1.4 Вимоги до доступності сучасних веб-додатків

Доступність означає, що сайт розроблений таким чином, що їм можуть користуватися люди з обмеженими можливостями. Під використанням розуміється сприйняття інформації, навігація по інтерфейсу і взаємодія з ним.

Основні характеристики доступності:

– контраст кольорів;

– розпізнавання голосу;

– підтримка скрінрідерів – озвучування сайту для сліпих і слабозорих людей;

– логічна і проста навігація;

– великі елементи управління;

– можливість налаштувати контент – наприклад, збільшити розмір шрифту [4].

Про доступність сайтів та додатків кажуть в контексті використання їх людьми зі слуховими, візуальними, мовними, когнітивними, неврологічними, фізичними проблемами. Однак доступність інтернету приносить користь і людям без інвалідності:

– користувачам смартфонів, розумних телевізорів і годинників, а також інших пристроїв з невеликими екранами або різними режимами вводу;

– літнім людям, у яких погіршуються фізичні можливості;

– людям з тимчасовими проблемами – наприклад, зламаною рукою або втраченими окулярами;

– користувачам з ситуаційними обмеженнями – наприклад, яскравим сонячним світлом або знаходженням в середовищі, де вони не можуть слухати аудіо;

– людям, які використовують повільне підключення до інтернету.

Багато пунктів, з перерахованих вище, можна протестувати за допомогою Lighthouse.

Lighthouse – це автоматизований інструмент для підвищення якості веб-сторінок із відкритим кодом. Його можна запустити на будь-якій веб-сторінці, загальнодоступній або яка вимагає автентифікації. Він перевіряє ефективність, доступність, прогресивні веб-програми, SEO та багато іншого.

Вигляд Lighthouse наведено на рис. 1.6, а вигляд одержуваного репорту наведено на рис. 1.7.

Існує декілька способів його використання, але найпростішим є варіант, де Lighthouse запускається прямо з інструментів для розробника у браузері Google Chrome.

Щоб протестувати будь-яку веб-сторінку необхідно:

– зайти на потрібну веб-сторінку у браузері Google Chrome;

– відкрити інструменти розробника, натиснувши кнопку F12, Ctrl + Shift + I або натиснути праву кнопку миші та обрати опцію «Перевірити»;

– перейти на вкладку Lighthouse та обрати потрібні опції, які потрібно протестувати;

– натиснути «Generate report».

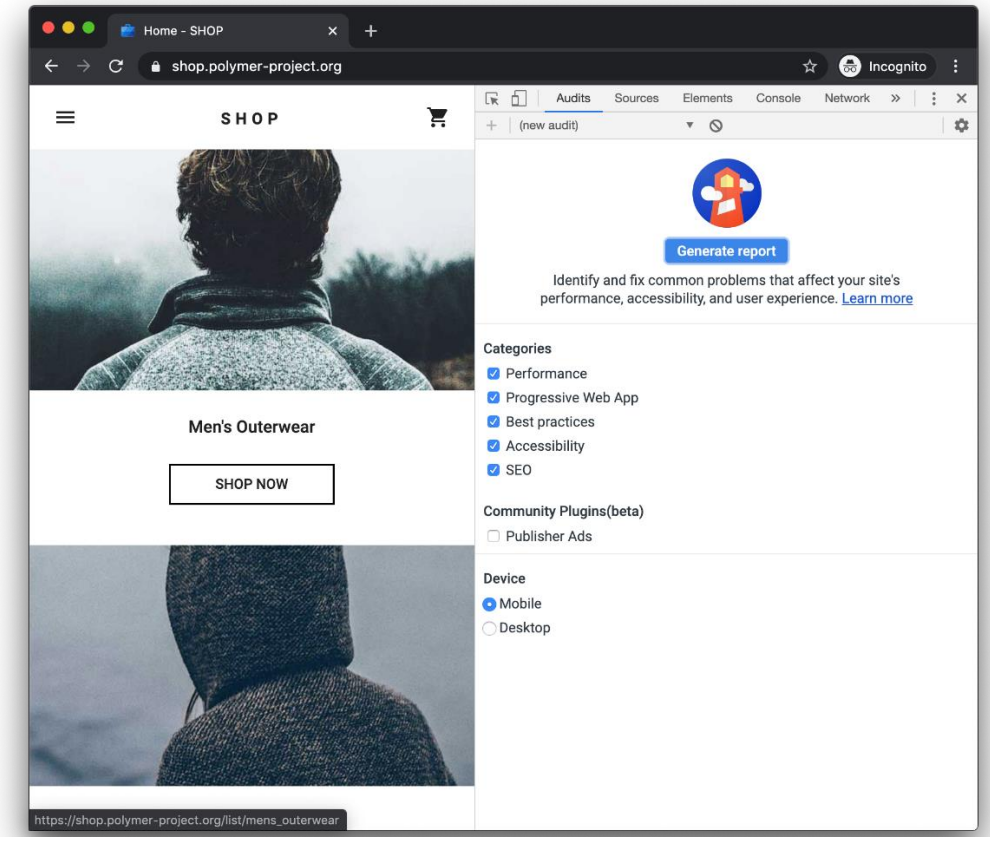

Рисунок 1.6 – Lighthouse у інструментах розробника

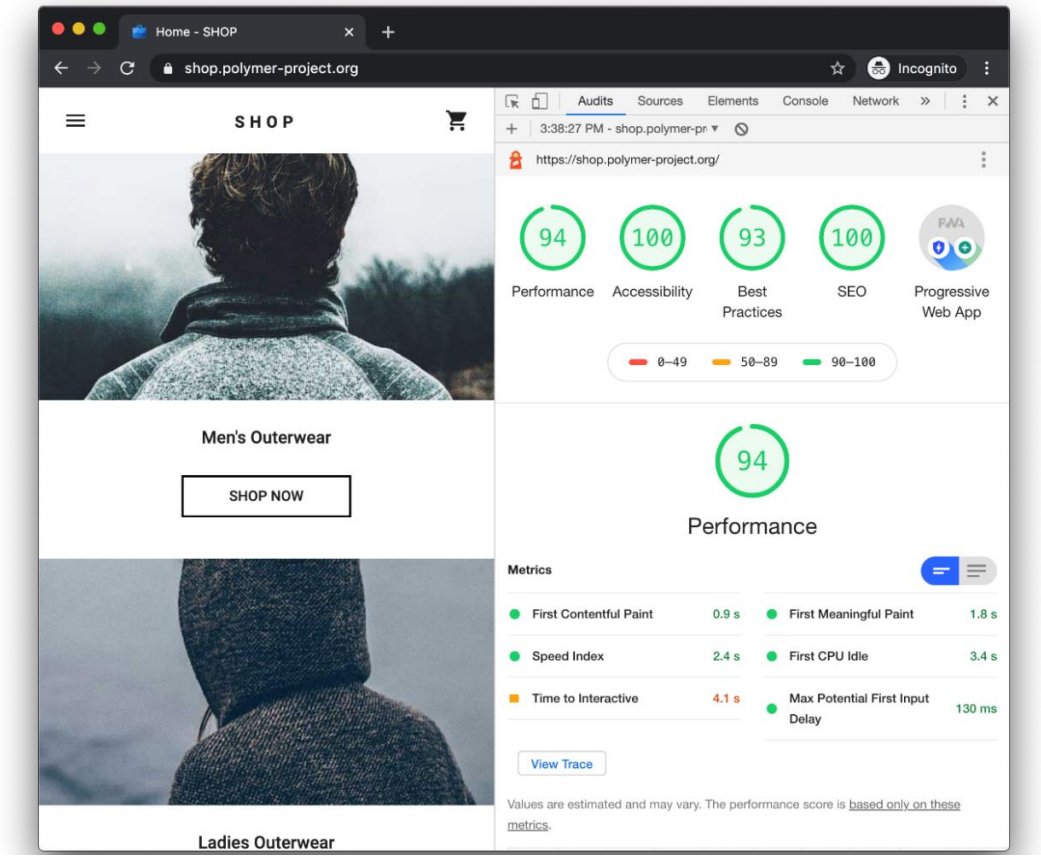

Рисунок 1.7 – Приклад результату тестів у Lighthouse

Також, крім докладного звіту про властивості веб-сторінки, Lighthouse надає корисні поради, які допоможуть значно поліпшити стан вашого сайту.

1.5 Постановка задач

Відповідно до п. 1.2, однією з основних проблем усіх існуючих систем керування доступом до ліфтів є те, що немає змогу поповнити картку віддалено для звичайних мешканців та немає змоги віддалено керувати картками та їх статусом для керуючих компаній. Це є незручним для мешканців, тому що кожен місяць потрібно ходити до керуючої компанії та наново поповняти свої картки та ускладнює роботу керуючим тому, що раз на місяць усі мешканці підконтрольних будинків приходять для поповнення своїх карток.

Тому метою кваліфікаційної роботи бакалавра є створення веб-додатку для керування доступом до ліфтів у багатоповерхових будинках з можливістю віддаленого керування доступом.

Веб-додаток повинен мати таку функціональність:

– два типи користувачів – звичайні та керуючі ОСББ або адміністратори, з розмежуванням прав доступу;

– для звичайного користувача повинна бути сторінка, де він зможе переглянути інформацію про його статус оплати та про прив'язані електронні ключі;

– керуючий ОСББ або адміністратор повинен бути здатний обрати будинок зі списку або на онлайн-мапі, обрати потрібний під'їзд у цьому будинку та переглянути інформацію щодо мешканців цього під'їзду;

– керуючий ОСББ або адміністратор повинен бути здатний керувати ключами доступу для кожного мешканця окремо, активувати або деактивувати існуючі ключі, додавати нові або видаляти старі ключі;

– повинна бути сторінка логіну з використанням Google reCAPTCHA та двохфакторної аутентифікації;

– веб-додаток повинен бути доступний для користувачів з обмеженими можливостями;

– веб-додаток повинен бути працездатним на пристроях з маленькими екранами.

2 АНАЛІЗ СЕРВЕРНОЇ ЧАСТИНИ

2.1 Протокол взаємодії з сервером

Веб-браузери взаємодіють з веб-серверами за допомогою протоколу передачі гіпертексту (HTTP). Коли користувач натискає на посилання на сторінці, заповнює форму або робить пошук, браузер відправляє на сервер HTTP-запит [5].

Цей запит включає:

– шлях (URL), який визначає цільової сервер і ресурс (наприклад, HTML-файл, конкретна точка даних на сервері або інструмент, що запускається);

– метод, який визначає необхідну дію (наприклад, отримати файл, зберегти або оновити будь-які дані);

– додаткова інформація може бути закодована в запиті (наприклад, дані HTML-форми). Інформація може бути закодована як URL-параметри, POST дані або кукі-файли клієнтської частини.

Веб-сервери очікують повідомлень із запитами від клієнтів, обробляють їх, коли вони приходять і відповідають веб-браузеру через повідомлення з HTTP-відповіддю. Відповідь містить Код статусу HTTP-відповіді, який показує, чи був запит успішним (наприклад, «200 OK» означає успіх, «404 Not Found» якщо ресурс не може бути знайдений, «403 Forbidden», якщо користувач не має права переглядати ресурс, і т.д.). Тіло успішної відповіді на запит GET буде містити запитуваний ресурс [5].

Абревіатура HTTP розшифровується як HyperText Transfer Protocol, «протокол передачі гіпертексту». Відповідно до специфікації OSI, HTTP є протоколом прикладного (верхнього, 7-го) рівня [6].

HTTPS (розшифровується як HyperText Transfer Protocol Secure – безпечний протокол передачі гіпертексту) - це розширення протоколу HTTP, що підтримує шифрування за допомогою криптографічних протоколів SSL і TLS [6].

Відмінності HTTP от HTTPS:

– HTTPS не є окремим протоколом передачі даних, а являє собою розширення протоколу HTTP з надбудовою шифрування;

– дані, які передаються по протоколу HTTP, не захищені, HTTPS забезпечує конфіденційність інформації шляхом її шифрування;

– HTTP використовує порт 80, HTTPS – порт 443.

Використання HTTPS потрібно там, де необхідно забезпечити безпечну передачу даних. Наприклад на сайтах, де вводиться і передається конфіденційна інформація (особисті дані користувачів, деталі доступу, реквізити платіжних карт) – на будь-яких сайтах з авторизацією, взаємодією з платіжними системами, поштовими сервісами. Шифрування таких даних дозволить запобігти їх отриманню і використанню третіми особами [6].

Виходячи з вищевикладеної інформації, розроблюваному додатку буде потрібно взаємодіяти з сервером, використовуючи HTTPS протокол.

2.2 JSON як формат даних при взаємодії з сервером

За останні 15 років JSON став широко розповсюдженим в Інтернеті. На сьогоднішній день це формат, який вибирають майже для кожного загальнодоступного веб-сервісу, і він часто використовується також для приватних веб-сервісів [7].

Популярність JSON також призвела до вбудованої підтримки JSON у багатьох базах даних. Реляційні бази даних, такі як PostgreSQL та MySQL, тепер мають вбудовану підтримку для зберігання та запиту даних JSON.

Приклад даних у форматі JSON наведено на рис. 2.1.

```
{
  “firstName”: “Іван”,
  “lastName”: “Іванов”,
  “loginCount”: 4,
  “isWriter”: true,
  “worksWith”: [“Ім'я компанії”, “Ім'я компанії 2],
  “pets”: [
    {
      “name”: “Лілі”,
      “type”: “Єнот”
    }
  ]
}
```

Рисунок 2.1 – Дані у форматі JSON

Раніше сайти працювали так: коли користувач натискав на посилання або кнопку в браузері, на сервер надсилався запит, сервер готував необхідну інформацію у вигляді HTML, а браузер відображав цей HTML як нову сторінку. Цей шаблон був неефективним, що вимагало від браузера повторного відтворення всього на сторінці, навіть якщо змінився лише розділ сторінки [7].

Оскільки повне перезавантаження сторінок коштувало дорого, з’явилися новіші технології. Тим часом можливість надсилання веб-запитів у фоновому режимі під час показу сторінки виявилася життєздатним підходом до поступового завантаження даних для відображення. Замість того, щоб перезавантажувати весь вміст сторінки, натискання кнопки оновлення викликає веб-запит, який завантажується у фоновому режимі. Коли вміст завантажувався, даними можна було маніпулювати, зберігати та відображати на сторінці за допомогою мови JavaScript.

Спочатку ці дані передавались у форматі XML за допомогою протоколу обміну повідомленнями під назвою SOAP (Simple Object Access Protocol). Але XML був багатослівним і важким для управління в JavaScript. JavaScript уже мав об'єкти, які є способом вираження даних у мові, тому Дуглас Крокфорд

взяв підмножину цього виразу як специфікацію для нового формату обміну даними і назвав його JSON. JSON було набагато легше читати та він швидше аналізувався браузерами [7].

Як зазначалося вище, основною альтернативою JSON є XML. Однак XML стає все рідшим в нових системах, і легко зрозуміти, чому. На рис. 2.2 наведена версія даних, яка зображена на рис. 2.1, цього разу в форматі XML:

```
<?xml version="1.0"?>
<person>
  <first_name>“Іван”</first_name>
  <last_name>“Іванов”</last_name>
  <login_count>4</login_count>
  <is_writer>true</is_writer>
  <works_with_entities>
    <works_with>Ім'я компанії</works_with>
    <works_with>Ім'я компанії 2</works_with>
  </works_with_entities>
  <pets>
    <pet>
      <name>Лілі</name>
      <type>Єнот</type>
    </pet>
  </pets>
</person>
```

Рисунок 2.2 – Дані у форматі XML

Окрім того, що XML є більш багатослівним (у цьому випадку рівно вдвічі), XML також вносить певну неоднозначність при синтаксичному аналізі структури, зручної для JavaScript. Перетворення XML в об'єкт JavaScript може зайняти від десятків до сотень рядків коду і, зрештою, вимагає налаштування на основі аналізу конкретного об'єкта. Перетворення JSON в об’єкт JavaScript займає один рядок коду і не вимагає жодних попередніх знань про об’єкт, що аналізується.

2.3 Запити до серверу

Нижче наведені структури запитів до серверу, які будуть використовуватися у розроблюваному додатку.

2.3.1 Запит для логіну

Запит, на який відправляються дані, уведені користувачем у формі авторизації.

Маршрут запиту: /AuthForm. Тип запиту: POST. Тіло запиту наведено на рис. 2.3, де username – це логін користувача, а password – пароль користувача. Тіло відповіді наведено на рис. 2.4, де status – число, що відповідає статусу відповідності введених даних авторизації та uid – id користувача у базі даних. Можливі статуси наведені у табл. 2.1.

Таблиця 2.1 – Легенда поля status

Код статусу	Пояснення
0	користувач з таким логіном відсутній
1	користувач з таким логіном існує, але пароль введено не вірно
2	користувач існує, пароль вірний, але користувачу заблоковано доступ в систему з якихось причин (причини з’ясовуються у керуючої компанії)
3	користувач існує, пароль вірний, користувачу відправлено перевірочний код двофакторної авторизації

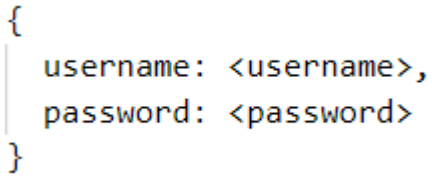

Рисунок 2.3 – Тіло запиту для логіну

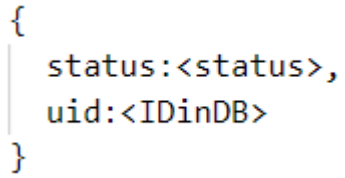

Рисунок 2.4 – Тіло відповіді на запит для логіну

2.3.2 Запит для коду двофакторної авторизації

Запит, на який відправляється код двофакторної авторизації для підтвердження логіну.

Маршрут запиту: /AuthCode. Тип запиту: POST. Тіло запиту наведено на рис. 2.5, де uid – id користувача у базі даних, code – отриманий код підтвердження. Тіло відповіді наведено на рис. 2.6, де uid – id користувача у базі даних, acc_lvl – рівень доступу користувача, flats – масив з квартирами, якими може керувати користувач (якщо він є адміністратором або керуючим ОСББ).

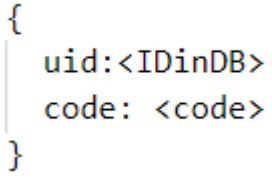

Рисунок 2.5 – Тіло запиту для коду двофакторної авторизації

```
{
  uid: <IDinDB>,
  is_correct: <true/false>,
  acc_lvl: <acc_lvl>,
  flats:[{
    flat_id: <flat_id>,
    city: <city>,
    street: <street>,
    house_numb: <house>,
    house_latitude: <house_latitude>,
    house_longitude: <house_longitude>,
    entrance: <entrance>,
    flat_numb: <flat_numb>,
    is_access: <true/false>,
    man:[{
        surname: <surname>,
        name: <name>,
        fname: <fname>,
        card_id: <card_id>
    }]
  }]
}
```

Рисунок 2.6 – Тіло відповіді на запит для коду двофакторної авторизації

2.3.3 Запит для відновлення паролю

Якщо користувач забув свій пароль, він може отримати новий в повідомленні, використовуючи свій логін.

Маршрут запиту: /AuthRestore. Тип запиту: POST. Тіло запиту наведено на рис. 2.7, де username – логін користувача. Тіло відповіді наведено на рис. 2.8, де status – статус, який показує, чи було відправлене смс з новим паролем.

```
{
  username: <username>
}
```

Рисунок 2.7 – Тіло запиту для відновлення паролю

```
{
    status:<true/false>
}
```

Рисунок 2.8 – Тіло відповіді на запит для відновлення паролю

2.3.4 Запит для зміни паролю

Запит, який використовується для зміни паролю.

Маршрут запиту: /ChngPass. Тип запиту: POST. Тіло запиту наведено на рис. 2.9, де uid – id користувача у базі даних, old_pass – старий пароль, new_pass – новий пароль. Тіло відповіді наведено на рис. 2.10, де status – статус, який показує була зміна паролю успішною чи ні.

```
{
  uid:<IDinDB>,
  old_pass: <old_pass>,
  new_pass: <old_pass>,
}
```

Рисунок 2.9 – Тіло запиту для зміни паролю

```
{
    status:<true/false>
}
```

Рисунок 2.10 – Тіло відповіді на запит для зміни паролю

2.3.5 Запит для зміни даних про квартиру

Запит, який використовується, коли керуючому потрібно змінити якісь дані по певній квартирі.

Маршрут запиту: /ChngFlat. Тип запиту: PATCH. На рис. 2.11 наведено тіло запиту, де uid – id користувача у базі даних, flat_id – id квартири у базі

даних, man – масив людей які проживають у цій квартирі, is_access – чи є доступ до користування ліфтами. Тіло відповіді наведено на рис. 2.12, де status – статус, який показує, чи була зміна даних по квартирі успішною.

```
{
  uid:<IDinDB>,
  flat_id:<flat_id>,
  is_access:<true/false>,
  man:[{
    surname:<surname>,
    name:<name>,
    fname:<fname>,
    card_id:<card_id>
  }]
}
```

Рисунок 2.11 – Тіло запиту на зміну даних про квартиру

```
{
    status:<true/false>
}
```

Рисунок 2.12 – Тіло відповіді на запит на зміну даних про квартиру

3 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ

3.1 Вибір UI-бібліотеки

UI-kit (user interface kit) – це готовий набір елементів, призначених для користувацького інтерфейсу. Ще їх називають фреймворком для дизайнера, UX-kit (від user experience). Готовий кіт, найчастіше, є набором графіки в шарах, але зустрічаються і зверстані елементи.

Вони потрібні, щоб швидко зробити проект в єдиній стилістиці. На рис. 3.1 та 3.2 наведені приклади UI-kit [8].

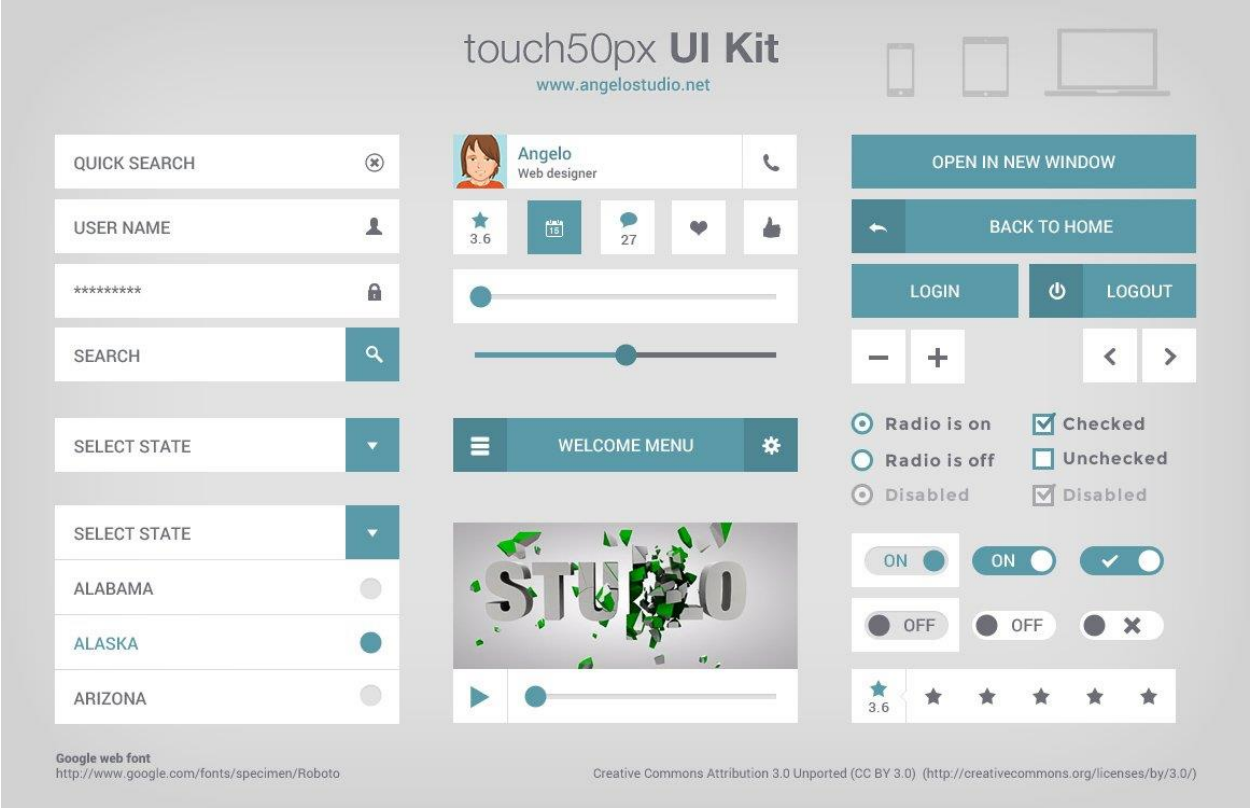

Рисунок 3.1 – Приклад UI-kit

Рисунок 3.2 – Приклад UI-kit

Використовуючи UI-kit, ми заощаджуємо години і дні роботи. UI-kit – готова дизайн концепція з усіма елементами, залишилося тільки розташувати компоненти майбутнього інтерфейсу. Цей підхід кращий за використання шаблонів, тому що набагато гнучкіший і швидший.

Також можна обрати якийсь існуючий UI-kit тому, що в нас нема власного бренду і нам не потрібна унікальність та впізнаваність. Натомість ми будемо використовувати рішення, які перевірені часом та багатьма іншими продуктами.

Зі стрімким зростанням React та його спільноти користувачів, зростає спектр бібліотек підтримки, особливо бібліотеки компонентів інтерфейсу користувача. Ці бібліотеки заощаджують нам багато часу та сил при створенні сучасних програм React. Вони надають багато готових компонентів, таких як набори піктограм, кнопки, підбір часу та дати, форми, календар, меню, пагінація, картки та багато іншого.

Зараз існує багато таких бібліотек, тому ми розглянемо лише декілька найпопулярніших.

Ant Design – це набір високоякісних компонентів React, готових для створення веб-додатків. Він повністю написаний на TypeScript, щоб зробити процес розробки простішим та зручнішим, мінімізуючи помилки. На рис. 3.3 наведений приклад компонентів кнопок у стилі Ant Design.

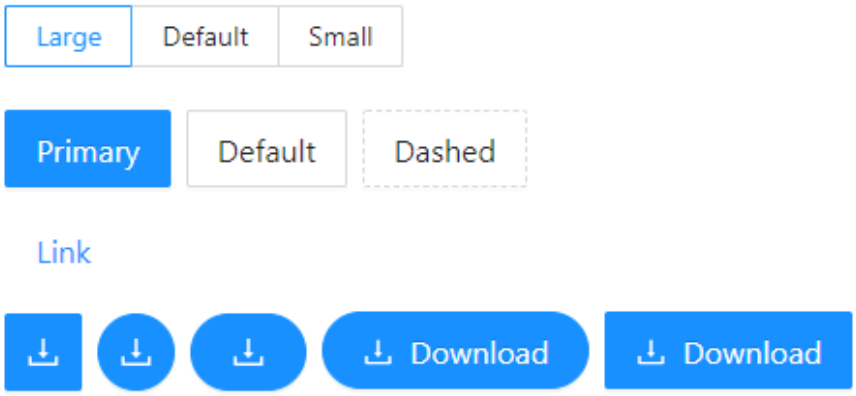

Рисунок 3.3 – Приклади компонентів кнопок у стилі Ant Design

Material UI – це набір компонентів, побудований на основі принципів матеріал дизайну компанії Google. Компоненти використовують лише ті стилі, які їм потрібно відобразити, що може призвести до підвищення продуктивності вашої програми. Хоча кількість зірок на Github трохи менша, ніж у Ant Design, кількість завантажень для Material UI набагато вища. На рис. 3.4 наведений приклад компонентів кнопок у стилі Material UI.

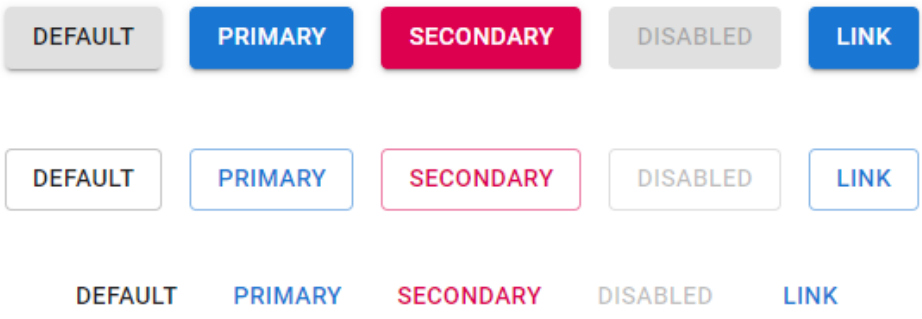

Рисунок 3.4 – Приклади компонентів кнопок у стилі Material UI

React Bootstrap. Будучи однією з найстаріших бібліотек React, React Bootstrap був оновлений і вирiс поряд з React. Кожен його компонент реалізований з урахуванням доступності. Результатом є набір доступних за замовчуванням компонентів, порівняно з тим, що можливо із звичайного Bootstrap. На рис. 3.5 наведений приклад компонентів кнопок у стилі React Bootstrap.

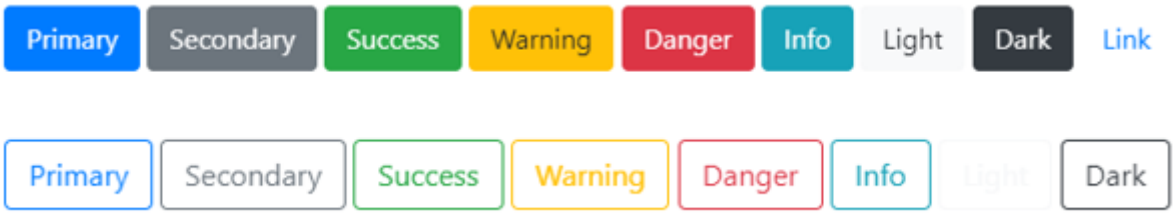

Рисунок 3.5 – Приклад компонентів кнопок у стилі React Bootstrap

Semantic UI React. Поставляється з величезним переліком попередньо побудованих компонентів, розроблених спеціально для розуміння та створення семантично зручного коду. На рис. 3.6 наведений приклад компонентів кнопок у стилі Semantic UI React.

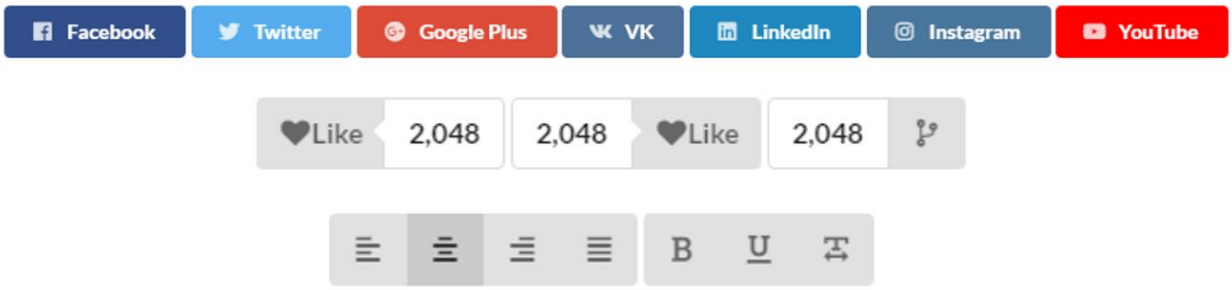

Рисунок 3.6 – Приклад компонентів кнопок у стилі Semantic UI React

Semantic UI React забезпечує спосіб об'єднання компонентів React за допомогою «as» параметру (рис. 3.7). Це дозволяє компонувати особливості компонентів без додавання зайвих вкладених компонентів [9].

```
<>
  {/*  Кожен компонент Semantic UI React має значення за замовчуванням для параметра `as` */}
  {/* Виведе: <button class='ui button' /> */}
  <Button />
  {/* Використовує інший тег: <a class='ui button' /> */}
  <Button as='a' />
</>
```

Рисунок 3.7 – Приклад використання «as» параметру

Було вирішено обрати Material UI тому, що Material UI слідує Material Design від Google. Через це нам надається бібліотека компонентів, яка, ймовірно, буде виглядати знайомою базі користувачів. Це дозволяє швидко розпочати роботу з бібліотекою, витрачаючи мінімум часу на стилізацію компонентів. Завдяки Material Design можна отримати повну систему дизайну, яка описує цілі багатьох доступних компонентів та приклади того, як їх слід використовувати.

Також з цією бібліотекою не тільки надаються компоненти, які вже відповідають Material Design, але й доступ до іконок, також у Material Design. Це означає однаковий набір піктограм на вибір, який відповідає використовуваній системі дизайну.

Бібліотека регулярно оновлюється виправленнями помилок та додатковими функціями. Всі їх оновлення добре задокументовані, а історія оновлень доступна на їх сайті.

Ще одним цікавим аспектом роботи з Material UI є "лабораторія". Лабораторія – це окремий пакет, де команда Material UI може випустити нові компоненти для використання та тестування розробниками, не порушуючи основний пакет. Це дозволяє розробникам використовувати нові компоненти та забезпечувати зворотний зв’язок з розробниками Material UI.

3.2 Розробка структури веб-додатку

Структура веб-додатку – спосіб компоновки, розташування, а значить, і подачі інформації, який, завдяки використанню спеціальних інструментів, що спрощують її сприйняття, дозволяє за короткий проміжок часу максимально докладно розповісти про надані веб-додатком послуги. Структура веб-додатку вибудовується логічно, залежно від цінності інформації.

Важливою частиною структури веб-додатку є глобальна система навігації. Це система програмних і візуальних засобів, за допомогою яких користувач може переміщатися по веб-додатку і знаходити потрібну йому інформацію. За визначенням, глобальна система навігації повинна бути присутня на кожній сторінці веб-додатку. Часто вона реалізується у вигляді панелі навігації, що розташовується у верхній частині сторінки. Такі глобальні системи навігації дають можливість прямого доступу до головних розділів і функцій незалежно від того, в якій частині ієрархічного дерева знаходиться користувач в даний момент.

Оскільки панелі глобальної навігації часто виявляються єдиним послідовно реалізованим елементом на сайті, вони мають величезний вплив на юзабіліті. Вигляд глобальної навігації наведений на рис. 3.8. Глобальна система навігації проектованого веб-додатку містить три пункти:

– особистий кабінет (сторінка, де користувач може переглянути інформацію по прив'язаним електронним ключам та статусу його оплати за користування ліфтами);

– опцію виходу з облікового запису;

– мої будинки (доступна тільки для керуючих ОСББ. Сторінка, де користувач може переглянути інформацію по підконтрольним йому будинкам, під'їздам, тощо).

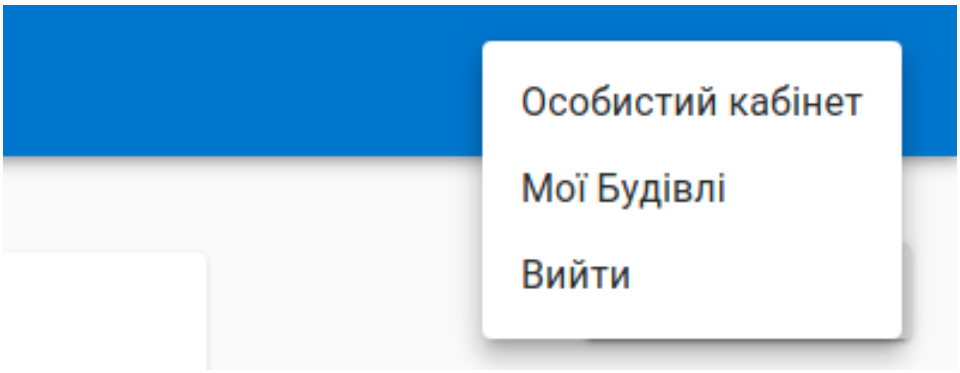

Рисунок 3.8 – Випадаюче головне меню

Також на панелі навігації присутній селектор перемикання мови (наведений на рис. 3.9) та назва веб-додатку з іконкою (рис. 3.10), що водночас є посиланням на сторінку «Особистий кабінет» для звичайного користувача, та посилання на сторінку «Мої будинки» для керуючих ОСББ.

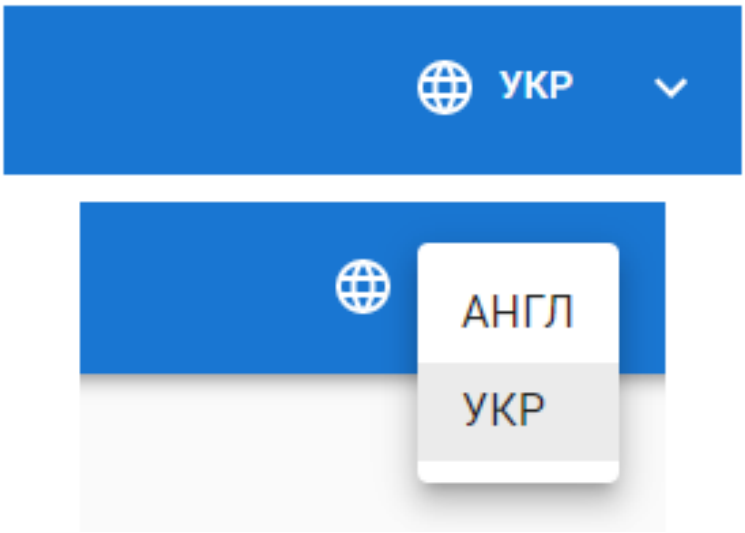

Рисунок 3.9 – Селектор перемикання мови

Рисунок 3.10 – Назва веб-додатку та посилання на відповідні сторінки

3.3 Перевірка даних у веб-додатку

Веб-додаток містить форму входу, призначену для введення інформації користувачем. Одним з найважливіших етапів отримання даних від користувача є їх перевірка, необхідна для того, щоб виключити зайву, неінформативну або суперечливу інформацію.

У проектованому додатку критеріями перевірки є наявність даних в полях, обов'язкових для заповнення, відповідність введених даних необхідному формату і відповідність довжини введених даних необхідній. Перевірка даних, що вводяться, здійснюється на стороні клієнта, тому що в цьому випадку користувач може відразу ж, ще до відправки даних на сервер, отримати повідомлення про наявні помилки.

Клієнтська перевірка даних, що вводяться, здійснюється за допомогою функцій, написаних на мові JavaScript. Перевірка даних, що вводяться, здійснюється при натисканні користувачем кнопки введення або після того, як поле втрачає фокус.

При наявності помилок в формі введення, користувачу виводиться повідомлення про помилку яскраво-червоного кольору (наведено на рис. 3.11, 3.12 та 3.13). Повідомлення підказує користувачеві, чому сталася помилка.

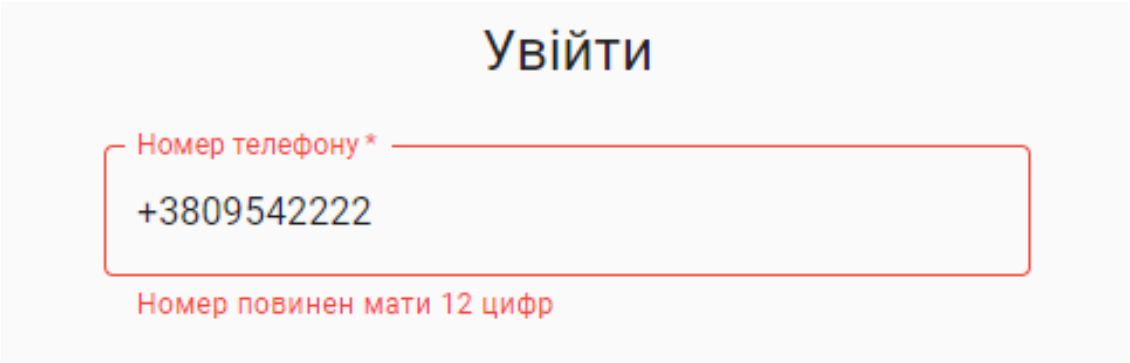

Рисунок 3.11 – Повідомлення про помилку валідації номеру телефону

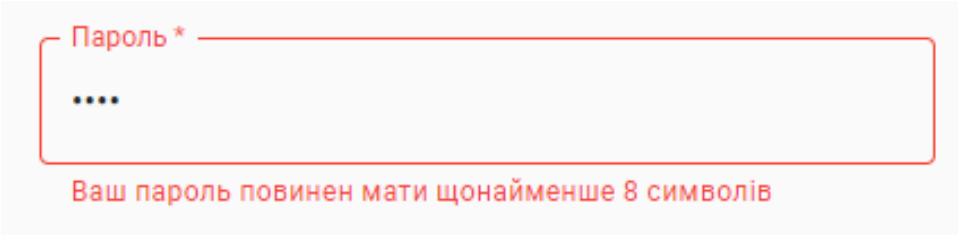

Рисунок 3.12 – Повідомлення про помилку валідації паролю

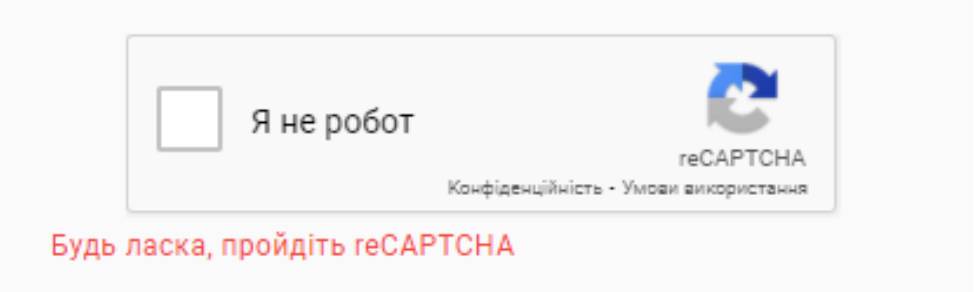

Рисунок 3.13 – Повідомлення про помилку, якщо користувач не пройшов Google reCAPTCHA

3.4 Тестування ергономічності інтерфейсу додатку

Тестуванням програмного забезпечення є процес виявлення наявності дефектів в системі. Дефект може бути привнесений на стадії розробки або супроводу, в результаті чого з'являється одна або більше помилок [10].

Тестування має велике значення в силу того, що цей процес суттєво сприяє тому, щоб конкретне програмне забезпечення робило саме те, що очікують від нього проектувальники і відповідало вимогам технічного завдання. Це вносить істотний внесок в захист користувачів від відмов програмного забезпечення, які можуть спричинити за собою втрати часу, матеріальних цінностей або замовників.

Найбільш раціонально проводити тестування на різних стадіях проектування системи. Для тестування проектованого веб-додатку були взяті основи методу евристичного тестування, описані Якобом Нільсеном.

Евристична оцінка – це один з методів науки юзабіліті, який служить для виявлення проблем з юзабіліті в дизайні для користувача інтерфейсу,

виправлення яких є частиною багатоетапного процесу дизайну. Евристична оцінка проводиться невеликою групою людей, які оцінюють інтерфейс і судять про його правильність, спираючись на певні загальновизнані принципи юзабіліті (які називаються "евристиками") [11].

Робота веб-додатку перевіряється групою користувачів, яка складається з п'яти осіб. Під час проведення тестування кожен учасник виконує свою роботу індивідуально. Тільки після того, як всі учасники закінчать виконання роботи, їм дозволяється поговорити один з одним і об'єднати отримані відомості.

Під час оцінки тестувальник кілька разів розглядає різні діалогові елементи інтерфейсу, порівнюючи їх зі списком принципів юзабіліті (евристик). Ці евристики є загальними правилами, які описують загальні властивості всіх інтерфейсів, які вважаються зручними у використанні. На додаток до цього списку загальних евристик, тестувальник може створити свій список на основі виявлених проблем з інтерфейсом. Список евристик для проектованого веб-додатку наведений нижче.

Для сторінки «Особистий кабінет» та «Мої будинки» список наступний:

– чи дає вона зрозуміти користувачеві навіщо вона потрібна;

– чи корисна вона користувачам, які бували тут раніше і які знають, що саме вони шукають.

Для глобальної навігації список наступний:

– чи збалансовані ширина і глибина навігації;

– чи зрозумілі назви посилань і пунктів меню;

– чи використовується одноманітна навігація на всіх сторінках сайту.

Для контекстної навігації список наступний:

– чи зрозуміло, на якій сторінці сайту знаходиться користувач;

– чи легко відрізнити один рівень заголовків від іншого;

– чи використовується на сайті прийнятна і одноманітна мова текстів;

– як знайти потрібну користувачу сторінку;

– як повернутися на головну сторінку або почати все з самого початку.

Для доступності для користувачів список евристик наступний:

– чи ламається яким-небудь образом компонування сторінки при збільшенні розміру шрифту;

– чи достатньо контрастні та яскраві кольори на сторінках сайту;

– чи використовується тільки колір для виділення критичної інформації;

– чи використовується затримка в випадаючих меню.

Для доступності для пристроїв список наступний:

– чи можна працювати з матеріалами сайту при відключених зображеннях або при відсутності підтримки їх виведення на екран;

– чи працює сайт у вікнах різних розмірів.

3.5 Огляд розробленого веб-додатку

Сторінка «Особистий кабінет» (рис. 3.14) має таблицю з потрібною інформацією для користувача про його будинок, під’їзд, квартиру та інформацію про прив’язані електронні ключі. Вид вікна з інформацією про ключі наведений нижче.

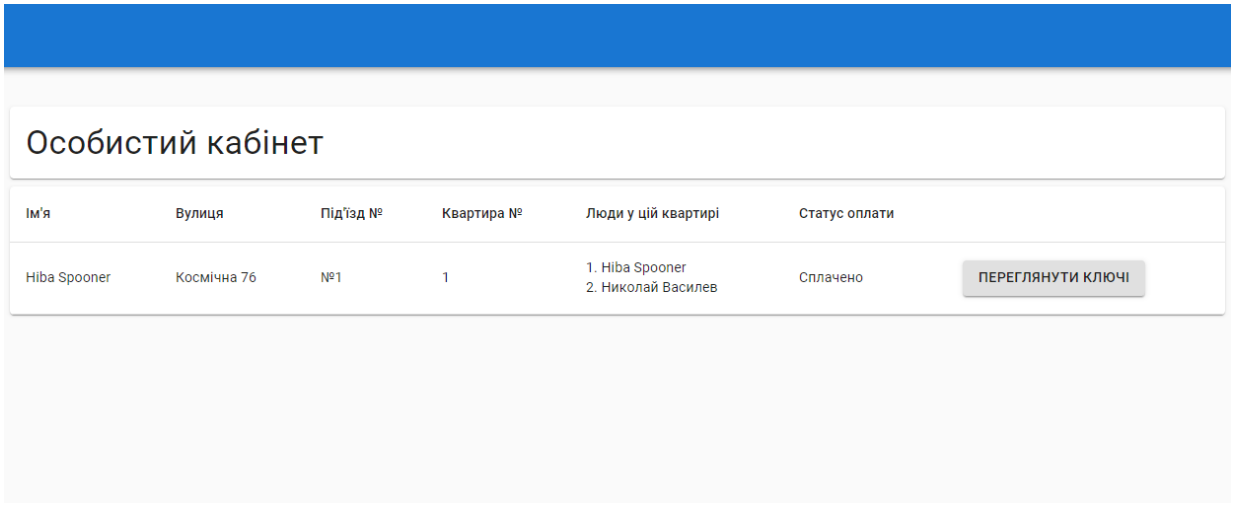

Рисунок 3.14 – Сторінка «Особистий кабінет»

Сторінка «Мої будинки» (рис. 3.15) для керуючих ОСББ має два види: список або мапа. На цій сторінці користувач може обрати будинок, щодо якого йому потрібно переглянути інформацію.

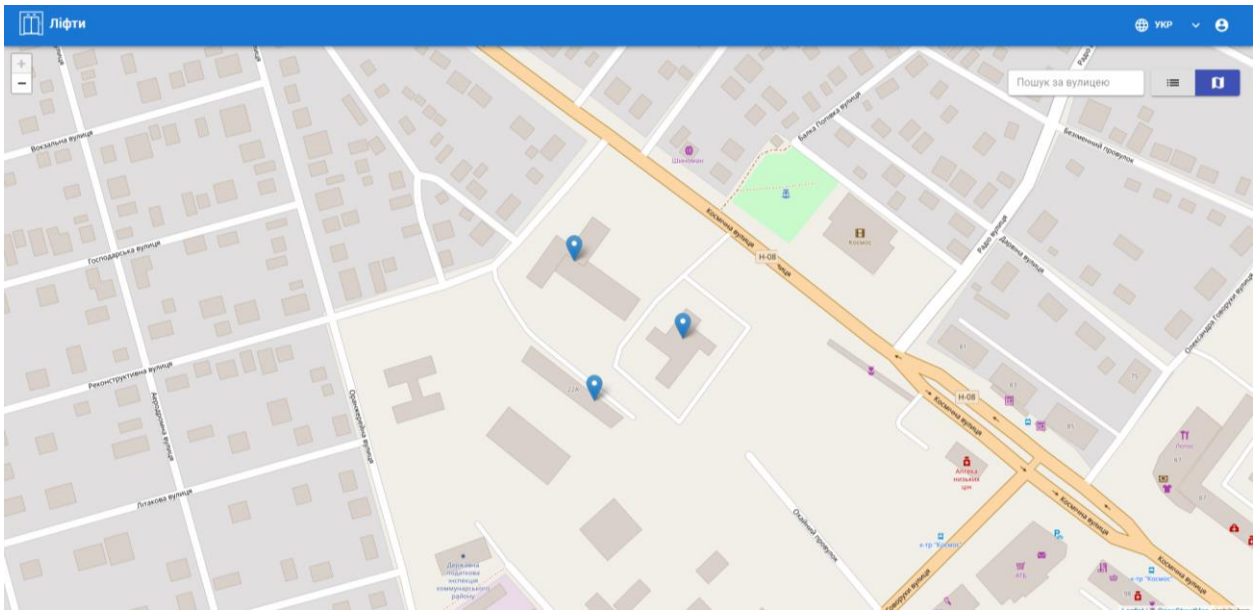

Рисунок 3.15 – Сторінка «Мої будинки». Вид «Мапа»

У верхньому правому кутку знаходиться перемикач видів та пошук за будинками (рис. 3.16).

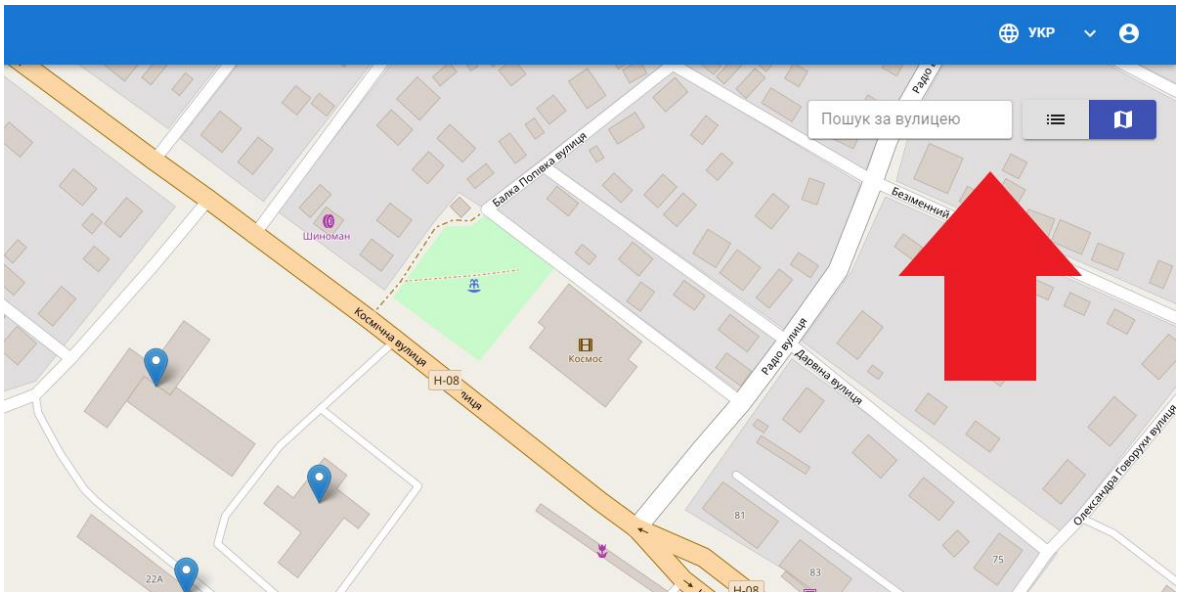

Рисунок 3.16 – Перемикач видів «Список» обо «Мапа» та пошук за будинками

Перейти до обраного будинку можна, натиснувши на потрібну мітку на мапі. З’явиться вікно (рис. 3.17), у якому буде вказана адреса, за якою знаходиться будинок, та кнопка «Переглянути під’їзди».

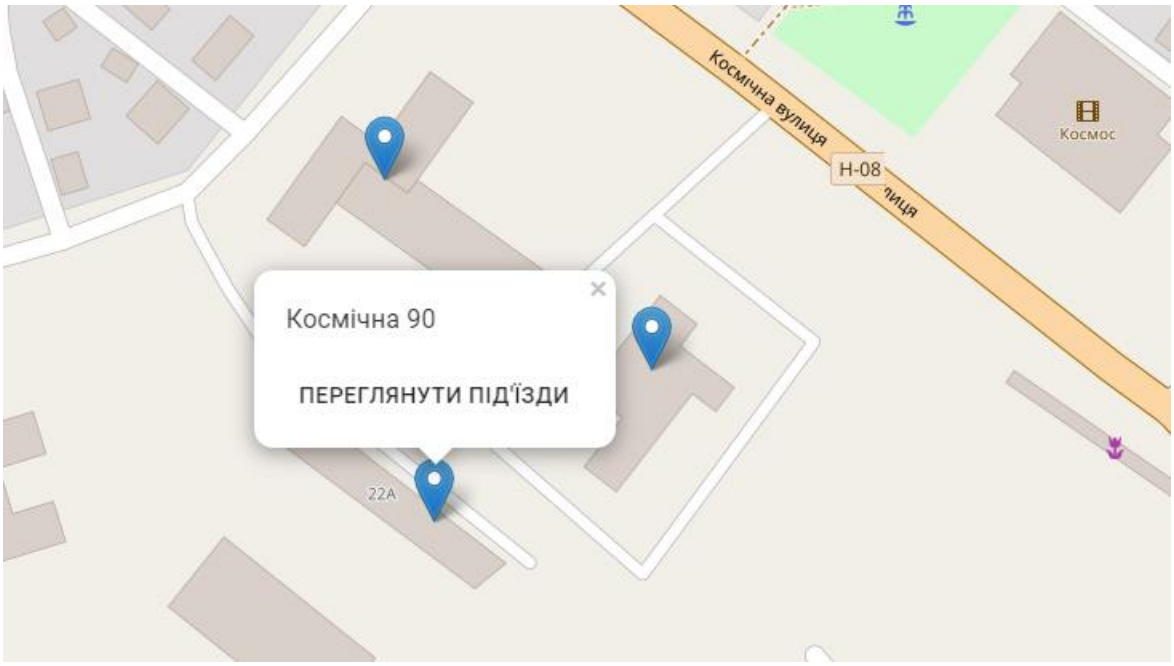

Рисунок 3.17 – Вікно з інформацію про обраний будинок

Також за необхідності, користувач може переключити режим відображення на список. З’явиться список з усіма будинками (рис. 3.18), які підпорядковані користувачеві.

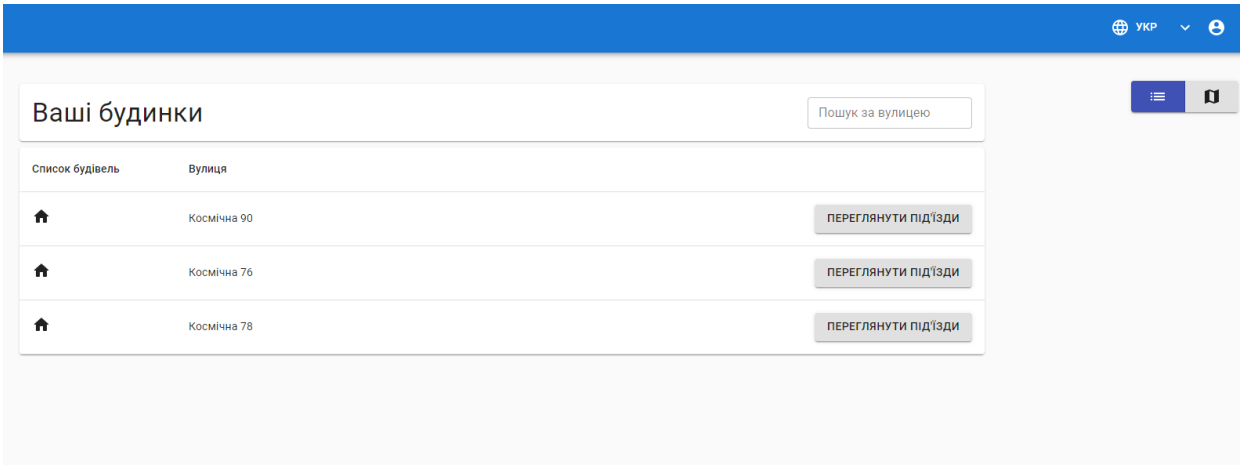

Рисунок 3.18 – Сторінка «Мої будинки». Вид «Список»

Коли користувач натискає на кнопку «Переглянути під’їзди», незалежно від того де, у вікні на мапі або у списку, його перенаправляє на сторінку з під’їздами та квартирами обраного будинку (рис. 3.19).

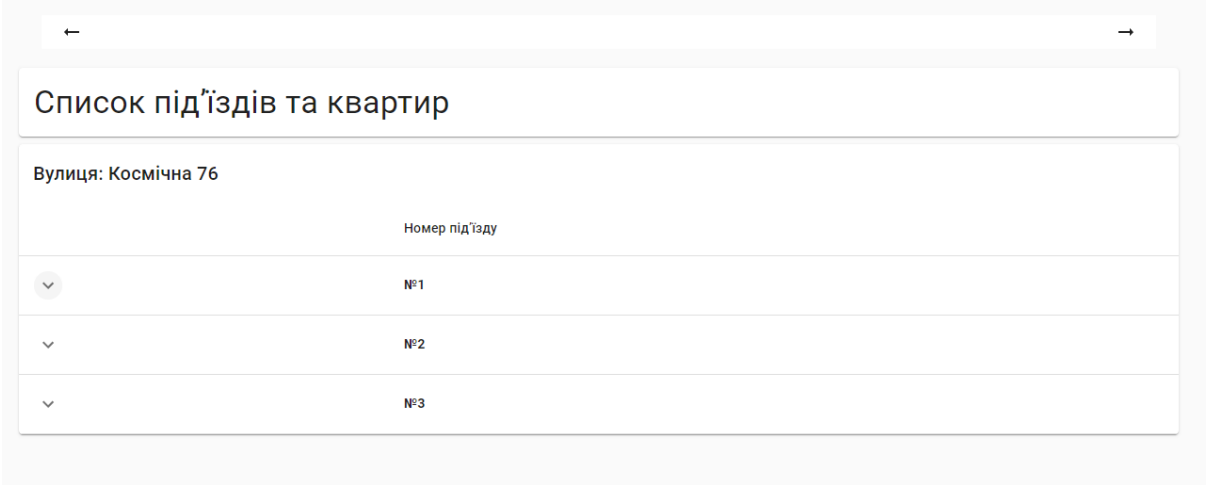

Рисунок 3.19 – Список під’їздів у обраному будинку

Щоб побачити список квартир у під’їзді, треба натиснути на рядок з потрібним під’їздом. Рядок розвернеться та дані про квартири будуть динамічно завантажені. На рис. 3.20 зображений розвернутий рядок зі списком квартир.

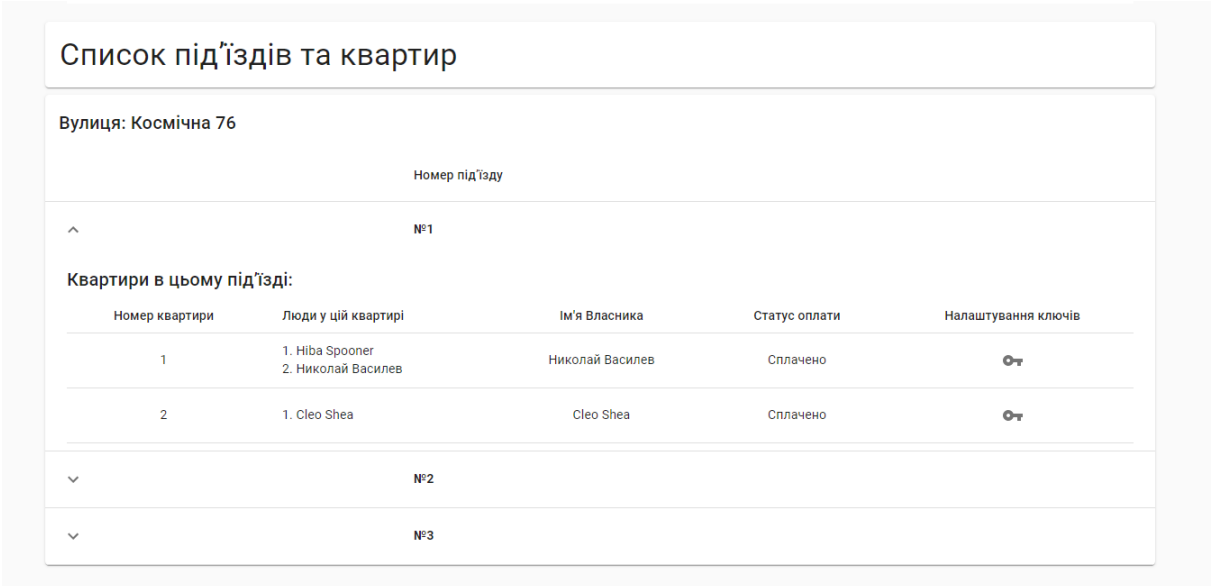

Рисунок 3.20 – Список квартир у обраному під’їзді

У рядку з інформацією про квартиру виводиться номер квартири, поверх, кількість людей, яка проживає в квартирі, ім’я власника квартири, статус оплати та кнопка, при кліку на яку з’являється вікно (рис. 3.21), в якому можна налаштувати ключі для обраної квартири.

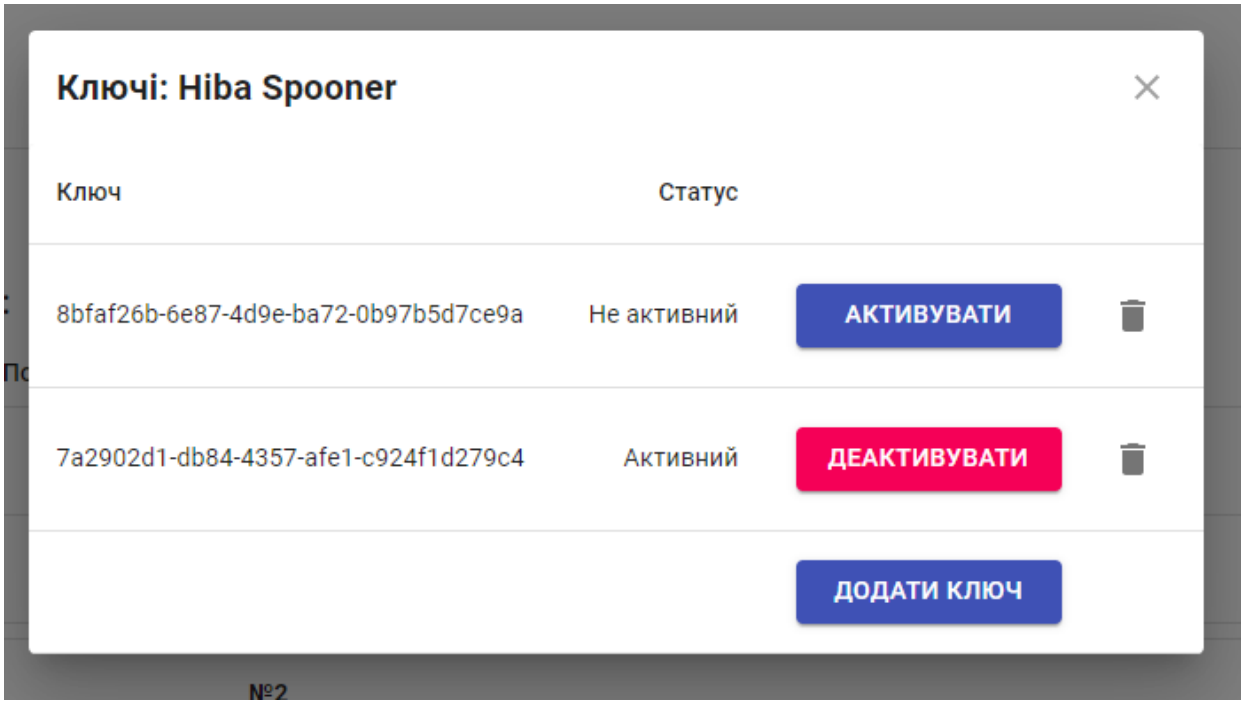

Рисунок 3.21 – Вікно налаштування електронних ключів

У цьому вікні відображаються усі ключі, які прив’язані до конкретної квартири, їх поточний статус та кнопки, за допомогою яких керуючий ОСББ може активувати або деактивувати ключі жильців, видалити застарілі або непотрібні ключі або додати нові. При кліку на кнопку «Додати ключ» з’являється форма (рис. 3.22), де можна додати новий ключ. Користувач вводить потрібні дані або загружає файл з електронним ключем, натискає кнопку «Зберегти», і новий ключ додається до списку існуючих. Новостворені ключі створюються деактивованими за замовчуванням.

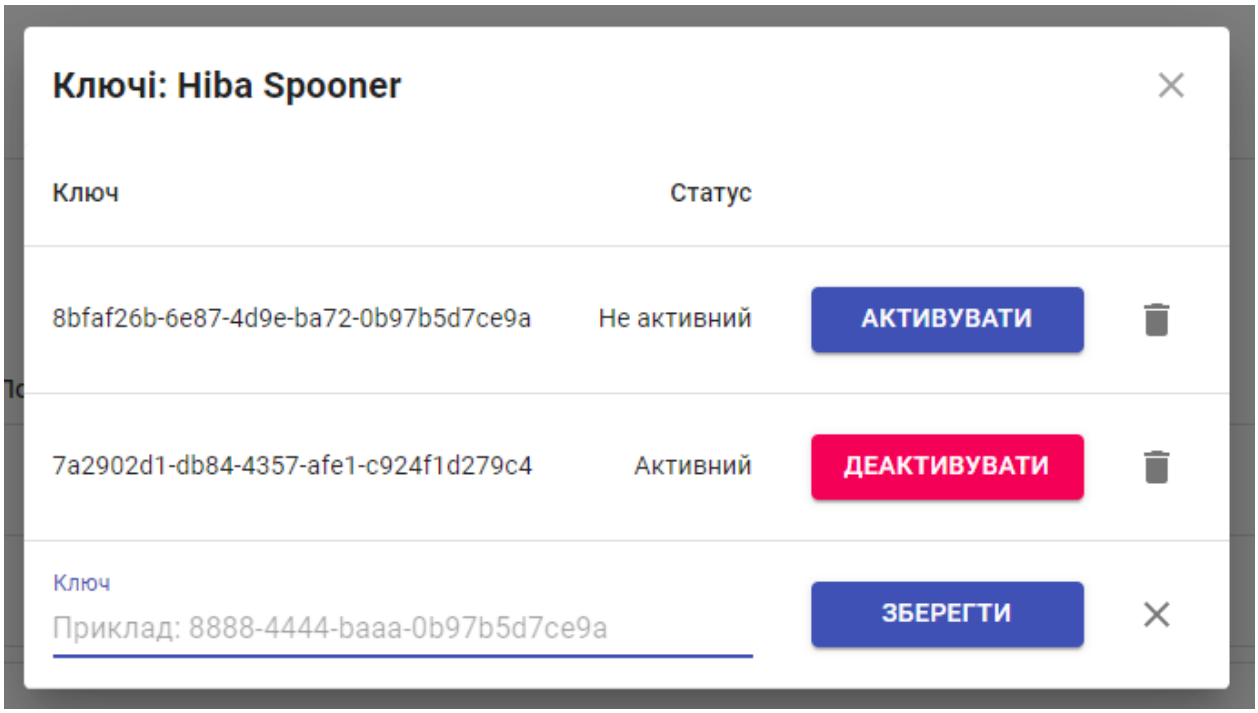

Рисунок 3.22 – Форма додавання нового ключа

Також схоже вікно (рис. 3.23) є в особистому кабінеті у звичайного користувача. Воно з’являється при кліку на кнопку «Переглянути ключі». У цьому вікні виводиться ім’я ключа та його статус.

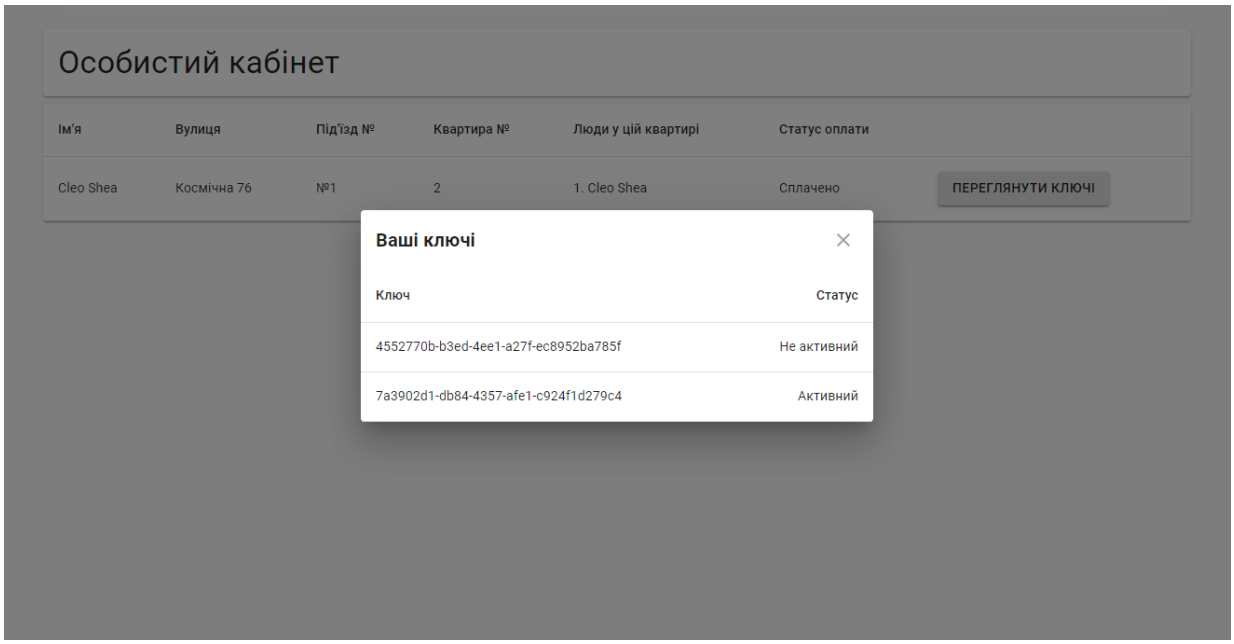

Рисунок 3.23 – Вікно з інформацію про прив’язані ключі користувача

Щоб увійти у свій обліковий запис, користувачеві потрібно ввести свій номер телефону та пароль у формі логіну (рис. 3.24), пройти Google reCAPTCHA та натиснути кнопку «Увійти».

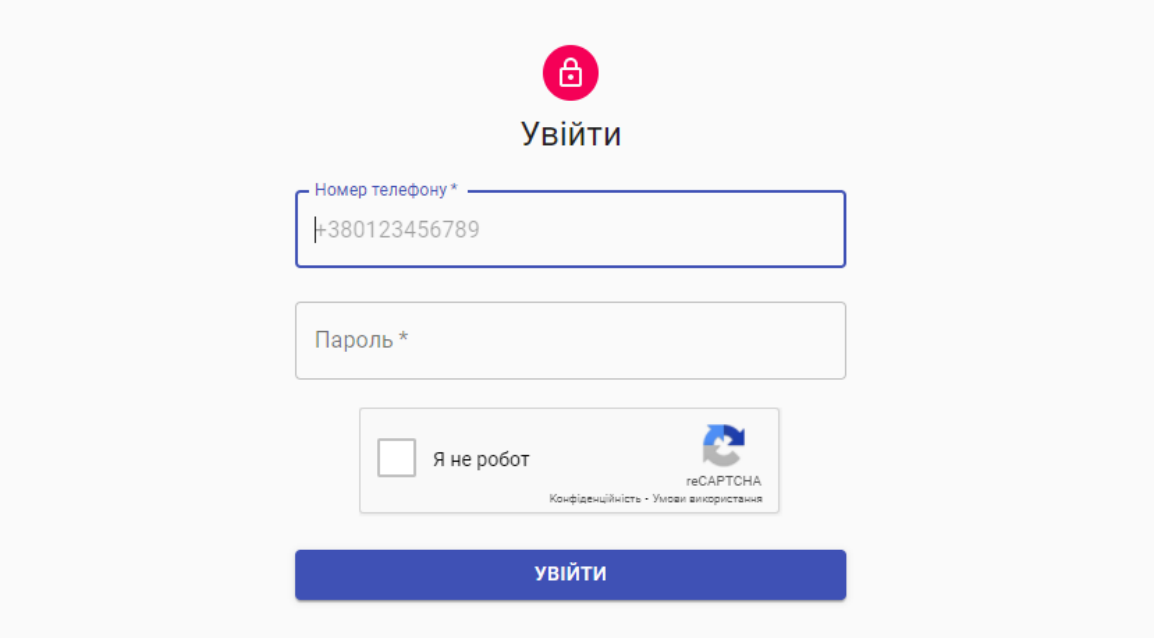

Рисунок 3.24 – Форма авторизації. Перший крок

Після цього кроку користувача перенаправляє на другий етап авторизації, де потрібно ввести код, який прийде в прив’язаний телеграм аккаунт. Форма вводу коду зображена на рис. 3.25.

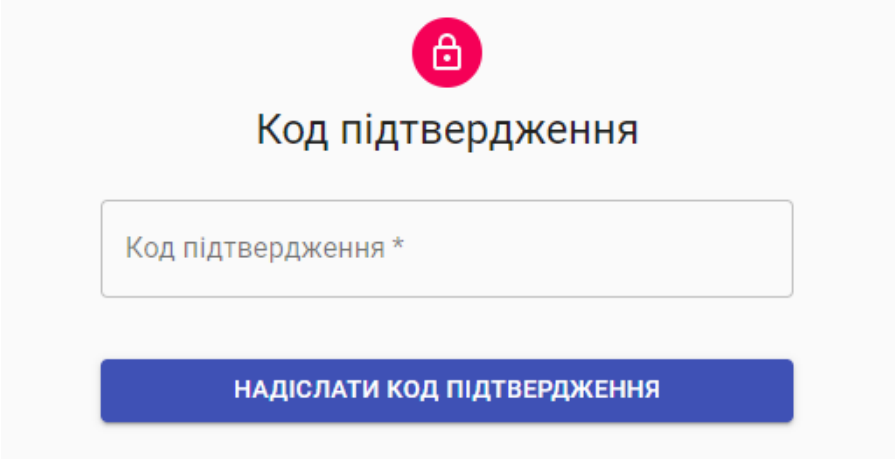

Рисунок 3.25 – Форма авторизації. Другий крок

3.6 Тестування додатку на предмет доступності

Щоб переконатися, що розроблюваний веб-додаток можна використовувати людям з обмеженими можливостями, треба провести тестування кожної сторінки за допомогою Lighthouse. Результати тестування сторінки «Мої будинки» з видом «Мапа» наведені на рис. 3.26. Результати тестування сторінки «Мої будинки» з видом «Список» наведені на рис. 3.27. Результати тестування списку під'їздів та квартир наведені на рис. 3.28. Результат тестування сторінки «Особистий кабінет» наведений на рис. 3.29. Результат тестування сторінки авторизації наведений на рис. 3.30.

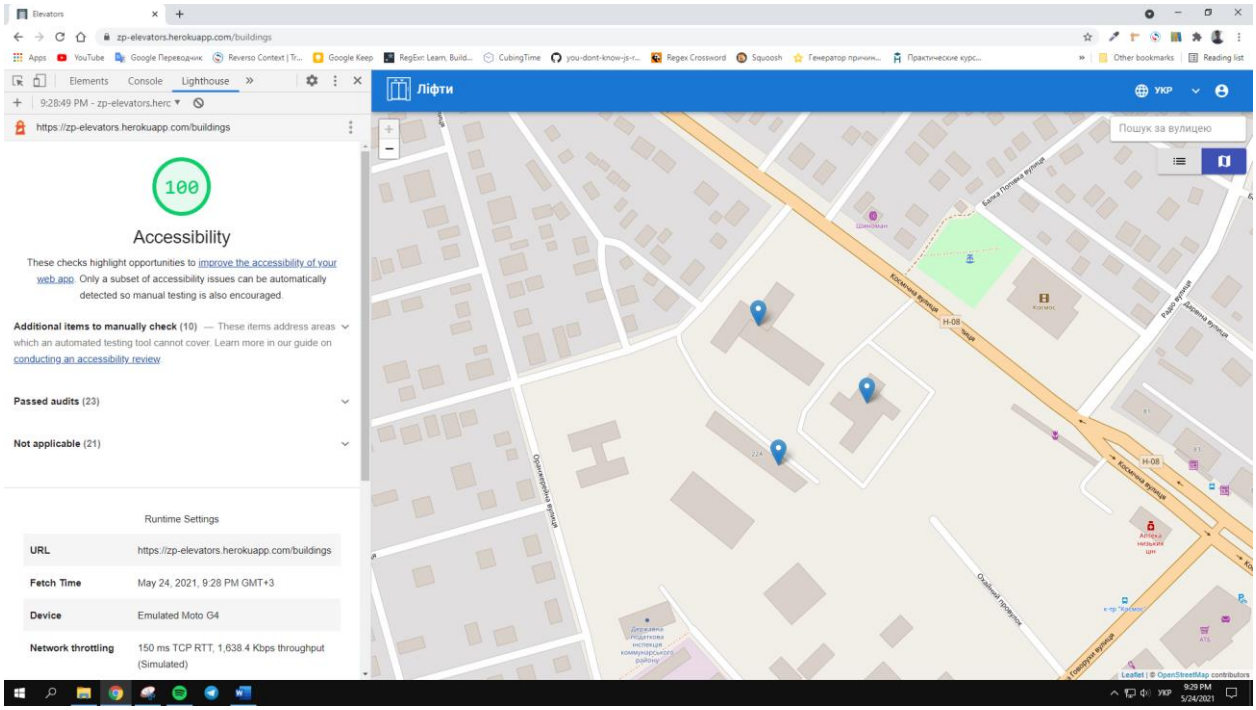

Рисунок 3.26 – Результат тестування сторінки «Мої будинки» з видом «Мапа»

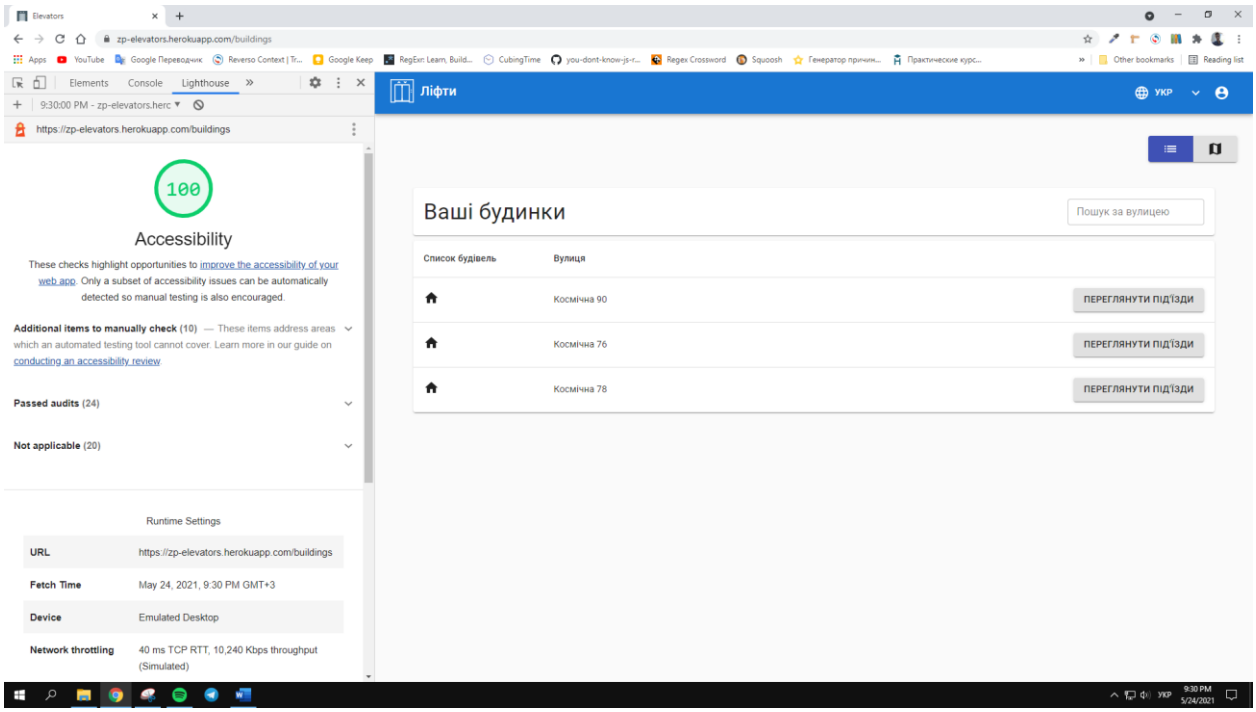

Рисунок 3.27 – Результат тестування сторінки «Мої будинки» з видом «Список»

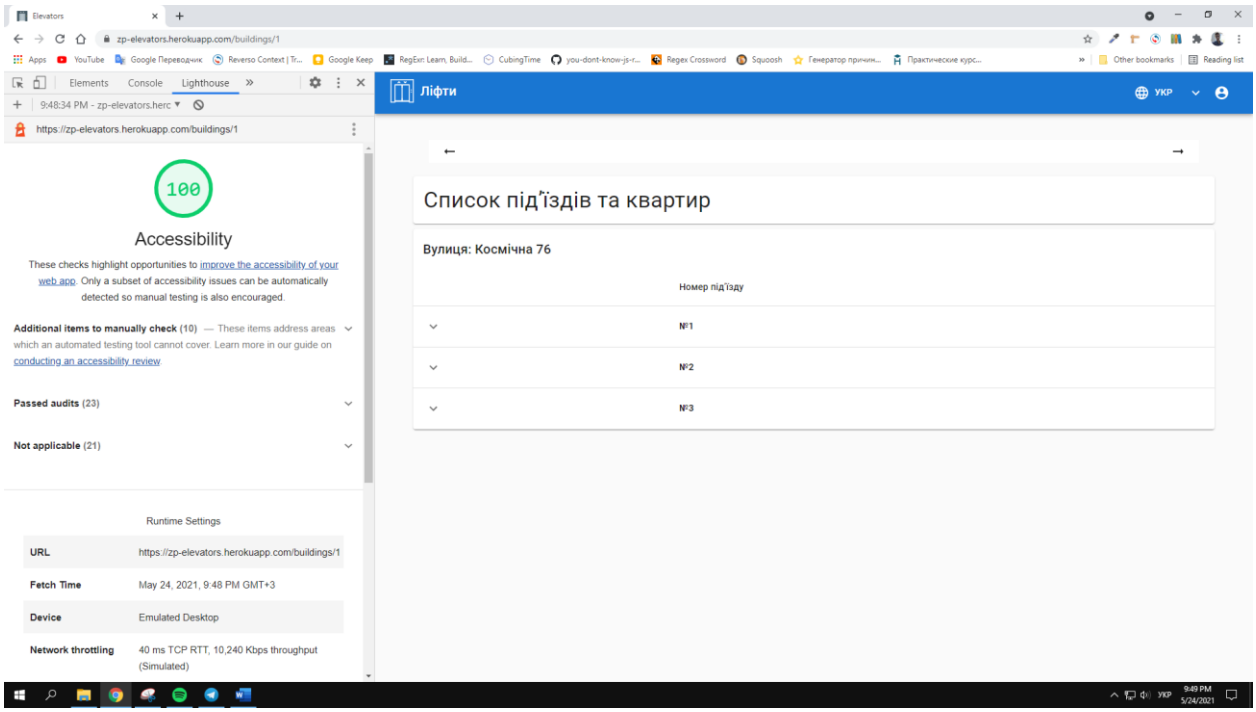

Рисунок 3.28 – Результат тестування списку під'їздів та квартир

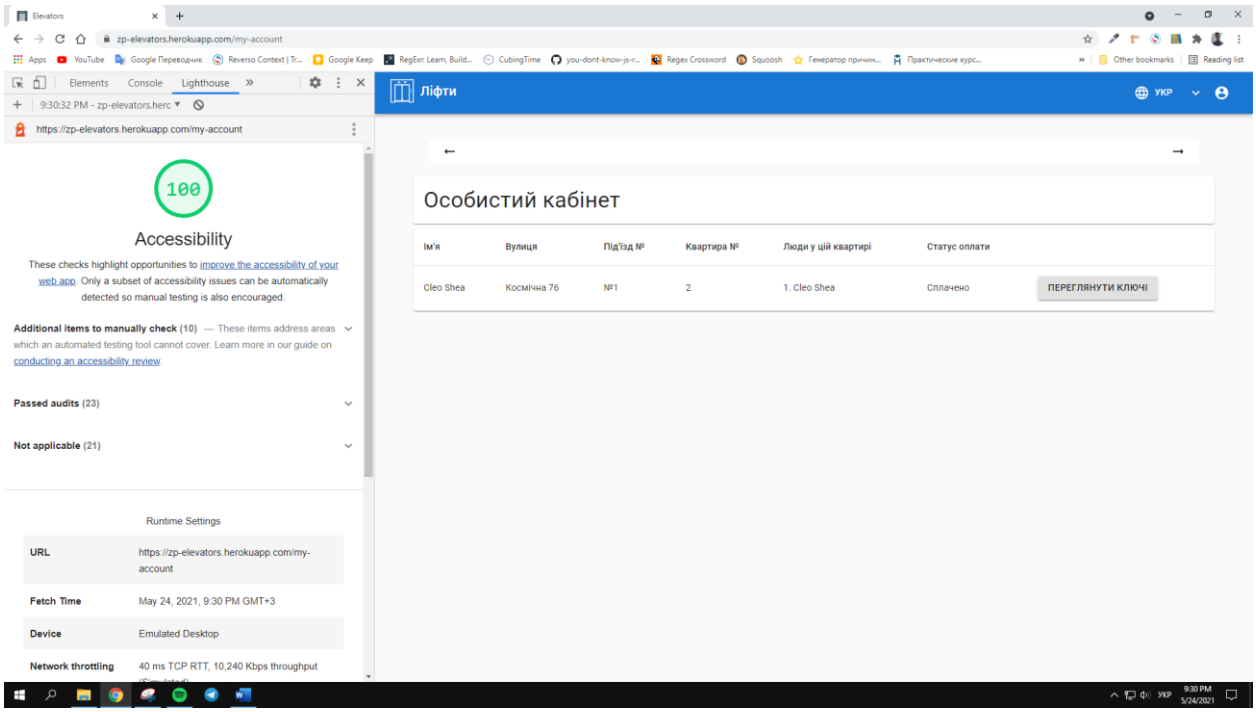

Рисунок 3.29 – Результат тестування сторінки «Особистий кабінет»

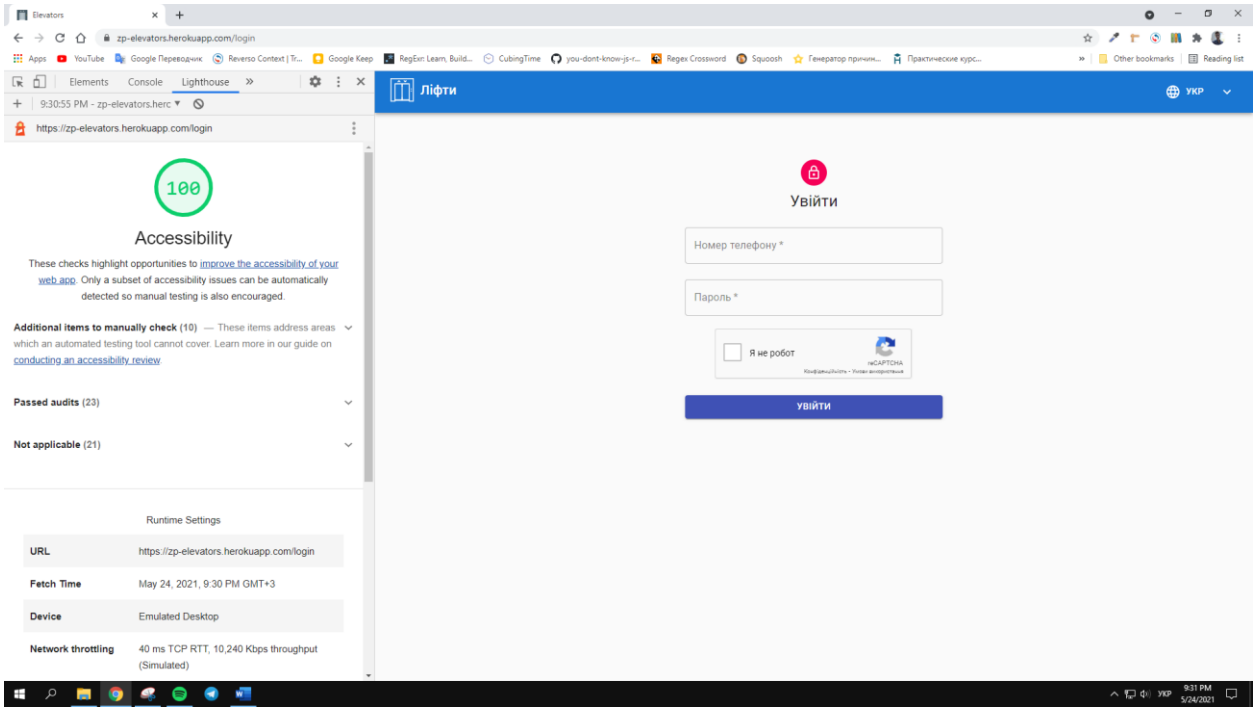

Рисунок 3.30 – Результат тестування сторінки авторизації

3.7 Тестування додатку на мобільних пристроях

Щоб переконатися, що додаток можна використовувати з мобільних телефонів або планшетів, треба у інструментах розробника увімкнути режим симуляції мобільного пристрою та перевірити вигляд усіх сторінок. У якості мобільного пристрою використовується iPhone X з роздільною здатністю 2436 на 1125 пікселів.

На рис. 3.31 зображений вигляд сторінки авторизації на екрані iPhone X. На рис. 3.32 зображений вигляд сторінки «Мої будинки» з видом «Мапа» на екрані iPhone X . На рис. 3.33 зображений вигляд сторінки «Мої будинки» з видом «Список» на екрані iPhone X . На рис. 3.34 зображений вигляд сторінки «Особистий кабінет» на екрані iPhone X . На рис. 3.35 зображений вигляд списку з під'їздами та квартирами на екрані iPhone X . На рис. 3.36 зображений вигляд вікна з налаштуваннями ключів екрані iPhone X .

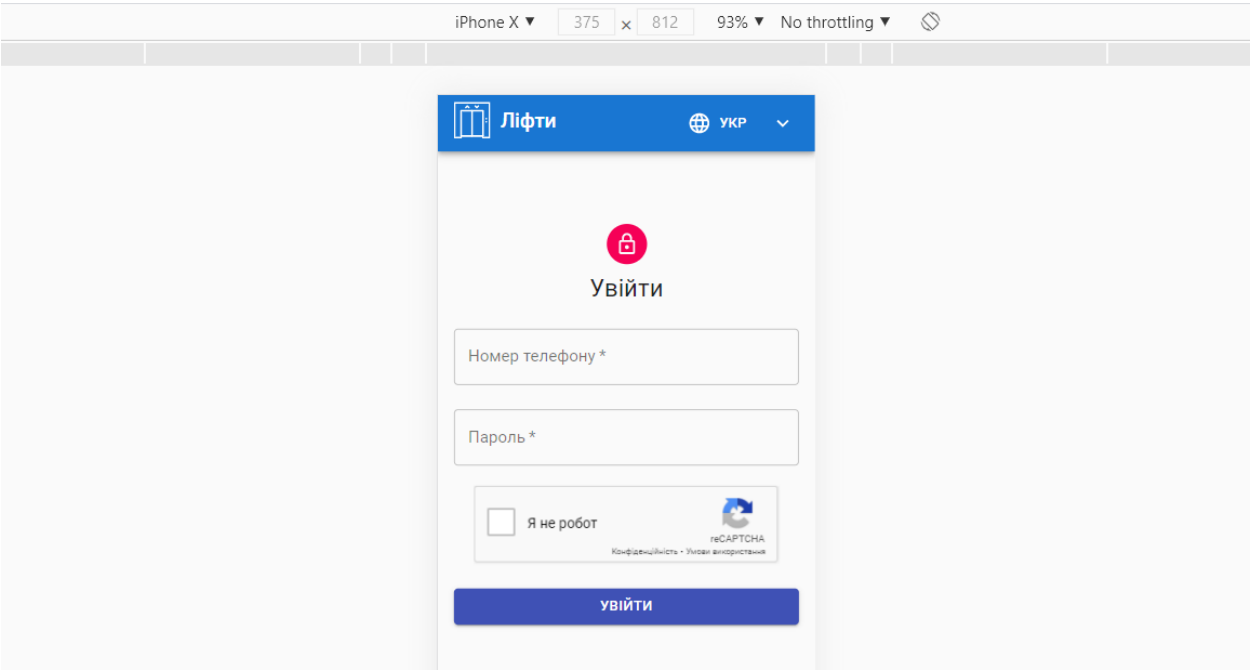

Рисунок 3.31 – Сторінка авторизації

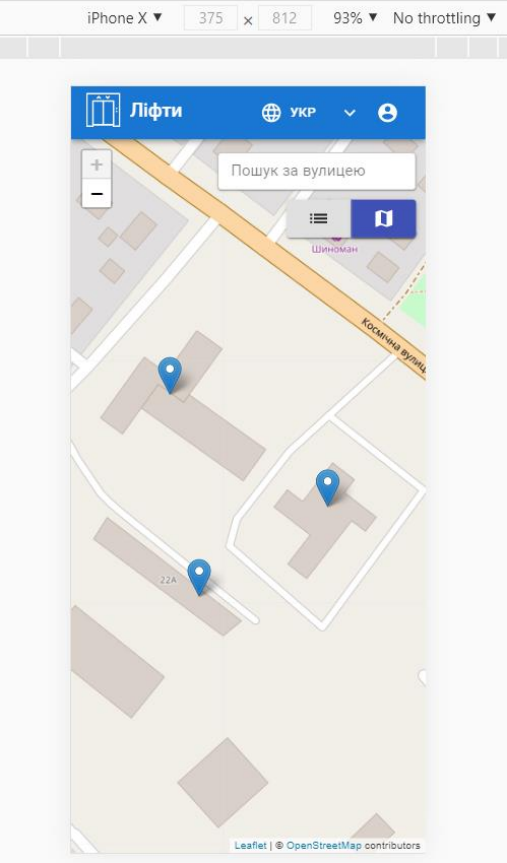

Рисунок 3.32 – Сторінка «Мої будинки» з видом «Мапа»

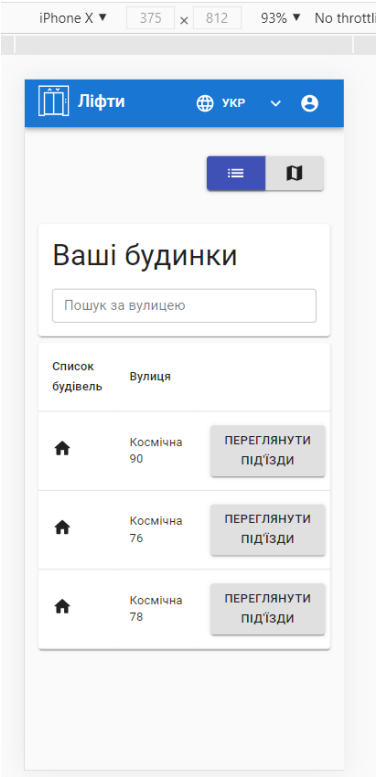

Рисунок 3.33 – Сторінка «Мої будинки» з видом «Список»

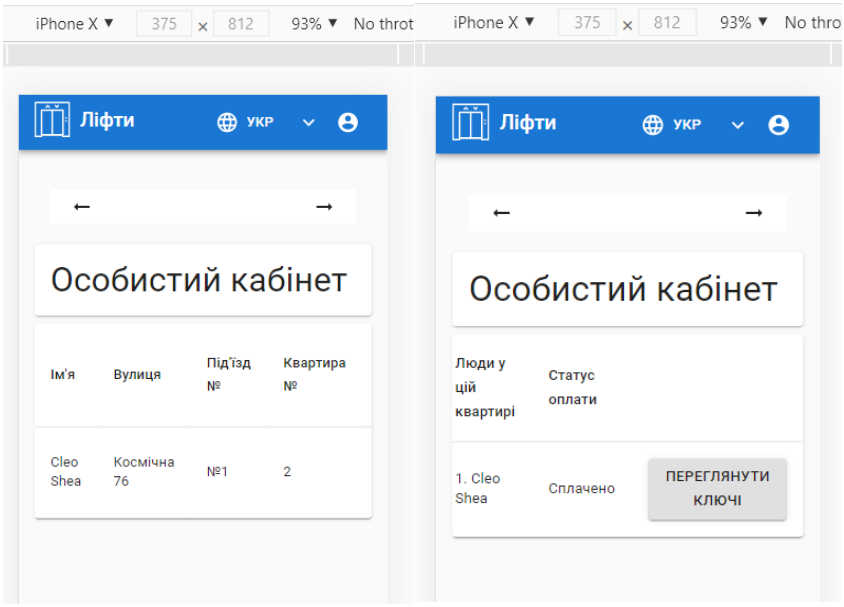

Рисунок 3.34 – Сторінка «Особистий кабінет»

Варто сказати, що на всіх таблицях на маленьких екранах буде присутній горизонтальний скрол. Сам по собі він не є гарною практикою, але в цьому випадку він необхідний і потрібен, щоб частина контенту не обрізалася та була доступною користувачеві.

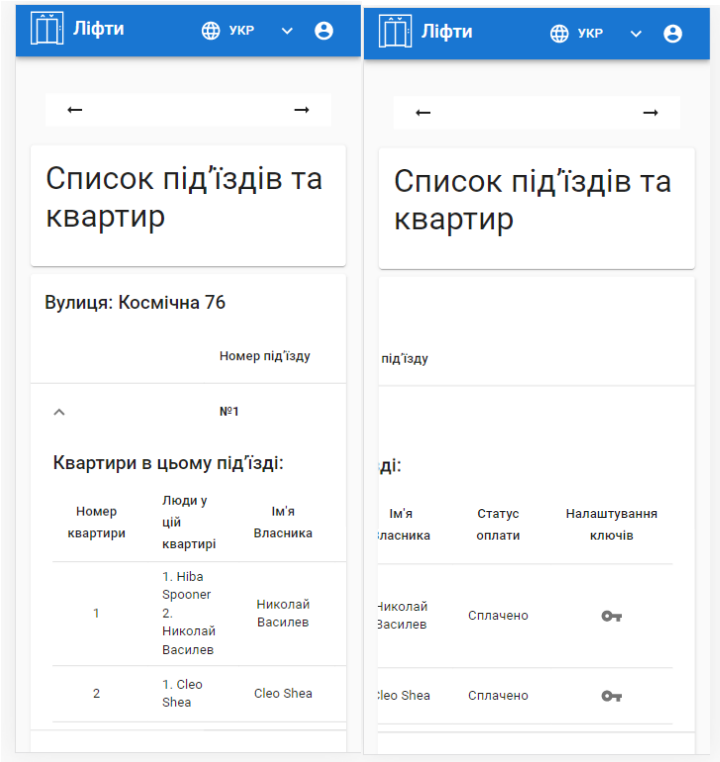

Рисунок 3.35 – Список з під’їздами та квартирами

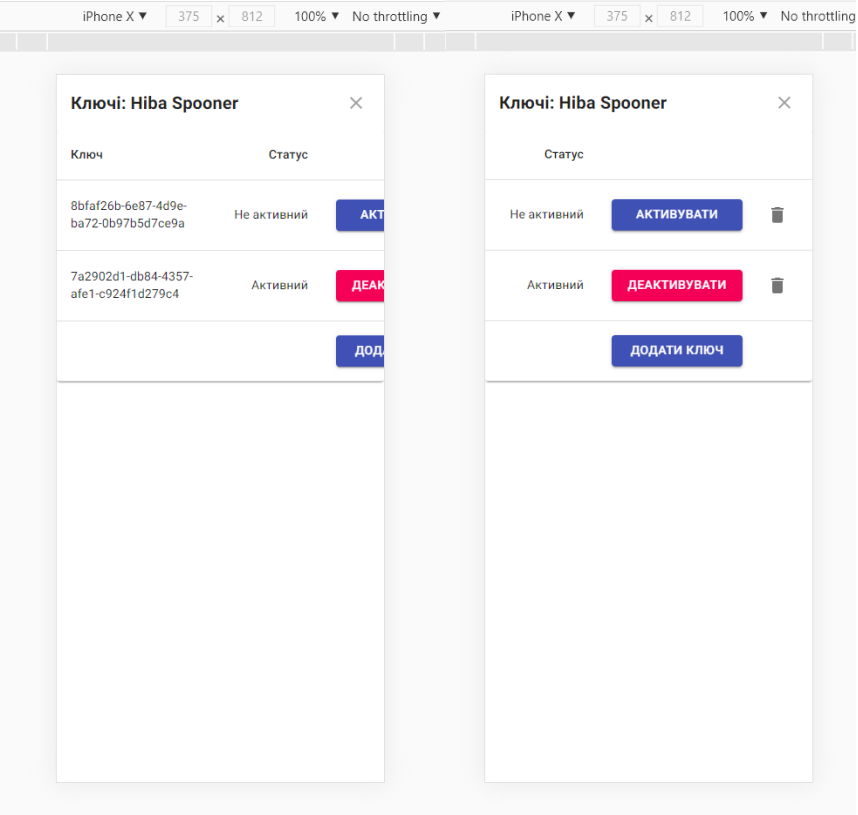

Рисунок 3.36 – Вікно з налаштуваннями ключей

ВИСНОВКИ

Рівень розвитку інформаційних технологій роблять веб-додатки доступним засобом надання послуг широкому колу користувачів.

В результаті дипломного проектування розроблено веб-додаток, що дозволяє людям, які проживають у багатоповерхових будинках зі встановленою системою диспетчеризації та керування ліфтами, переглядати інформацію про їх електронні ключі, а керуючим цими будинками – керувати електронними ключами людей, які там проживають. Даний веб-додаток значно заощаджує час, який раніше люди витрачали на вирішення питань, пов’язаних з користуванням ліфтами.

Веб-додаток створено за допомогою сучасних технологій та з використанням добрих практик, завдяки чому є кросбраузерним і кросплатформенним та може бути легко розширений за потребою.

Веб-додаток має ергономічний інтерфейс, що враховує особливості сприйняття і переробки інформації, а також аспекти роботи користувача з системою. Це зменшує час освоєння системи і підвищує загальну продуктивність роботи користувача.

Загалом у кваліфікаційній роботі бакалавра:

– проведено аналіз існуючих систем керування доступом до ліфтів;

– проведено аналіз сучасних технологій для розробки веб-додатків;

– проведено розробку веб-додатку для перегляду інформації про ліфти;

– проведено тестування на предмет доступності додатку для людей з обмеженими можливостями;

– проведено тестування на девайсах з маленькими екранами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Как работают веб-приложения [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/450282/

2. Фреймворк [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA

3. Сравнение Vue с другими фреймворками [Електронний ресурс]. — Режим доступу: https://ru.vuejs.org/v2/guide/comparison.html

4. Что такое доступность сайта и как её проверить [Електронний ресурс]. — Режим доступу: https://tproger.ru/articles/chto-takoe-dostupnost-sajta-i-kak-ejo-proverit/

5. Клиент-сервер [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Client-Server_overview

6. Разница между HTTP и HTTPS [Електронний ресурс]. — Режим доступу: https://hostiq.ua/wiki/http-https/

7. What is JSON? A better format for data exchange [Електронний ресурс]. — Режим доступу: https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html

8. Все об UI-китах [Електронний ресурс]. — Режим доступу: https://gruzdevv.ru/stati/ui-kit/

9. Most popular React Component UI Libraries (2021) [Електронний ресурс]. — Режим доступу: https://www.kindacode.com/article/most-popular-react-component-ui-libraries/

10. Кобб, Г. Быстрое тестирование / Г. Кобб, К. Браун, Р. Калбертсон. — М.: Вильямс, 2002. — 384 с.

11. Usable information technology [Електронний ресурс]. — Режим доступу: http://www.useit.com/